Høgskulen
på Vestlandet

BACHELOROPPGAVE

Prototype kabeldrevet robot

Prototype cable driven robot

Jon Eilert Liane, Tor Erik Haavik og

Bjørnar Brask Sittlinger

HO2-300 Bacheloroppgåve

Fakultet for ingeniør- og naturvitenskap

20.05.2019

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 10.*

TITTEL	RAPPORTNR	DATO
Hovedprosjekt H02-300-1 19V Sluttrapport		12.05.19
PROSJEKTTITTEL	**TILGJENGE**	**TAL SIDER**
Kabeldrevet robot	Åpen	90
FORFATTARAR	**ANSVARLEGE RETTLEIARARRETTLEIARAR**	
Jon Eilert Liane, Tor Erik Haavik og Bjørnar Brask Sittlinger	Olav Sande, Joar Sande, Preben Gråberg Nes og John Helge Vårdal	
OPPDRAGSGJEVAR		
En To Tre Teknikk		

SAMANDRAG

I denne oppgaven har oppdragsgiver satt som mål å undersøke styringen og posisjonering tilknyttet en kabeldrevet robot. En kabeldrevet robot er et system som kan flytte en plattform i rom ved hjelp av kabler.

Det er blitt bestemt å undersøke to styringsalgoritmer. I rapporten er det beskrevet to styringer. Den ene involverer en manuell styring mens den andre er automatisk. Det er også utviklet to kommunikasjonslag. Det ene brukes der protypen trenger sanntidskommunikasjon, det andre tar hånd om den kommunikasjon som ikke har behov for det. Kommentert kildekode ligger vedlagt.

SUMMARY

The goal of this thesis is to investigate steering and positioning related to a cable driven robot. A cable-driven robot is a system that can move a platform in a room by means of cables.

It has been decided to investigate two steering algorithms. This report describes two means of controlling the robot. One involves a manual steering while the other one is automatic. Two communication layers have also been developed. One is used where the protype needs real-time communication, and others take care of the communication that does not require it. Commented source code related to this project is included in the appendix.

EMNEORD

Automasjon, bachelor, kabledrevet robot, styring, posisjonering, prototype

Forord

Denne bacheloroppgave utgjør siste del av fulltidsstudiet Automatiseringsteknikk. Bacheloroppgaven er en del av et samarbeid mellom En To Tre teknikk og Høyskolen på Vestlandet (HVL). Arbeidet er utført ved Høyskolen på Vestlandet campus Førde.

En stor takk til Virkemiddel for regional FoU og innovasjon (VRI) og HVL, for finansiering av oppgaven, Øyrane Videregående skole for produsering av de mekaniske delene til prosjektet, oppdragsgiver En To Tre Teknikk for oppgaven og veiledning.

Til slutt vil vi takke styringsgruppen som består av førstelektor Olav Sande, høyskolelektor Joar Sande og førstemanusist Preben Gråberg Nes for å stå tilgjengelig med erfaringer, kunnskap og veiledning.

Førde den 20.05.2019

Jon Eilert Liane	Tor Erik Haavik	Bjørnar Brask Sittlinger

Sammendrag

Denne bacheloroppgaven er blitt utført som en del av et større samarbeid mellom En To Tre Teknikk og HVL. Der målet er å modernisere de tradisjonelle metodene som benyttes til uthenting av tømmer. De ser for seg å bruke en kabeldrevet robot til dette formålet. Dette kan muligens effektivisere og bedre sikkerheten på hogstarealet. I denne oppgaven har oppdragsgiver satt som mål å undersøke styringen og posisjonering tilknyttet en kabeldrevet robot. Derfor er det blitt valgt å utvikle en protype.

En kabeldrevet robot er et system som kan flytte en plattform i et rom ved hjelp av kabler. Det blir beskrevet en tradisjonell taubane kontra en kabeldrevet robot til bruk av uthenting av tømmer. Videre i rapporten blir også utviklingen av prototypen nevnt, og hvordan delene ble produsert og montert.

Det er blitt bestemt å undersøke to styringsalgoritmer. Den første involverer en manuellstyring med en håndkontroller. Den andre tillater en operatør å definere et settpunkt i et styrings og overvåkningssystem. Her har operatøren mulighet til visuell overvåkning av systemet. Systemet som har blitt utviklet er ikke ferdigstilt og må forskes videre på.

Kommunikasjonen til prototypen har to lag. Det ene laget er til sanntids kommunikasjon som er essensiell for driften av anlegget. Det andre lagets kommunikasjon er ikke essensiell for anleggets drift. Kildekoden som tilhører programvare som er utviklet i forbindelse med prosjektet er ikke beskrevet i rapporten men er kommentert i vedleggene.

Rapporten er delt inn i seks hoveddeler og fjorten kapitler. Kapitlene beskriver de ulike temaene som er blitt fokusert på.

Summary

This bachelor's thesis is a part of a greater collaboration between En To Tre Teknikk and HVL. The aim is to investigate whether it is possible to modernize the traditional cableway system by utilizing a cable-driven robot. It could potentially increase both the work area and the safety of the felling area. The teams mission is to investigate possibilities related to steering and positioning of a cable driven robot. Therefore, it has been chosen to develop a protype that can be used to examine the steering and the positioning of a cable driven robot.

A cable-driven robot is a system that can move a platform in a work area by means of cables. A traditional cableway is described against a cable-driven robot for the use of timber extraction. Moreover, development of the prototype is also mentioned, and how the parts were produced and assembled.

Two algorithms on how to control the system is described. One involves manual control with a gamepad. The second gives the operator the opportunity to define a set point in a control and surveillance system. Here the operator has the ability to visually monitor the system. The system we have developed is not finalized and must be further developed by a future team.

The communication to the prototype has two layers. The first layer is for real-time communication that is essential for the operation of the system, and the other layer of communication is not essential for the prototype’s operation.

The source code belonging to the software developed in conjunction with the project is not described in the report but is commented in the appendix.

The report is divided into six main sections and fourteen chapters. Chapters describe the various topics that have been focused on.

Innholdsfortegnelse

Side

FORORD ..... **II**

SAMMENDRAG ..... **III**

SUMMARY ..... **IV**

TABELLER ..... **VII**

FIGURLISTE ..... **VII**

1 INNLEDNING **1**

2 BAKGRUNN **4**

2.1 TRADISJONELT TAUBANESYSTEM 4

2.2 KABELDREVET ROBOT - TRE PUNKT SYSTEM 6

3 DESIGN/MODELLERING **8**

3.1 FUSION 360 8

3.2 MODELLERING 8

3.3 KOMPONENTER 10

4 ARBEIDSSTASJON **17**

4.1 HOVEDSTYRINGSENHET DELL PRECISION TOWER 3420 17

4.2 OPPGRADERING AV HOVEDSTYRINGSENHET 18

5 PROSJEKTERINGSPLATTFORM – TWINCAT 3 ..... **19**

5.1 TWINCAT 3 19

6 SCADA APPLIKASJONEN **26**

6.1 KRAV 26

6.2 SPESIFIKASJONER 26

6.3 UTVIKLINGSMILJØ - UNITY 27

6.4 KOMMUNIKASJON 28

6.5 PYTHON SKRIPTET 28

6.6 SCENER 29

6.7 SCADA 32

7 STYRESKAP **33**

Innholdsfortegnelse

Side

7.1 VALG ... 33
7.2 TAVLEN ... 33
7.3 STRØMFORSYNING CP5 - 242 ... 34
7.4 ETHERCAT SLAVEMODULEN EK1100 ... 34
7.5 STEPPERMOTORDRIVEREN - EL7031 ... 35

8 KOMMUNIKASJON ... 36
8.1 TEKNISK NETTVERK ... 36
8.2 ETHERCAT ... 37
8.3 OPEN PLATFORM COMMUNICATION UNIFIED ARCHITECTURE(OPC UA) ... 38
8.4 RUTER ... 39

9 STEPPERMOTORENE ... 40
9.1 STEPPERMOTOREN AS1010-0000 ... 40

10 SMARTSENSOREN ... 41
VALG ... 42
10.1 MIKROKONTROLLER - RASPBERRY PI ... 42
10.2 POTENSIOMETER ... 43
10.3 ULTRALYDSENSOR - HC-SR04 ... 44
10.4 ANALOG TIL DIGITAL KONVERTER - MCP3008 ... 45

11 POSISJONERING ... 46
11.1 TRILATERASJON ... 46
11.2 SENSOR POSISJONERING ... 48

12 STYRING ... 52
12.1 HÅNDKONTROLLER ... 52
12.2 PROGRAM ... 54

13 OPPSUMMERING ... 64

14 REFERANSER ... 66

15 VEDLEGG ... 70

Side

Innholdsfortegnelse

TABELLER

Tabell 1 Manuel kalibrering av skalerings faktor 22

Figurliste

Figur 1 Ferdig konstruert prototype 2
Figur 2 Taubane system til forflytning av tømmer [3] 4
Figur 3 bilde av digitalt sammensatt tårn 9
Figur 4 Printet og sammensatt tricopter ramme 11
Figur 5 Topplokk med braketter for sensorer og trinse 11
Figur 6 Holder for potensiometer 12
Figur 7 Arm til potensiometer for å måle vinkel i Z akse 13
Figur 8 Hjul til potensiometer for å måle vinkel i planet 13
Figur 9 Montert trinse på topplokk 14
Figur 10 Brakett til ultralydsensor 14
Figur 11 Trommel modell 15
Figur 12 Tårn med bunnplate 15
Figur 13 Roterende hode til tårn 16
Figur 14 Steppermotor montert på brakett 16
Figur 15 Arbeidsstasjon hovedstyringsenhet 17
Figur 16 Oppgradering av hovedstyringsenhet 18
Figur 17 Oppsett bilde av konfigurasjon til aksenes simulerte enkoder 21
Figur 18 Bilde av ramping til steppermotoren til prototypen 23
Figur 19 Node-tre OPC UA 25
Figur 20 Visualisering i Unity annen kameravinkel 1 27
Figur 21 Visualisering i Unity kameravinkel 2 27
Figur 22 Unity logo [45] 28
Figur 23 Hovedmeny SCADA applikasjonen 29

Innholdsfortegnelse

Side

Figur 24 Alarm liste 30
Figur 25 Settings scenen 31
Figur 26 Styreskapet 33
Figur 27 Spenningsforsyningen CP5-242 fra PULS Dimension 34
Figur 28 Beckhoff slavemodul EK1100 35
Figur 29 Beckhoff steppermotor driver EL7031 35
Figur 30 Teknisk nettverk 36
Figur 31 Datatelegram i EtherCAT [17] 37
Figur 32 Nettverkstopologi EtherCAT 37
Figur 33 Oppsett av trådløs kommunikasjon 39
Figur 34 Beckhoff steppermotor AS1010-0000 40
Figur 35 Grafisk fremstilling av hvordan smartsensoren er koblet 41
Figur 36 Mikrokontrolleren Raspberry Pi 42
Figur 37 Tårn med roterende hode og smartsensor 43
Figur 38 topplokk med ultralydgiver og Potensiometer 44
Figur 39 Forenklet skisse av skjæringspunkt til kulene i 2D 48
Figur 40 Matematisk skisse av roboten i rommet 50
Figur 41 Projeksjon av roboten i rommet 51
Figur 42 Amerikanske marinen bruk av X-box 360 kontroller [24] 53
Figur 43 Joystick akser 54
Figur 44 Funksjonsblokken MC_Reset [28] 56
Figur 45 Funksjonsblokken MC_Power [29] 56
Figur 46 Funksjonsblokken MC_MoveContinuousAbsolute [30] 57
Figur 47 Funksjonsblokken MC_Stop [32] 58
Figur 48 Funksjonsblokken MC_Halt [33] 58
Figur 49 Plassering av trekantene i koordinatsystemet til håndkontrolleren 62

1 Innledning

På Vestlandet er taubanedrift mye brukt for uthenting av tømmer. Dette kommer tydelig frem i Sogn og Fjordane der hele 50 % av skogen ligger i vanskelig terreng [1].

Denne bacheloroppgaven er blitt utført som en del av et større samarbeide mellom En To Tre Teknikk og HVL. Der målet er å modernisere de tradisjonelle metodene som benyttes til uthenting av tømmer. De ser for seg å bruke en kabeldrevet robot til dette formålet. Dette kommer muligens til å effektivisere og bedre sikkerheten på hogstarealet. I denne oppgaven har oppdragsgiver satt som mål å undersøke styringen og posisjonering tilknyttet en kabeldrevet robot. Derfor er det blitt valgt å utvikle en protype.

En kabeldrevet robot defineres abstrakt som en flyttbar plattform som er koblet opp til et antall kabler. Plattformen blir manipulert ved å endre lengdene på kablene. Plattformen blir i denne oppgaven definert som roboten. Den ideelle applikasjonen til en kabeldrevet robot involverer et stort arbeidsområde og tung last [2]. For å gi leseren et bedre innblikk i hva En To Tre teknikk ønsker å oppnå er en tradisjonell taubane system beskrevet. Det er også gitt en generell beskrivelse på hvordan en kable drevet robot kan bli benyttet til å frakte tømmer.

Bacheloroppgaven gir en utdypende beskrivelse av utviklingen av prototypen. Utviklingen har forekommet i flere steg. Her har det blitt utført et samarbeid mellom Øyrane Videregående Skole. Dette samarbeide gikk ut på at Øyrane Videregående skole utførte den mekaniske delen av oppgaven etter bachelorgruppens design. Det er også produsert diverse deler ved hjelp av 3D printer. Disse delene ble senere montert og resultat kommer frem på figur 1.

Figur 1 Ferdig konstruert prototype

Bachelorgruppen har benyttet et styreskap for å beskytte de essensielle komponentene til systemet. Dette skapet beskytter mot støv, fukt og støt. Det øker også sikkerheten mot berøring av strømledende deler.

I rapporten beskrives to forskjellige styringssystemer. I det første systemet brukes det en håndkontroller for å styre prototypen. Den andre styringen består av et styrings og overvåkningssystem (SCADA) som gir operatør mulighet til å angi ønsket posisjon. Videre gir SCADA applikasjonen operatøren mulighet til å overvåke prototypen visuelt.

Vi har delt enhetskommunikasjonen til prototypen opp i to lag. Det første laget er kommunikasjon som må forgå i sanntid. Dette er kommunikasjonen som er essensiell for systemets drift. Det andre kommunikasjonslaget er ikke essensielt for prototypens drift. Det er også mulig for andre enheter å koble seg på det tekniske nettet og kommunisere med systemet.

Det er utviklet flere programvarer knyttet til styringen i dette prosjektet. Prototypen benytter disse til å overføre data som er essensiell for styringen. Beskrivelse av

kildekoden er ikke gitt i detalj i rapporten, men all kilde kode som er benyttet er godt kommentert i vedleggene.

Prototypen vi har utviklet er ikke ferdigstilt og må videreutvikles av en fremtidig prosjektgruppe. Det anbefales å benytte en steppermotordriver som enten støtter en stepper akse eller en roterende giver. Dette vil kunne gi en mer nøyaktig posisjonering.

Rapporten er delt inn i seks hoveddeler og fjorten kapitler. Kapitlene beskriver de ulike temaer som er fokusert på.

Den første hoveddelen (kapittel 3) gir en innføring i et tradisjonelt taubanesystem kontra en kabeldreven robot.

Den andre hoveddelen (kapittel 4) handler om design. Her fremlegges design og modelleringsprosessen. Det fremlegges en beskrivelse av produksjonen og hensikten til de ulike delene.

Den tredje hoveddelen (kapittel 5, 6, 7 og 8) beskriver hvilke utviklingsplattformer og programvarer som er blitt brukt i prosjektet. Det er også beskrevet hvordan vi har benyttet programvaren. Ikke minst er det gjort rede for det tekniske nettet og hvilke kommunikasjonstyper som er blitt brukt til styringen.

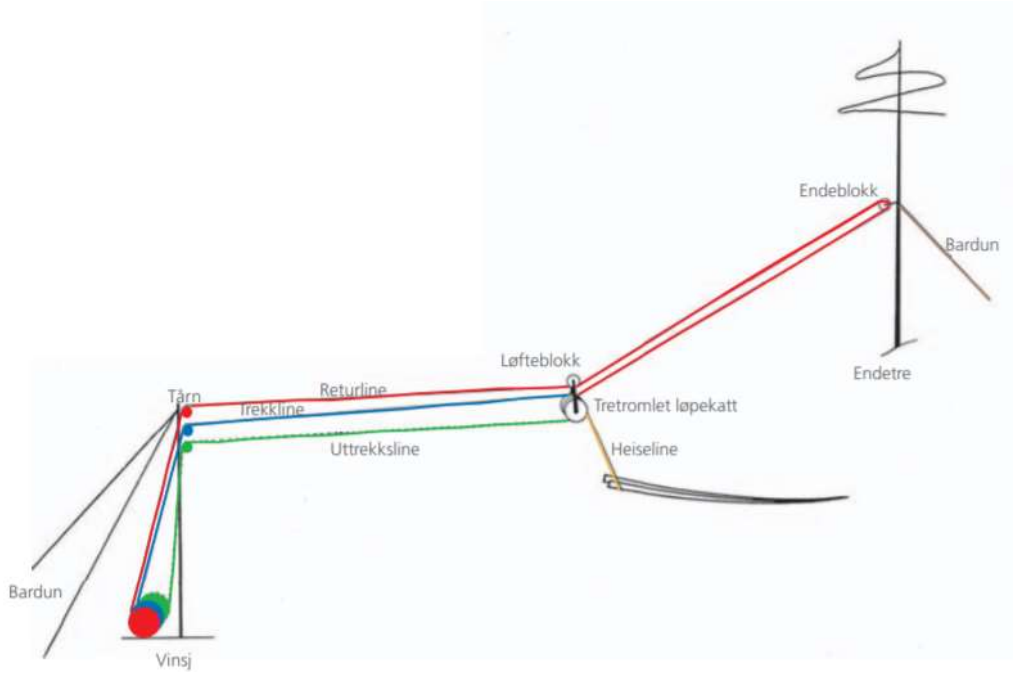
I den fjerde hoveddelen (kapittel 9, 10, 11) blir komponentene som er essensiell for å styre roboten beskrevet. Her blir komponentenes funksjonalitet gjort rede for.

Den femte hoveddelen (kapittel 12 og 13) omhandler en teknisk beskrivelse av styring og posisjonerings algoritmer. Her kommer det også frem hvilke algoritmer som kan benyttes på prototypen. Det beskriver også utførelsen av algoritmene.
I den siste hoveddelen (kapittel 14)fFremlegges resultatet av prosjektet og videreutvikling av prototypen.

2 Bakgrunn

2.1 Tradisjonelt taubanesystem

I dagens tømmerdrift benyttes det ulike taubanesystem. I dette del kapitelet gis det en generell beskrivelse av et tradisjonelt taubanesystem. Det beskrives også ulemper ved et tradisjonelt taubanesystem. Videre blir det gitt en kort innføring i hovedkomponentene i et tradisjonelt taubanesystem. Oppriggingen til et tradisjonelt taubanesystem beskrives også i dette kapitelet. Til slutt gir vi en kortbeskrivelse av styringen til et tradisjonelt taubanesystem.



Figur 2 Taubane system til forflytning av tømmer [3]

2.1.1 Opprigging

Oppriggingen til et tradisjonelt taubanesystem er svært tidkrevende. Operatøren må bardunere basestasjonen og endetreet i flere vinkler. Barduneringen blir gjort i flere vinkler for å fordele kreftene taubanen blir påført under drift. Etter bardunering trekker operatøren returlinen fra basestasjonen gjennom løpekatten. Returlinen føres videre gjennom endeblokken og festes etterpå til løpekatten, se figur 2. Operatøren må også føre en trekkline ut til løpekatten. Trekklines oppgave er å trekke løpekatten langs returlinen.

2.1.2 Hoveddeler og virkemåte

Hoveddelene til et tradisjonelt taubanesystem er basestasjonen, returlinjen og løpekatten. Basestasjonen er hovedpunktet til taubanesystemet. Operatøren styrer hastigheten til vinsjene fra basestasjonen. Det er hastighetene på vinsjene som bestemmer farten til løpekatten. Løpekatten er komponenten som forflytter seg langs returlinjen. Treet som skal forflyttes er festet i en heisline ned fra løpekatten. Operatøren har mulighet til å heise treet ned og opp fra basestasjonen.

Arbeidsområdet til tradisjonelle taubanesystem defineres ut ifra to punkt. Det første punktet blir definert av basestasjonen. Det andre treet blir definert som et endetre. Returlinjen går mellom basestasjonen og endetreet. Det er returlinen som avgrenser bevegelses området til løpekatten.

2.1.3 Styring og kontroll

Operatøren styrer taubanen fra basestasjonen. Operatøren heiser løpekatten og linene ved å kjøre vinsjen til returlinen og trekklinen parallelt. Ettersom trekklinen og returlinen beveger seg i hver sin retting heiser kreftene på systemet løpekatten opp. På det nåværende taubanesystemet er operatøren avhengig av å styre løpekatten fra basestasjonen. Dette gir operatøren et lite overblikk over arbeidslinjen som er på det jevne 300m [3].

2.1.4 Fremtidig system

En To Tre Teknikk ønsker å frigjøre operatøren fra basestasjonen på et 3 -punkts system. For å frigjøre operatøren fra basestasjonen skal en liten håndkontroller styre 3-punktsystemet. Operatøren har dermed mulighet til å bevege seg nærmere forflytnings område til treet. Dette gir operatøren muligheten til en kontrollert forflytning av trærne.

2.2 Kabeldrevet robot - tre punkt system

I dette delkapitelet beskrives En To Tre Teknikks 3 punksystem. Vi definerer 3-punkts systemet som en kabeldrevet robot. Videre i kapitelet beskrives funksjonaliteten og arbeidsområdet til den kabeldrevne roboten. I kapitelet beskrives også opp riggingen til En To Tre Teknikks kabeldrevne roboten med mulige fordeler. Til slutt gir vi en kort beskrivelse av styringen til en kabeldrevet robot.

2.2.1 Trepunkts system

En To Tre Teknikk ønsker å utvikle et tre punkt system til forflytning av tømmer i bratt terreng. Et av punktene defineres som basestasjon. De to andre punktene defineres som satellitter. Hvert av de tre punktene har sin egen vinsj. Vinsjen skal ha muligheten til å kjøre individuelt. Dermed kan lengden på returlinene ut til roboten forandres individuelt.

2.2.2 Opprigging

En operatør må bardunere basestasjonen og satellittene. Barduneringen vil dermed redusere de påførte kreftene til den kabeldrevne roboten under drift. For dypere beskrivelser av bardunerings teknikker henviser vi til tidligere bacheloroppgaven SatElite Skyline skrevet av Høyskolen på Vestlandet avd. maritime og almen maskin. Etter barduneringen skal operatøren trekke ut tre returliner til roboten. Returlinene skal trekkes fra basestasjonen og satellittene. Returlinene blir trukket fra hver sin vinsj og over sin egen trinse. Videre festes de tre returlinene til roboten. Trinsene er fastmontert til hvert sitt endetre tilknyttet enten en basestasjon eller en av satellittene. Det er trinsens oppgave er å guide returlines bevegelse mellom vinsjen og roboten. Trinsene som skal benyttes anbefales å være svingbare. Svingbare trinser kommer til å skape en friere bevegelse til returlinen. Ved en friere bevegelse på returlinene får roboten et større arbeidsområde innenfor den trekantede prismen.

Oppsettet til en kabeldrevet robot er tidkrevende i bratt terreng. Dette kommer frem i det siste avsnittet, men en kaldrevet robot kan muligens redusere riggkostnadene sett i forhold til et tradisjonelt taubanesystem. En kabeldrevet robot trenger kun en

opprigging. Etter første opprigging har den kabeldrevne roboten muligheten til å forflytte tømmer i et trekantet prisme. Dette har ikke et tradisjonelt taubanesystem muligheten til. Mulig reduksjonen av riggkostnadene er sett i forhold til forflytningsområdet til tømmeret.

2.2.3 Arbeidsområdet

Det er tre punkt som definerer arbeidsområdet til en kabeldrevet robot i terrenget. Avstanden mellom punktene kan være opp til 300m i luftlinje. Dette kom frem i mail samtale mellom En To Tre Teknikk og Norges Tekniske-Naturvitenskaplige Universitet (NTNU).

2.2.4 Basestasjon

Basestasjonens hovedoppgave er å kommunisere med satellittene for å posisjonere roboten. På en kabeldrevet robot blir løpekatten definert som roboten. Vi kommer dermed til å benytte robot istedenfor løpekatt videre i rapporten.

2.2.5 Håndkontroller

Det er ønskelig å benytte en enkel håndkontroller med skjerm som styringsenhet til den kabeldrevne roboten. Styringen av roboten utføres ved å bevege joysticken. De ulike bevegelsene til joysticken forflytter roboten i planet.

2.2.6 Robotens forflytning

Returlinene påføres ulike strekk krefter i forhold til hastighetene på vinsjene. Det er de ulike strekk kreftene som forflytter roboten til en ny posisjon i planet. Knappene på håndkontrolleren skal benyttes til heving og senkning av roboten. Skjermen sin hovedoppgave er å vise posisjonen til roboten. Operatøren skal også ha mulighet til å angi ønsket posisjon til roboten ved hjelp av skjermen. Ettersom En To Tre Teknikk skal benytte trådløskontroller frigjøres operatøren fra basestasjonen. Operatøren har da muligheten til å bevege seg nærmere roboten. Dette gir et større overblikk og større sikkerhet under forflytningen til treet.

3 Design/modellering

3.1 Fusion 360

Det ble valgt et modelleringsmiljø fra Autodesk kalt Fusion360 da vi tidligere har fått dette anbefalt av noen medstudenter. I tillegg er verktøyet gratis for studenter, som bygger opp under våre budsjettbegrensninger.

Fusion 360 kan også eksportere filer til programmer for 3D printere. Dette har vært en uvurderlig egenskap da et stort antall deler for prosjektet er blitt 3D printet.

3.2 Modellering

Bestilling fra oppdragsgiver var at de ønsket en prototype. Vår angrepsvinkel var å gå fra design til produkt. På denne måten ble konstruksjonsprosessen effektivisert. Samtidig har dette bidratt til å unngå uønskede problemstillinger i konstruksjonsfasen.

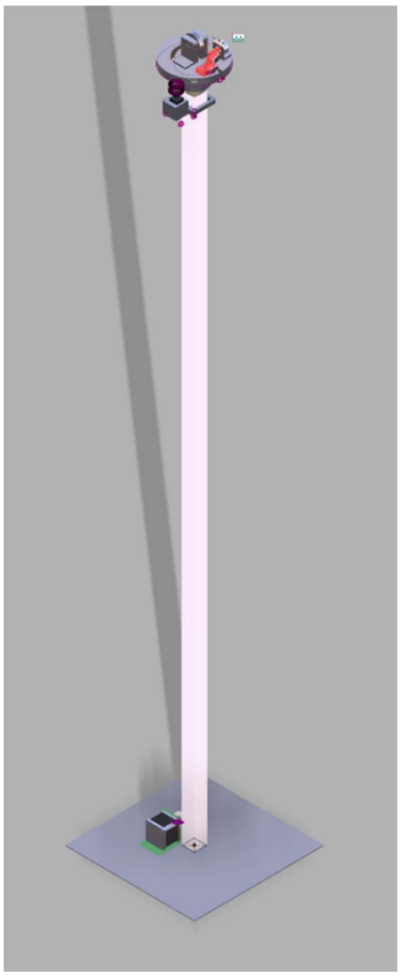
Vi begynte med et grovt utgangspunkt der det ble definert hvilke mekaniske egenskaper det var ønskelig at prototypen skulle ha. Det ble fastslått at tårnene skulle være 1,5m høye for å gi roboten nok arbeidsrom. For å kunne måle robotens vinkel i planet, ble det bestemt at trinsen måtte kunne rotere.

Modellerings metoden som er blitt benyttet kalles direkte modellering [4]. Dette gir oss den fordelen at vi kan gjøre endringer direkte på en modell. Dette muliggjør at modellens størrelse og endringer kan defineres i millimeter eller mindre. Den digitale modell ble designet med nøyaktige proporsjoner i forhold til den ferdige modellen. Den ferdige modellen er blitt designet med hensyn på styringen til prototypen.
Deler som ble 3D printet er følgende:

- Trommel til linene
- Holdere og braketter til potmeter
- Topplokk til tårnene
- Brakett til ultralyd sensor
- Trinsehjul

- Vinkler til trinsen
- Roboten.

Modellene vi har designet og 3D printet er vedlagt i vedlegg 26-34. Designet av komponentene kommer fram under delkapitlet 3D printede deler. Dersom 3D printing ikke hadde vært en mulighet, hadde delene måtte blitt håndlaget i tre eller metall. Dette ville medført stort tidsforbruk. 3D skriveren arbeider selvstendig, og her ligger det en stor besparelse i tid.



Figur 3 bilde av digitalt sammensatt tårn

3.3 Komponenter

3.3.1 3D printede deler

For å gi tårnene ønsket funksjonalitet var det behov for å lage mekaniske komponenter som er spesielt designet til vårt bruk. Vi benyttet skolens 3D printer, en Zotrax M200 for å lage disse komponentene. Det ble benyttet et filament av typen ABS for printing av komponentene. ABS er et av de vanligste typene filament og har god mekanisk styrke sammenlignet med for eksempel PLA [5]. Under prosjektet ble det oppdaget at dette filamentet deformere seg litt etter printing. Dette har blitt kompensert for i vårt design.

3.3.1.1 Roboten

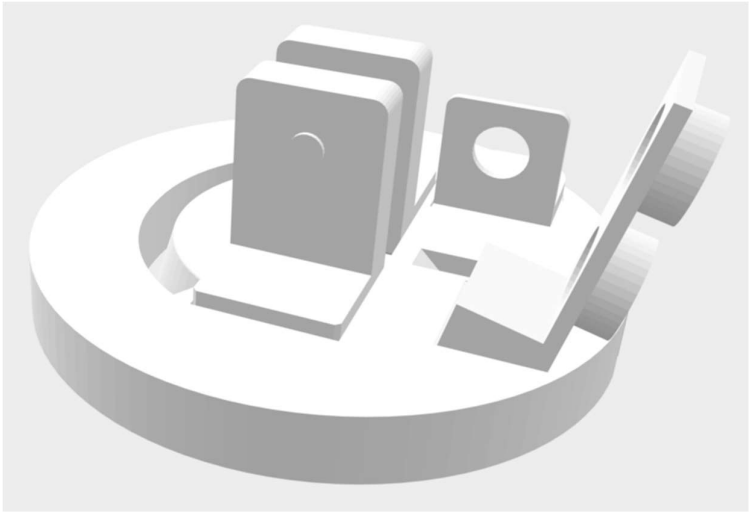
Roboten består av en tricopter ramme som er designet for droner. Denne er den eneste 3D printede delen i vårt prosjekt som vi ikke har designet selv. Det ble bestemt å anskaffe robotmodellen fra en ekstern kilde. Dette ble utført fordi designet ville ta for lang tid utføre.

Tricopter rammen består av fem sammen limte komponenter. En topp plate, en bunnplate og tre «armer». Grunnen til at vi valgte denne modellen var at den har gode muligheter for å feste komponenter, og fordi de tre armene er ideelle å feste til tråden fra tårnene.

Figur 4 Printet og sammensatt tricopter ramme

3.3.1.2 Topplokk

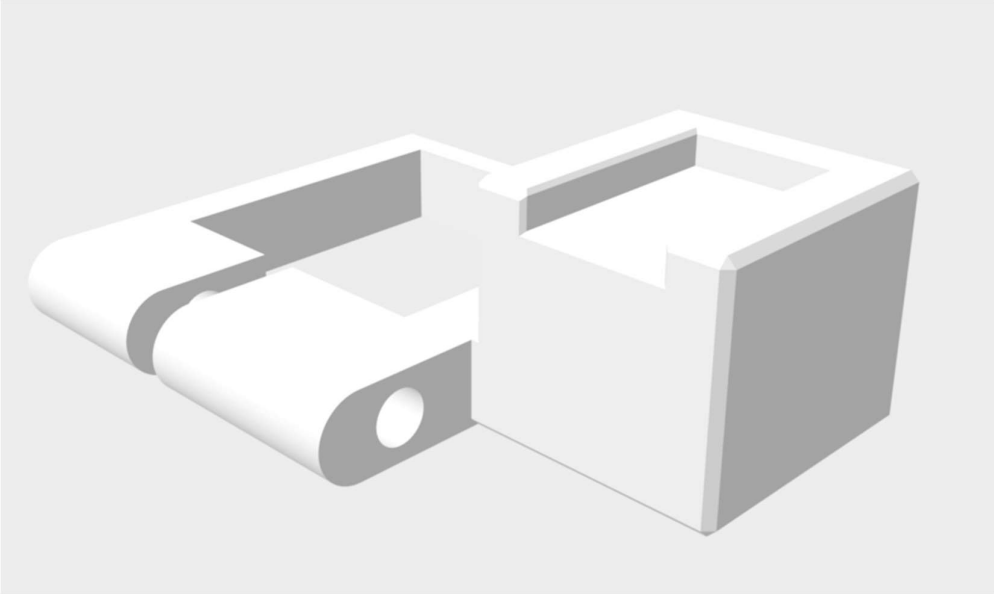
For å kunne feste trinsen samt sensorene designet vi en plattform som kan monteres på kronen til det roterende hode på tårnet. Dette topplokket har spor for å lede tråden og spor til plassering av brakettene som holder ultralydsensor, potensiometer og trinsen. Tegning for topplokk og braketter er vedlagt i vedlegg nr.2 .



Figur 5 Topplokk med braketter for sensorer og trinse

3.3.1.3 Potensiometer holder

For at potensiometeret skal kunne stå inntil kronen til det roterende hodet har vi designet en holder som kan festes på firkant røret. Denne holderen kan festes med en skrue og mutter. Holderen er designet en del større enn firkant stålet, da vi ønsket å plassere gummi mellom holderen og røret for å skape fjæring mellom firkantstål og holder.



Figur 6 Holder for potensiometer

For å feste et potensiometer til topplokket har vi designet en enkel brakett. Vi har valgt å benytte den medfølgende festemutteren til å feste potmeteret til braketten. Tegning for brakett og potensiometerholder er vedlagt i vedlegg nr.7 og 8.

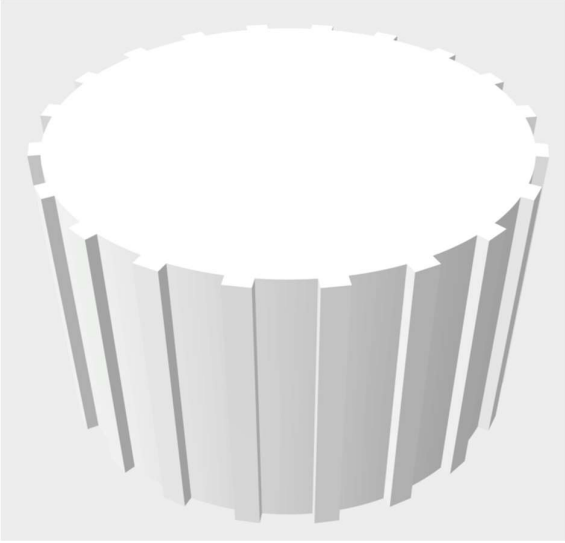
3.3.1.4 Arm til potensiometer

For å kunne kalkulere robotens høyde i rommet designet vi en arm, som kan plassere på våre potensiometere. Armen har et hull i enden for gjennomføring av tråden. Armen vil derfor følge tråden ut til roboten og vi kan bruke et potensiometer for å måle robotens vinkel i forhold til tårnet. Tegning i vedlegg nr. 9.

Figur 7 Arm til potensiometer for å måle vinkel i Z akse

3.3.1.5 Hjul til potensiometer

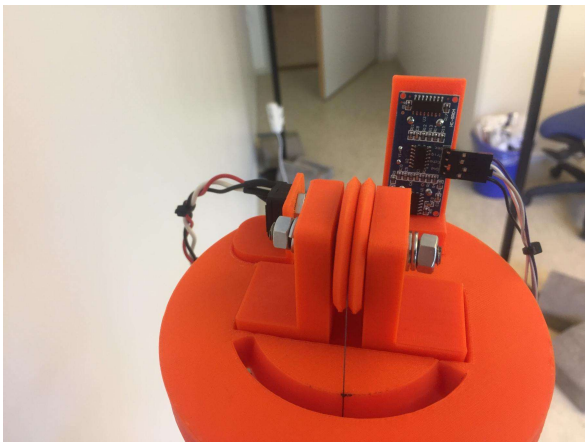
For å kunne kalkulere robotens posisjon i xy-planet ble det designet et hjul som kan monteres på potensiometrene. Dette hjulet er en sylinder med «tenner» for å gi bedre overflate mot det roterende hodet. Tegning i vedlegg nr. 1.



Figur 8 Hjul til potensiometer for å måle vinkel i planet

3.3.1.6 Trinse

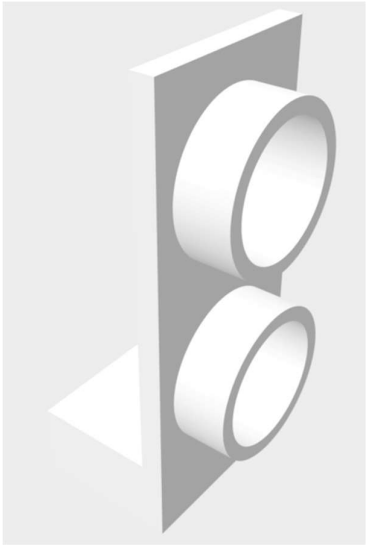
Trinsens design består av to vinkler, et hjul, en akse og et lager. Hjul og vinkler er 3D printet. Akslingen består av en gjengestang som er festet til vinklene med muttere. Hjulet har et spor for å holde linen på plass. Tegning i vedlegg nr.3 og 6.



Figur 9 Montert trinse på topplokk

3.3.1.7 Brakett til ultralydsensor

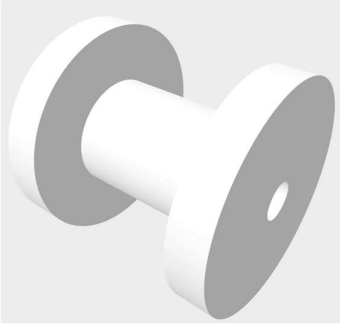
For å feste ultralydsensoren til topplokket er det blitt designet en brakett. Denne har støtte for sender og mottaker på sensoren. Bunnen av braketten er hevet da sensoren skal være vinklet nedover. Dette designet gjør at ultralydsensoren kan festes i braketten uten behov for lim. Braketten er limt fast i topplokket. Tegning i vedlegg nr. 5.



Figur 10 Brakett til ultralydsensor

3.3.1.8 Trommel

For å kunne spole inn linen med steppermotor er det designet egen trommel. Trommelen er 30mm i lengden og har en diameter på 15mm der linen ligger. Den ytre diameteren er på 30mm. Hullet for stepper motoren er på 5mm med et spor på den ene enden tilpasset akslingen på steppermotoren. Tegning i vedlegg nr. 4.



Figur 11 Trommel modell

3.3.2 Mekaniske komponenter

3.3.2.1 Tårn

Tårnet består av et enkelt firkantrør festet i en bunnplate. Begge disse er laget i metall da de må tåle litt krefter. Firkantrøret kan separeres fra bunnplaten ved å løsne en skrue nederst på firkantrøret. Både røret og bunnplaten ble laget av Øyrane videregående skole.

Figur 12 Tårn med bunnplate

3.3.2.2 Roterende hode

For at trinsen og sensorene skulle kunne følge roboten, må hode til tårnet kunne rotere. For å løse denne utfordringen fikk vi god assistanse fra Øyrane Videregående skole. Det roterende hode består av fire deler: Et feste til firkantrøret som er den sylindriske delen synlig nederst på figur 13, en klemhylse som står mellom lageret og festet til firkantrøret, et kulelager og en krone som det har blitt påmontert trinse og sensorer på. Tekniske tegninger for det roterende hode ligger som vedlegg nr. 35-39.

Figur 13 Roterende hode til tårn

3.3.2.3 Brakett til steppermotorene

Braketten til steppermotoren ble utviklet av mekanisk avdeling på Øyrane VGS. Her ble det benyttet en CNC fres. På CNC fresen ble det benyttet G-kode for automatisk fremstilling av steppermotor braketten. G-koden kan finnes i vedlegg nr. 25.

Figur 14 Steppermotor montert på brakett

4 Arbeidsstasjon

Prototypen er tilknyttet en arbeidsstasjon. På arbeidsstasjonen skal en operatør ha mulighet til å styre den kabeldrevne roboten. Arbeidsstasjonen består av en hovedstyringsenhet med to skjermer.

4.1 Hovedstyringsenhet DELL Precision Tower 3420

Det blitt benyttet en stasjonær datamaskin til å representere en eventuell industriell datamaskin. Den stasjonære datamaskinen kommer fra produsent Dell. Prosjektgruppen fikk lånt den stasjonære datamaskinen fra HVL til prototypen. Datablad for denne er vedlagt i vedlegg nr. 23.

Operativsystemet som fulgte med hoved styringsenheten var Windows 10 Entprise fra Microsoft. Det ble valgt å ikke bytte operativsystem. Hovedgrunnen til dette var at flere programvarer som prototypen benytter, støtter kun Windows.

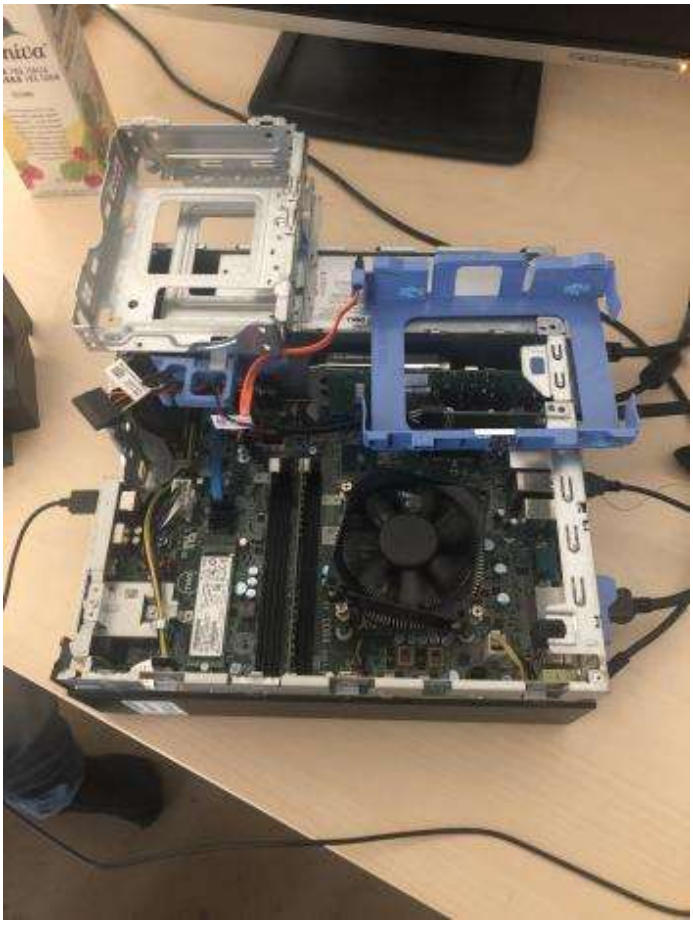
Hoved styringsenheten har en Intel Core i7 prosessor som består av 4 logiske prosessorer og hver logisk prosessor har 2 kjerner. Dette gir til sammen 8 logisk kjerner som er fordelt mellom Windows og TwinCAT.

Hoved styringsenheten har også en RJ45 Ethernet port. Prosjektgruppen har installert EtherCAT driver på porten for utgående kommunikasjon til EK1100 modulen.

Figur 15 Arbeidsstasjon hovedstyringsenhet

4.2 Oppgradering av hovedstyringsenhet

Det har blitt gjort diverse tiltak for å gjøre arbeid på hovedstyringsenheten så effektiv som mulig. Det ble blant annet installert Teamviewer for å gi prosjektgruppens medlemmer tilgang til hoved styringsenheten hvor og når som helst. Et annet tiltak var å låne en ekstra skjerm av skolen for å øke produktiviteten. Det eneste utfordringen var datamaskinens støtte for flere skjermer. Den var skrudd av i maskinens basic input outputs system (BIOS). BIOS tilgangen var også passord beskyttet. For å få vekk passordbeskyttelsen må det kobles fra en lask internt på hovedkortet. Denne koblingen fins i de fleste datamaskiner. Når passordet til BIOS hadde blitt fjernet, kunne støtten for flere skjermer slåes på. Driveren til skjermkortet ble også oppdatert. Dette løste utfordringen.



Figur 16 Oppgradering av hovedstyringsenhet

5 Prosjekteringsplattform – TwinCAT 3

En To Tre Teknikk må benytte seg av en kraftig prosjekteringsplattform på hovedstyringsenheten. Prosjekteringsplattformen som skal benyttes må være robust og driftssikker. Det finnes flere anerkjente leverandører som kan benyttes. Noen eksempler på leverandører er Beckhoff Automation, Simens og Rockwell Automation.

5.1 TwinCAT 3

Det ble valgt å benytte produsenten Beckhoff Automation som leverandør av prosjekteringsplattform. De har spesialisert seg på PC baserte sanntids kontrollere innen software PLS, numerisk kontroller, og robotikk. TwinCAT 3 er automatiserings programvare som benyttes for å utvikle PC baserte sanntids kontrollere [6].
Ved å benytte TwinCAT 3 ble ikke prototypen påført lisenskostnader under bacheloroppgaven, da selskapet tilbyr gratis 7 dagers lisenser. Disse lisensene kunne oppdateres gjennom hele prosjekt perioden [7]. En To Tre Teknikk har også muligheten til å benytte seg av TwinCAT 3 til en eventuell endelig kabeldrevet robot.

For å kunne benytte seg av TwinCAT 3 må det velges et utviklingsmiljø. Her var det mulighet til å velge mellom blant annet Visual Studio og Codesys. Vi har valgt å benytte Visual Studio som utviklingsmiljø. Visual Studio er et svært populært utviklingsmiljø blant programvareutviklere. I 2018 benyttet 39.8% seg av Visual Studio som standard utviklings miljø, der 1860 programvareutviklere deltok i spørreundersøkelsen [8]. Bachelorgruppen har også tidligere erfaring med å utvikle applikasjoner i Visual Studio. Dermed ble det naturlig å benytte dette som utviklingsmiljøet til TwinCAT 3.

TwinCAT 3 ble tildelt fire isolerte kjerner under konfigurasjon av sanntids prosessering til styringen. De fire isolerte kjernene kjøres i Windows Kernel Mode. Her kjøres programvare uavhengig av hverandre [9]. Dermed har ikke annen programvare mulighet til å avbryte operasjoner som utføres av TwinCAT 3.

5.1.1 Software PLS

På prototypen har det blitt benyttet software PLSen til TwinCAT 3. Software PLSen er en virtuell enhet som benyttes til programbar logisk styring. Software PLSen ble benyttet til å utvikle en styring til prototypen. En To Tre Teknikk har også muligheten til å benytte software PLSen på en fullkommen kabeldrevet robot.

Da operativsystemet består av flere kjerner er det blitt valgt å tildele en kjerne til hovedprogrammet. Syklustiden for opphenting av hovedprogrammet er blitt satt til 1ms. Dette er blitt utført for å sikre en rask behandling av dataen som benyttes til styringen av roboten. Det er ikke nødvendig med høyere syklustid da 1ms er nok for å kjøre en iterasjon av programmet.

5.1.2 Numerisk kontroller

På TwinCAT 3 har det blitt valgt å benytte den numeriske kontrolleren. Den numeriske kontrolleren kobles opp mot EL7031 for styring av steppermotorene. Det er også mulig å benytte en numerisk kontroller til motorstyringen på en fullskalert kabeldrevet robot.

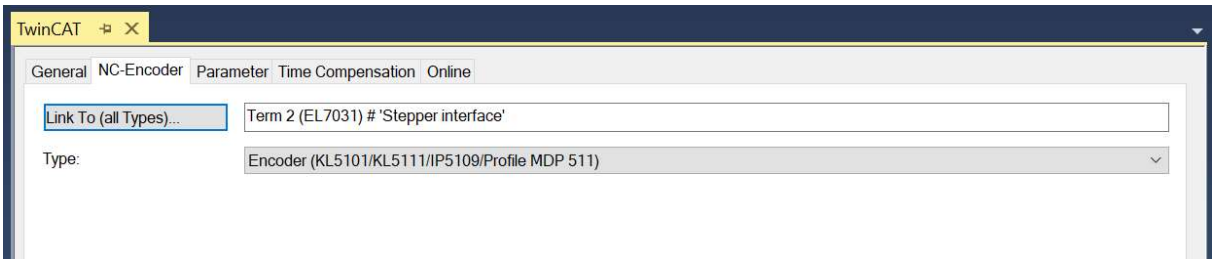
Under sanntidskonfigurasjonen til TwinCAT 3 har den den numeriske kontrolleren blitt tildelt en egen kjerne. Dermed kjøres den numeriske kontrolleren uavhengig av andre Windows applikasjoner. Syklustiden til den numeriske kontrolleren er satt til 2ms. Dette er standard syklustid som kommer ved opprettelse av en numerisk kontroller. Det ble ikke valgte å endre denne syklustiden da denne frekvensen er tilfredsstillende for å utføre en iterasjon av den numeriske kontrollerens oppgave.

På hver akse har det blitt valgt å benytte en kontinuerlig akse på hver steppermotor driver. Den kontinuerlige aksen ble valgt, ettersom dette er den eneste akse typen EL7031 støtter. Dermed kunne ikke steppermotor aksen benyttes, noe som var ønskelig. Dette var ønskelig ettersom stepper aksen kun teller antall stepp og ikke benytter posisjonering. Ved videreutvikling av prototypen bør det vurderes å anskaffe driverkort spesifisert i vedlegg nr. 13.

Den kontinuerlige aksen blir automatisk satt opp med steppermotordriveren EL7031, en simulert enkoder og en PID regulator. Hver akse er blitt knyttet opp til en global tabell som software PLSen bruker for å overføre av data. På den numeriske kontrolleren er det et integrert grafisk grensnitt. Prosjektgruppen benyttet det grafiske grensnittet til å styre steppermotoren under kalibreringen.

5.1.2.1 Enkoder

Det ble benyttet først en virtuelt simulerte enkoder til å registre antall stepp. Dette viste seg å være feil fremgangsmåte etter å ha vært i kontakt med Beckhoff. Vi ble informert av Beckhoff at alle enkodere blir simulert, dersom ikke en fysisk enkoder er tilkoblet. Dermed rekonfigurerte vi enkoderinstillingene.



Figur 17 Oppsett bilde av konfigurasjon til aksenes simulerte enkoder

Kalibrering av encoderen forekom i to forsøk. I det første forsøke benyttet vi EL7031 sin manual til å stille inn skaleringsfaktoren på encoderen. Da ble følgende formel benyttet

$$SF = \frac{Trommel\ omkrets}{Fullestepp \cdot antall\ mikrostepp} = \frac{2 \cdot r_{trommel} \cdot \pi}{200 \cdot 64} = \frac{47{,}1}{12800}$$ [10].

Skaleringsfaktoren til encoderen kan skrives inn som teller og nevner. Dette ble benyttet ettersom utregningen gir et irrasjonelt tall.

For å kalibrere encoderen ble det benyttet en fast lengde på 250mm. Etter flere forsøk forekom det et positivt avvik på 40mm. Det ble benyttet to nye lengder på 500mm og 1000mm for å undersøke om avviket var lineært. Dette avviket viste seg å være ulineært. Dermed kunne det ikke benyttes en prosentfaktor i programmet for å rette opp avviket.

En annen kalibreringsmetode som ble benyttet, var en manuell metode for kalibrering av steppermotorene. Her ble følgende formel benyttet.

$$SF_{ny} = SF_{gammel} \cdot \frac{Målt\ lengde}{Indikert\ lengde}$$

Den manuelle metoden går ut på å sette skaleringsfaktoren til en ubestemtverdi. Deretter kjøres steppermotoren en fast lengde som deles på den indikerte lengden. Dette gir en prosentvisfaktor som forteller om den satte skaleringsfaktoren er for høy eller lav. Den prosentvisefaktoren multipliseres med den satte skaleringsfaktoren og gir en ny skaleringsfaktor. Denne skaleringsfaktoren skrives inn og kalibreringen utføres til ønsket presisjon er oppnådd. Her ble det også benyttet en fast test lengde på 250mm. Resultatet av kalibreringen kommer frem i tabell 1.

Tabell 1 Manuel kalibrering av skalerings faktor

SF_{gammle}	Indikert lengde i mm	Målt lengde i mm	SF_{ny}
0.001	249.986	985	0.00394
0.00394	250.013	252	0.003971
0.003971	250.0107	250	

Det ble valgt å gjennomføre en etterkontroll av den manuelle kalibreringen med en lengde på 500mm og 1000mm. Det viste seg at steppermotoren hadde avvik i forhold til ønsket verdi. Det viste seg at avviket var ulineært. Dermed var det ikke mulig å benytte en prosentvis faktor for å rette opp avviket. Dette resulterte i en unøyaktig posisjonering av roboten.

5.1.2.2 Steppermotor driveren EL7031

Steppermotor driveren ble konfigurert med referansehastigheten, maksimalt tillatte hastighet og overvåkning. Referansehastigheten på driveren forteller den maksimale hastigheten steppermotoren kan kjøre på uten å miste stepp. Dersom steppermotoren mister stepp blir posisjoneringen til roboten unøyaktig. For å stille inn referanse hastigheten benyttetes manualen til EL7031 etter gitte formler

$V_{max} = \frac{Grunn\ frekvens}{Motor\ frekvens} = \frac{2000\ full\ steps/s}{200\ full\ steps/omdreining} = 10\ omdreninger\ /s$ [10].

$V_{ref} = 10\ omdreinger/s \cdot trommel\ omkrets = 10\ omdreinger/\ s \cdot r \cdot \pi \approx 471mm/s$

I formelen benyttes en grunnfrekvens på 2000 full steps/s. Dette er standard grunnfrekvensen til EL7031. Det ble valgte å ikke endre grunnfrekvensen ettersom roboten aldri vil komme til å bli kjørt raskere enn 471mm/s.

For å unngå varmgang på steppermotoren ble også viklingsresistansen og strømbegrensning satt. Viklingsresistansen ble satt til 4.1Ω og maksimale strømmtrekket ble satt til 1A. Verdiene til viklingsresistansen og det maksimale strømmtrekket er hentet fra databladet til steppermotoren se vedlegg nr.14.

På steppermotor driveren er det også mulighet for å stille inn rampingen til steppermotoren. Rampingen forteller om steppermotoren skal ha en hard eller myk stop. Ut ifra grafen nedenfor har vi en stiv ramping. Dette er ønskelig for å få en hurtig respons på prototypen.

Figur 18 Bilde av ramping til steppermotoren til prototypen

5.1.2.3 PID-regulatoren

PID-regulatoren på aksene kan benyttes til å regulere ønsket posisjon. Det ble valgt å benytte en ren P-regulator. Proporsjonalbåndet viste ingen synlig innvirkning på posisjoneringen til steppermotoren. Dette kom frem under benyttelsen sanntidstrenden til TwinCAT 3. Bachelorgruppen mener at regulatoren ikke hadde innvirkning på posisjoneringen ettersom en simulert enkoder ble benyttet.

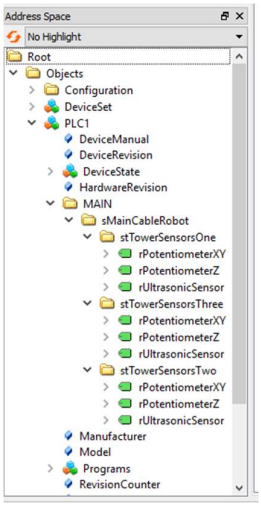
5.1.3 OPC UA server

En To Tre Teknikk må benytte seg av en server for innsamling av sentral data. På den fullverdige kabeldrevne roboten til En To Tre Teknikk anbefaler prosjektgruppen å benytte en OPC UA server. Det anbefales å benytte en OPC UA server ettersom den er enkel å konfigurere. Den kan også integreres opp mot alle klienter som støtter OPC UA [11].

På prototypen benyttes OPC UA serveren til TwinCAT 3 for innsamling av data fra smartsensoren. OPC UA serveren er konfigurert opp uten kryptering. Det har blitt valgt å ikke bli benyttet kryptering ettersom sensordataen som overføres ikke er konfidensiell.

OPC UA serveren bruker node-trær til organisering av data. Det har blitt valgt å bruke tre underliggende noder til hvert tårn for organisering av smartsensordataen. Hver tårnnode er organisert under software PLSens node.

Det er blitt opprettet en variabel for hver sensor på smartsensoren. To av variablene benyttes til innhenting av potensiometer data. Den siste variabelen benyttes til innhenting av data fra ultralydsensoren. Alle variablene er gitt lese og skrive rettigheter dermed kan OPC UA klienter hente data, samt skrive data til variablene.



Figur 19 Node-tre OPC UA

6 SCADA applikasjonen

For å lage et brukervennlig produkt ble det bestemte å utvikle en SCADA applikasjon. Der en eventuell operatør kan styre og overvåke anlegget. SCADA applikasjonen som er blitt utviklet, kan videreutvikles og overføres til et større system.

6.1 Krav

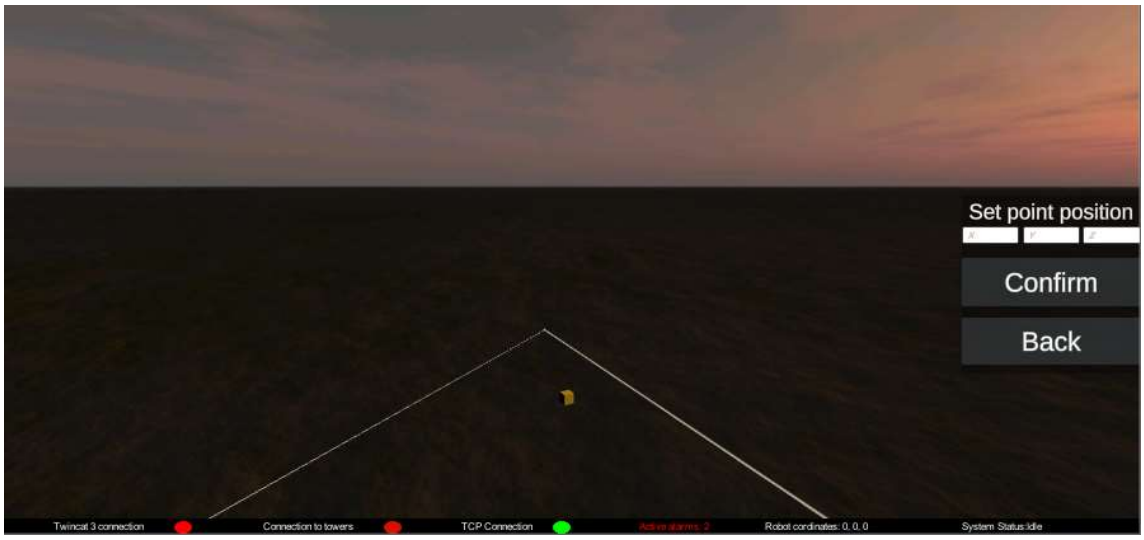
SCADA står for «Supervisory Controll And Data Acquisition», som er et system som kan samle data, overvåke og styre systemet [12]. På SCADA systemet har det blitt satt følgende krav:

- Alarmliste som viser aktuelle alarmer
- Mulighet for å velge mellom flere styringsformer
- Visualisering i 3D over hvor roboten befinner seg i arbeidsområdet
- Mulighet til å vise historiske data
- Systemet skal være tilgjengelig for alle PCer og enheter på det tekniske nettverket

6.2 Spesifikasjoner

For å få 3D visualisering i SCADA applikasjonen kan det enten brukes en C# applikasjon med et tredjeparts Matlab skript eller bruke en grafikkmotor til å lage en egen applikasjon.

Matlab blir brukt til industrielle applikasjoner og forskningsinstitutter, og er veldig bra til simulering og beregninger. 3D visualiseringen vil i så fall bestå av en 3D-graf. Det kan bli lettere å utvikle eventuelle grafer og lagring av historisk data.
3D grafikkmotorer blir for det meste brukt til dataspill design. 3D grafikkmotoren som er aktuell for dette prosjektet heter Unity. Unity er en plattform som brukes mest til å lage spill i 3D. Det er mulig å kjøre en applikasjon som er lagd i Unity på de fleste operativsystemer, til og med på operativsystemer på mobiler og nettbrett [13]. Dette kan gi stor fleksibilitet for en eventuell operatør. Det var også en stor fordel at skriptspråket Unity bruker er C# Assembly som er en nedskalert versjon av C# som er brukt i denne oppgaven.



Figur 20 Visualisering i Unity annen kameravinkel 1

Figur 21 Visualisering i Unity kameravinkel 2

Dette vil også skape en stor fleksibilitet når det gjelder operativsystem. Ikke minst gir dette store muligheter til å videreutvikle systemet.

6.3 Utviklingsmiljø - Unity

En Unity applikasjon består av mange scener. En scene er et 3 dimensjonalt miljø, der det er mulig å legge til objekter. Disse objektene inneholder attributter. Eksempler på slike kan være en fysisk form eller et skript som blir kjørt ved oppstart av scenen eller ved bildeskift. Det er også mulig å legge til skript som er avhengig av en

hendelse som for eksempel et tastetrykk. Dette gjør Unity til en god plattform for å visualisere systemet vårt.

Figur 22 Unity logo [45]

6.4 Kommunikasjon

SCADA applikasjonen trenger pålitelig kommunikasjon med TwinCAT 3. Det har blitt benyttet OPC UA til kommunikasjonsprotokollen mellom TwinCAT 3 og SCADA applikasjonen. Unity har et OPC UA bibliotek, men dette er ikke lengre støttet av produsenten. Derfor har OPC UA klienten som blir brukt til å motta sensor informasjon til TwinCAT 3 blitt modifisert. Ved Unity applikasjonens oppstart blir Python skriptet startet automatisk. Dersom applikasjonen skal kjøres på mobil eller nettbrett må OPC UA klienten tilpasses og kjøres på hoved styringsenheten.

6.5 Python skriptet

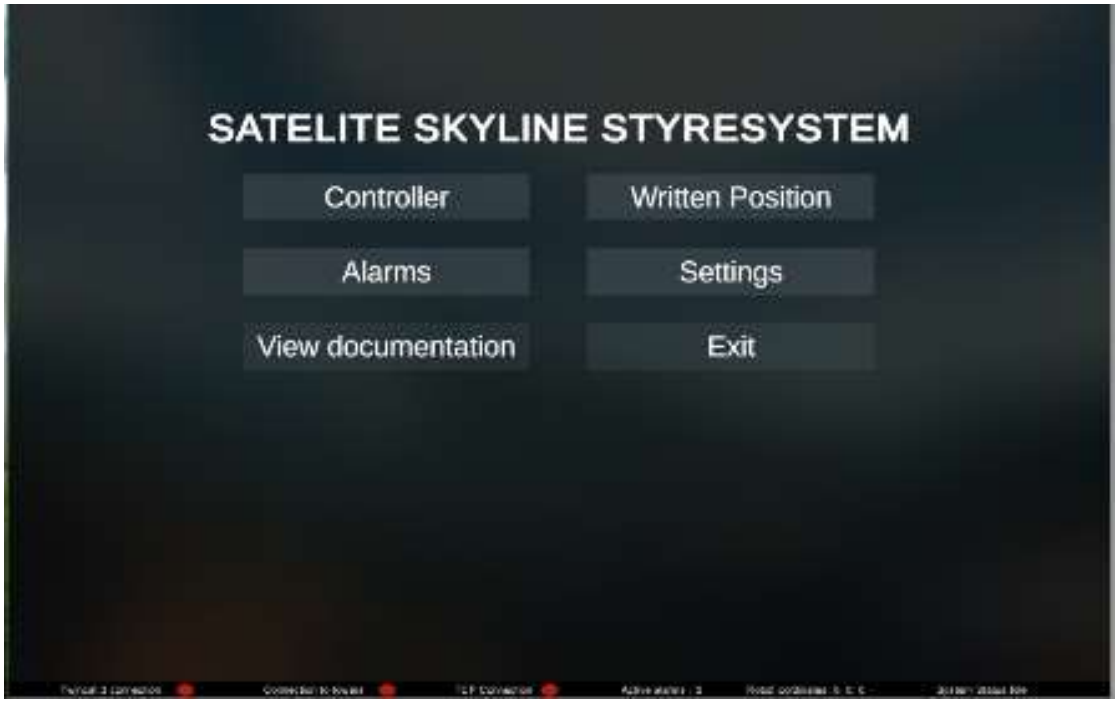
Python skriptet inneholder en OPC UA klient og en TCP-server. Ved oppstart opprettes OPC UA klienten. Klienten abonnerer automatisk på alle variabler som ligger i en spesiell mappe på OPC UA serveren. Når OPC UA klienten mottar verdier fra OPC UA serveren blir disse lagret i en lang tekststreng. TCP serveren mottar deretter et signal og svarer med en tekststreng som inneholder de siste verdiene til prototypen.

6.6 Scener

Det ble valgt å opprette 4 scener i SCADA applikasjonen. En hovedmeny, alarmliste, innstilinger og et 3D-miljø til å visualisere robotens posisjon i arbeidsområdet. Alle scenene inneholder også en statusbar. Statusbaren gir operatøren nåværende system status. Her vises det blant annet roboten sin nåværende posisjon, antall aktive alarmer og informasjon om nåværende system status.

6.6.1 Hovedmeny scenen

Hovedmeny scenen er applikasjonens hovedmeny der operatøren kan velge hvilken funksjon som er ønskelig å benytte. Hovedmeny scenen inneholder også en dokumentasjonsknapp som kan åpne dokumentasjon tilknyttet prototypen. Denne kan senere enkelt byttes ut med en fullverdig brukerdokumentasjon eller hjelpe funksjon. Her kan også operatøren velge mellom styremodus. Ved valg av styremodus vil applikasjonen gå inn i det simulerte 3D miljøet.



Figur 23 Hovedmeny SCADA applikasjonen

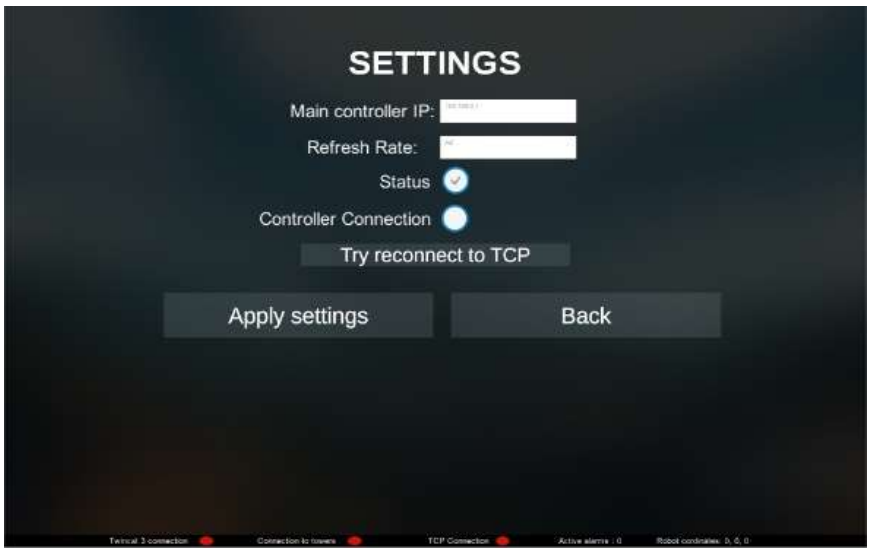
6.6.2 Alarm scenen

Alarm scenen består av en alarmliste der aktiverte alarmer blinker. Alarmsystemet består også av fargekoding. Der kritiske alarmer har fått fargekoden rød mens ikke-kritiske alarmer har fått fargekoden gul. I tillegg inneholder listen tidsstempler som viser når alarmen ble aktivert og kvalitet på OPC UA noden. Det har blitt utviklet to alarmer, OPC UA Connection og TCP Connection for å vise funksjonalitet til alarmlisten. Resterende alarmer har ikke blitt integrert på OPC UA serveren, grunnet fokus på utvikling av TwinCAT programvare. Dette kan gjøres ved en eventuell videreutvikling.

Figur 24 Alarm liste

6.6.3 Settings scene

Settings inneholder innstillinger for applikasjonen. Her kan operatøren å koble seg opp på ulike OPC UA servere. Operatøren har også muligheten til å endre abonnerings tiden på nodene og variabler som ligger på OPC UA serveren. På denne scenen er det også mulig å kjøre omstart av Python skriptet. Ved videreutvikling bør denne scenen oppdateres. Skriptene til denne scenen er ikke utført, da arbeid på steppermotoren tok lengre tid enn antatt.



Figur 25 Settings scenen

6.6.4 Robot scenen

Dette er Scenen som skal visualisere roboten. Denne siden viser for øyeblikket en trekant og et objekt som står i origo på koordinatsystemet. Dette er fordi styringsalgoritmen ikke er blitt ferdigstilt. Dersom SCADA systemet skal videreutvikles anbefales det å fokusere videre på denne delen. Den visuelle delen bør oppgraderes, det er mulig å benytte seg av de 3D designede modellene. Forflytning til robot er ikke lagd, fordi styring av robot ikke er ferdigstilt.

6.7 SCADA

SCADA applikasjonen som har blitt utviklet kan være et verdifullt verktøy for posisjonering av roboten. Denne kan gi muligheten til å posisjonere roboten ut ifra operatørens ønsket posisjon. Alarmene er også viktig for en eventuell operatør, for å forsikre seg om at systemet fungerer optimalt. Ved videreutvikling er det også mulighet til å lage en digital tvilling i Unity miljøet for å simulere drift. En SCADA applikasjon kan også være en essensiell del av en fullskalert kabeldrevet robot.

7 Styreskap

Rammen for valg av komponenter ble satt ut ifra ressursene som var delvis tilgjengelige på skolen. Dette omhandlet i særlig grad komponenter som skulle bli benyttet i styreskapet. Øvrige komponenter som er benyttet fremgår i regnskapet, se vedlegg 43.

7.1 Valg

I styreskapet er det montert flere elektriske komponenter. Disse komponentene har blitt valgt ut ifra økonomi, tilgjengelighet og integrasjonen mellom komponentene. Flere komponenter har blitt valgt ut ifra produsenten Beckhoff Automation. Grunnen til dette var at skolen hadde tilgjengelig utstyr fra denne produsenten, samt egne innkjøpsavtaler av komponenter. Dette ga reduserte kostnader på det elektriske utstyret som benyttes i styreskapet.

7.2 Tavlen

Til protypen er det blitt benyttet et styreskap fra Eldon som har IP 54. Der det første tallet angir beskyttelsen mot interegning av faste gjenstander og støvpartikler. Det andre tallet angir beskyttelsen mot inntrenging av vann. Her angir klassifikasjonen IP54 at styreskapet er støvtett og sikret for vannsprut [14]. Skapet beskytter også mot berøring av strømførende deler, som er i tråd med DSB sine forskrifter [15]. For å opprettholde IP graden er det blitt benyttet M20 plastnipler med IP grad 68 for selve kablegjennomføringen.

Figur 26 Styreskapet

7.3 Strømforsyning CP5 - 242

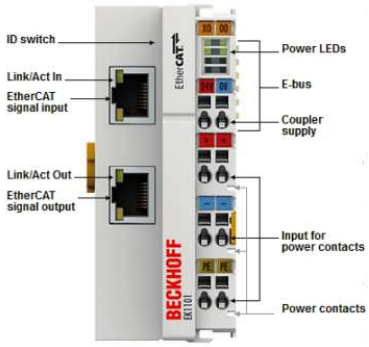
Prosjektgruppen benyttet en 24V DC strømforsyning, da alle andre komponenter i styreskapet trengte en 24V strømforsyning. På prototypen ble det benyttet strømforsyningen CP5-242 fra PULS Dimension. Tilførselspenning som benyttes på strømforsyningen er 230VAC. Det maksimale strømforsyningstrekket CP5-242 er på 5A. Datablad for CP5 – 242 er vedlagt i vedlegg nr. 22.

Figur 27 Spenningsforsyningen CP5-242 fra PULS Dimension

7.4 EtherCAT slavemodulen EK1100

På prototypen er EtherCAT slavemodulen EK1100 fra produsenten Beckhoff. EK1100 er blitt benyttet for å sammenkoble I/O enheter og den numeriske kontrolleren på hovedstyringsenheten.

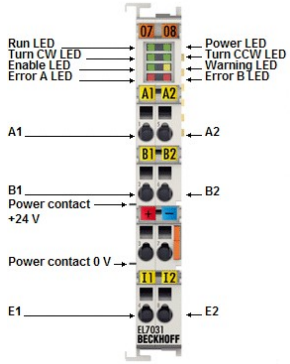
Funksjonaliteten til EK1100 er å omforme EtherCAT telegram fra hovedstyringsenheten. EtherCAT telegrammene omformers til E-bus signal. E-bus signalet går videre til steppermotor driveren som forsterkere E-bus signalet til et pulsmodulert bredde signal. Det er det pulsmodulerte bredde signalet som angir posisjonen til steppermotoren. Datablad for EK1100 er lagt ved i vedlegg nr. 12.



Figur 28 Beckhoff slavemodul EK1100

7.5 Steppermotordriveren - EL7031

EL7031 er kjøpt fra produsenten Beckhoff. På prototypen er EL7031 I/O grensesnittet mellom steppermotoren og software PLSen.
EL7031 omformer E-bus signalet fra slave modulen EK1100 til et pulsmodulert breddesignal. Det pulsmodulerte bredesignalet benyttes til å styre steppermotoren posisjon. EL7031 har ikke mulighet for oppkobling av enkoder. Dermed kan ikke den aktuelle posisjonen måles, men kun simuleres. Datablad for EL7031 i vedlegg 17



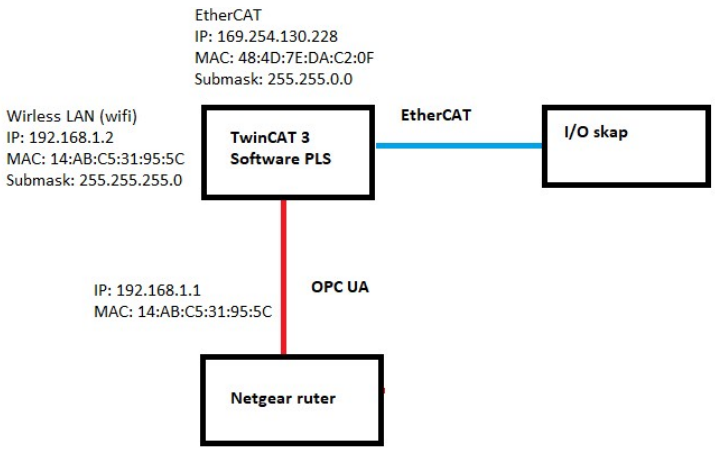
Figur 29 Beckhoff steppermotor driver EL7031

8 Kommunikasjon

8.1 Teknisk nettverk

Kommunikasjonen har blitt delt opp i to lag. Den første laget er kommunikasjon som må forgå i sanntid. Dette er kommunikasjonen som er essensiell for systemets drift. Laget består av kommunikasjonen mellom steppermotorene og hovedstyringsenheten. EtherCAT er blitt brukt som systemets sanntids kommunikasjonsprotokoll. Det ble blant annet anbefalt å bruke EtherCAT av Beckhoff, som er produsenten til komponentene som skal kommunisere i sanntid. Det andre laget er kommunikasjon som ikke er essensielt for drift av systemet. Dette er blant annet kommunikasjonen mellom hovedstyringsenheten, SCADA applikasjon og smartsensor.

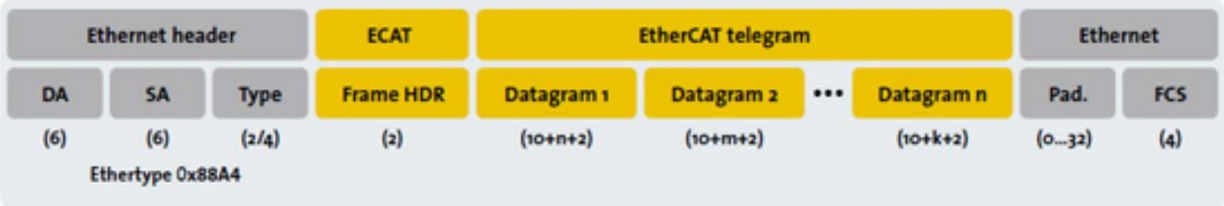
For at kommunikasjonen skal være robust og sikker er det blitt utviklet et teknisk nettverk. Det tekniske nettet består av hovedstyringsenhet og tårnene. Det er også mulig for andre enheter å koble seg på det tekniske nettet og kommunisere med systemet. Dette nettet er passord beskyttet for å sikre at uvedkommende ikke har tilgang.



Figur 30 Teknisk nettverk

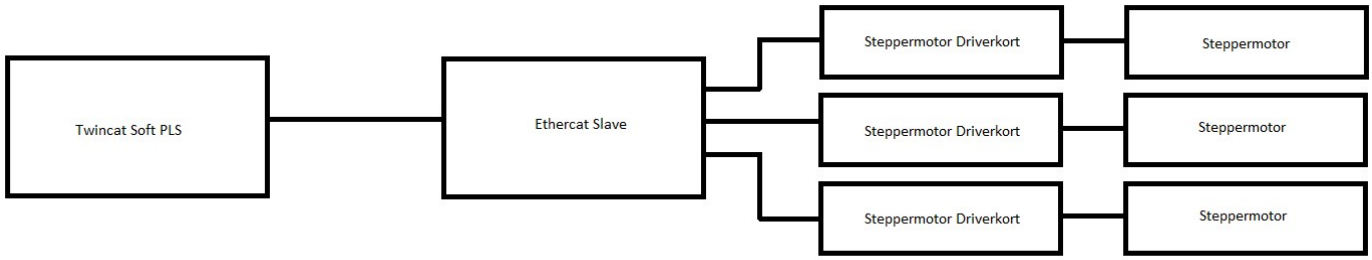
8.2 EtherCAT

EtherCAT er en ledende sanntids kommunikasjonsprotokoll som originalt var utviklet av Beckhoff i 2003 [16]. Hovedtrekkene til EtherCAT er lave sykluser mellom sendinger og lav responstid som gir en rask synkronisering. Kablene som blir brukt til EtherCAT er standard Ethernet kabel. En EtherCAT pakke består av mange datatelegram, hver klient har sitt datatelegram. Når datapakken passerer en slave oppdateres datatelegrammet [16]. På den måten har alle klientene tilgang til alle variablene. På prototypen bruker vi EtherCAT til å kommunisere mellom TwinCAT og EtherCAT slaven. EtherCAT slaven er tilkoblet moduler som driver steppermotoren



Figur 31 Datatelegram i EtherCAT [17]

EtherCAT-nettverk består av en master; Soft PLSen i vårt system og en slave, som er en EtherCAT slave modul. Slave modulen blir brukt til å styre steppermotorene. Her benyttes det en sanntidskommunikasjonsprotokoll for å få motorene til å samarbeide synkront. Dette er en essensiell del av dette prosjektet for å få roboten til å bevege seg med ønsket presisjon.



Figur 32 Nettverkstopologi EtherCAT

8.3 Open platform communication unified architecture(OPC UA)

OPC UA er en plattformuavhengig standard for datakommunikasjon som ble utviklet av OPC foundation i 2008. Et OPC UA nettverk kan bestå av flere servere og flere klienter. En OPC UA server kan bestå av noder, variabler og metoder. For eksempel kan klienten endre verdier eller gjøre en omstart på serveren. Til prosjektet vårt har vi en OPC UA server på hovedstyringsenheten og OPC UA klienter på smartsensorene og i SCADA applikasjonen. [18]

8.3.1 Python 3

Python er et imperativt, dynamisk, objekt orientert språk. Python skiller seg ut fra andre programmeringsspråk ved at det legges stor vekt på enkle syntakser og variabeltyper blir ikke definert i programmeringskoden [19]. Python blir regelmessig oppdatert av Python Software Foundation som er en veldedig organisasjon [19]. Python er et språk som blir sanntidstolket, noe som betyr at programmet blir tolket samtidig som det kjøres. Dette gjør Python litt tregere enn kompilerte språk. Den siste versjonen av Python er Python 3.7, som vi benytter til OPC UA kommunikasjon og avlesning av sensorene på tårnene.

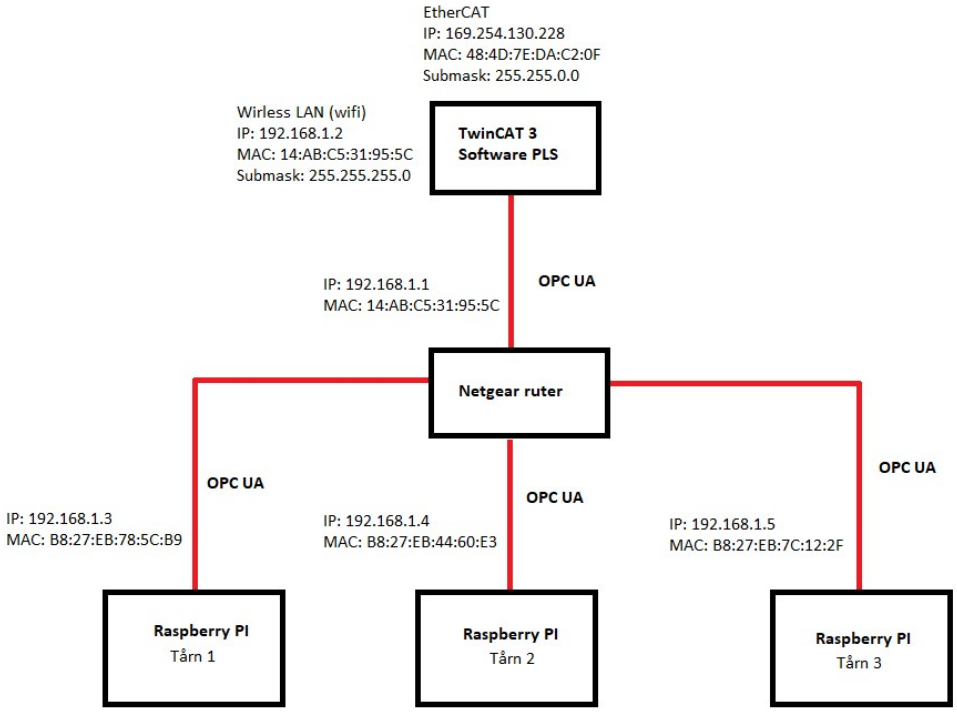
Arkitekturen til OPC UA klienten som har blitt utviklet til smartsensorene, består av en loop som kjører helt til et uhåndtert unntak forekommer i skriptet. En iterasjon av loopen forekommer for hvert tidsintervall som bruker har definert i millisekund. Skriptet leser først av verdiene fra ultralydføleren og ADCen. Den lagrer deretter verdiene og skriver over de valgte OPC UA variablene. På den måten har software PLSen til prosjektet tilgang på verdier fra smartsensoren.

8.3.2 Python og OPC UA

Det har blitt lagt stor vekt på å bruke programmeringsspråkene som passer best til formålet. På den måten sparer vi tid og kvaliteten øker. Vi har utviklet programvare til Raspberry Pi i Python. Dette er det flere grunner til: Python kommer ferdig konfigurert og installert på Raspbian operativsystem, gratis OPC UA klient uten tidsbegrensninger, bra støtte til Raspberry Pi biblioteker og lite resurskrevende [19]. OPC UA blibloteket som blir brukt til gratis og åpen kildekode [20].

8.4 Ruter

Ruteren som benyttes til dette prosjektet er av typen Netgear AC1000. Den trådløse radioen har både 5 GHZ bånd og 2,4 GHZ bånd. Oppgaven til ruteren er å opprettholde det trådløse nettverket, og rute pakkene til rett mottaker. Tradisjonelt sett har også ruteren som oppgave å tildele IP adresser, men det har blitt valgt å gi enhetene statiske IP adresser. Dette er fordi programvaren som er utviklet til dette prosjektet er stilt inn med spesifikke IP adresser. Datablad for ruteren ligger i vedlegg nr. 21.



Figur 33 Oppsett av trådløs kommunikasjon

9 Steppermotorene

I fullskalerte system er motoren en sentral del. Motorer som skal benyttes må kunne styres med høy presisjon. På prototypen er det derfor blitt valgt å benytte steppermotor som innehar disse egenskapene, se vedlegg nr. 14.

9.1 Steppermotoren AS1010-0000

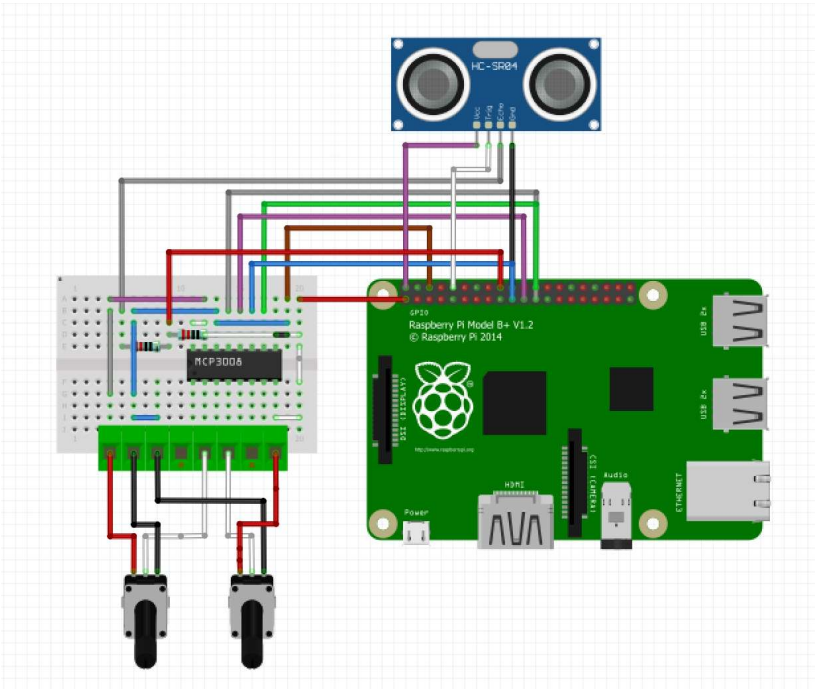
AS1010-0000 er en steppermotor fra Beckhoff. Den benytter likespenning mellom 24-50VDC og har et stille stående moment på 0.38Nm. Steppermotoren er synkron og benytter permanent magneter. Dette gir høy presisjon til steppermotoren. Den har en oppløsning på 200 step per rotasjon, dette gir en oppløsning på 1.8° per stepp. Steppermotoren benytter seg også av delvis stepp noe gir høyre oppløsning. Enkoder er ikke blitt benyttet i dette prosjektet, dette er fordi AS1010-0000 ikke støtter dette.

Mellom steppermotoren og steppermotor driveren EL7031 benyttes det en motorkabel ZK4000-6200-2030, se vedlegg nr. 16. Dette er en 4-leder-kabel med ledertversnitt på 0.75mm^2. Kabelen er elastisk noe som gjør den fleksible ved strekbelastninger på systemet. Motorkabelen kommer med hurtigplugg noe som gjør det lettere å skifte kabelen på modellen ved eventuell slitasje.

Figur 34 Beckhoff steppermotor AS1010-0000

10 Smartsensoren

I de fleste industrielle prosesser er det krav til redundans. Prototypen er avhengig av en nøyaktig posisjonering. Det er derfor nødvendig og ha en grad av redundans til kalkulasjonene våre. For å oppnå dette benytter vi en smartsensor på hvert tårn. Koblingsskjema til smartsensoren kan finnes som vedlegg nr. 15.



Figur 35 Grafisk fremstilling av hvordan smartsensoren er koblet

Smartsensoren består av en Raspberry Pi, en ultralydgiver og to potensiometere. Disse sensorene er koblet til Raspberry Pi via et lite koblingsbrett montert over Raspberry Pien på tårnene. Programvaren som er blitt utviklet for smartsensoren ligger i vedlegg 41. Videre i dette kapittelet blir valgene av sensorer i prosjektet utdypet.

Valg

Selv om industrielle sensorer hadde vært ideelt til dette prosjektet, blir det for kostbart. Det har derfor blitt valgt å forholde oss til utstyr som skolen har tilgang til og kan benytte videre ved en senere anledning.

I dette prosjektet er det behov for tre sensorer per tårn. Det trenges to sensorer for å måle vinkel og en sensor for å hindre at roboten kjører over trinsen. Det er blitt valgt å benytte potensiometre som erstatning for tradisjonelle vinkelmålere da disse forenkler samarbeid med Raspberry Pi. For overkjøringssensor er det blitt benyttet en ultralydsensor. I likhet med potensiometeret er ultralydsensoren typisk skoleutstyr. Da skolen allerede hadde tre ultralydsensorer, slapp vi også å gå til innkjøp av slike.

10.1 Mikrokontroller - Raspberry Pi

Raspberry Pi er utviklet av Raspberry Pi Foundation. Raspberry Pi er utviklet for å gjøre det enklere for studenter å knytte sammen elektronikk og programmering. Operativsystemet som benyttes på Raspberry Pi er Raspbian. Operativsystemet er gratis og er det offisielle operativsystem til Raspberry Pi. Raspbian bygger på Debian er en stabil og skalerbar avart av Linux [20].

Det har blitt installert operativsystemet Raspbian på samtlige Raspberry Pi. Utgivelses datoen til operativsystemet som er installert er 26.11.2018 med Kernel versjon 4. Hovedfunksjonen til Raspberry Pi er å rute målesignaler fra sensorer på tårnet til hoved styringsenheten. Datablad for Raspberry ligger i vedlegg nr.18

Figur 36 Mikrokontrolleren Raspberry Pi

10.2 Potensiometer

Potensiometrene til tårnene blir benyttet for å måle vinklene som hvert tårn skaper relativt til roboten. Målingene i planet skjer ved at ett av potensiometrene er påmontert et hjul. Dette ligger inntil det roterende hodet på tårnet slik at hjulet roterer dersom tårnet roterer. Dette gir vinkelen til linen i forhold til tårnet i xy-planet. Det andre potensiometeret er festet ved trinsen oppå tårnet. Denne måler linens vinkel i z-aksen ved hjelp av en på montert arm beskrevet i kapittelet design.

Seks potensiometre ble innkjøpt til bruk i prosjektet. Det ble dessverre fort klart at disse ikke kunne benyttes da de hadde for høyt dreiemoment. Denne utfordringen ble løst ved at skolen hadde noen trådviklede potensiometer tilgjengelig.

Figur 37 Tårn med roterende hode og smartsensor

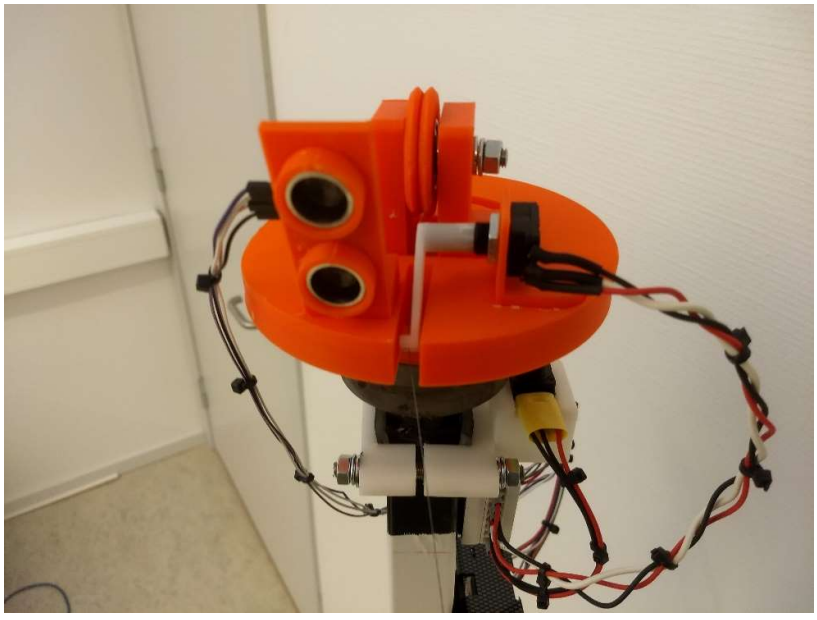
10.3 Ultralydsensor - HC-SR04

Ultralyd sensoren som benyttes i dette prosjektet er av typen HC-SR04. Denne sender et 40KHz sonisk puls hver gang den får et signal fra Raspberry Pi. Når den får tilbake igjen ekkoet av den soniske pulsen sendes et signal tilbake til Raspberry Pi [21].

Ultralyd sensoren er montert etter trinsen, på det roterende hodet til tårnet. Den benyttes som sikring for å hindre at roboten kjører over trinsen ved en eventuell feil. Den er vinklet 14° ned for å sikre at signalet treffer roboten.

Ultralyd sensoren sender ut et digitalt signal, men dette signalet er på 5V. Denne spenningen kan gjøre skade på inngangen til Raspberry Pi. Derfor er den blitt koblet til Raspberry Pi via koblingsbrettet. Her blir deler av signalet ledet til jord gjennom en motstand slik at spenningen er på ca. 3V. Datablad for HC-SR04 er vedlagt i vedlegg nr. 20. Utregning av inngangs spenning blir følgende:

$$Vi = 5V * \frac{470\Omega}{330\Omega+470\Omega} = 2{,}94V \approx 3V$$



Figur 38 topplokk med ultralydgiver og Potensiometer

10.4 Analog til digital konverter - MCP3008

Raspberry Pi kan kun lese inn binære spenningssignaler. For å kunne benytte potensiometrene trenger vi derfor en analog til digital konverterer.
Koblingsbrettet har fått påkoblet en analog til digital konverterer av typen MCP3008. Denne ble valgt ettersom det brukes lineære potensiometere uavhengig av resistansen. MCP3008 kommuniserer med Rasberry Pi ved hjelp av en SPI-protokoll for overføring av binær data [22]. Datablad for MCP3008 i vedlegg 19

11 Posisjonering

I dette kapitelet beskrives posisjoneringen til roboten, det vil si den enheten som beveger seg mellom tårnene. For å løse den aktuelle problemstillingen er det valgt å bruke to matematiske metoder. Disse er trilaterasjon og posisjonering ved hjelp av sensorer.

11.1 Trilaterasjon

Trilaterasjon er en matematisk metode som benyttes på GPS system. Metoden blir brukt for å gi bruker posisjon, fart og høyde. Til posisjoneringen benyttes de matematiske metodene som består av kuler, triangler og sirkler [23].

11.1.1 Arbeidsområde

Arbeidsområdet til roboten blir definert av tårnene. Tårnenes posisjonering står forhold til hverandre i en trekant. Sidene til trekanten er likesidet. Det har blitt valgt å utrykke hver side av trekanten med symbolet l_b. Trekanten tilhører et koordinatsystem. Hvert tårn er gitt som et punkt A_i i koordinatsystemet. Koordinatsystemet blir lagt som et horisontalt plan over toppen til tårnene. Det første tårnet er plassert i origo på koordinatsystemet. Dette tårnet er gitt som punktet $A_1 = (0,0,0)$. Det andre tårnet er definert ved punktet $A_2 = \left(\frac{l_b}{2}, \frac{\sqrt{3}}{2} l_b, 0\right)$. For å finne y-koordinater til det andre tårnet benyttes definisjonen på Pytagoras. Dette gir oss $l_b^2 = (\frac{l_b}{2})^2 + y^2$. Ved å løse utrykket på y ender vi ut med $y = \pm\sqrt{l_b^2 - \frac{l_b^2}{4}} = \pm\frac{\sqrt{3}}{2} l_b$. Dermed kan det velges å definere trekanten i første eller andre kvadrant i koordinatsystem. Det er ønskelig å jobbe langs den positive y-aksen til koordinatsystemet. Derfor ble det valgt å sette y-koordinat til A_2 lik positiv $\frac{\sqrt{3}}{2} l_b$. Det tredje tårnet ligger langs x-aksen. Dermed blir x koordinatene gitt som $x = l_b$. Punktet til det tredje tårnet gitt som $A_3 = (l_b, 0,0)$.

11.1.2 Algoritme

Algoritmen benytter tre teoretiske kuler til å regne ut posisjonen til roboten. Hver kule blir plassert på toppen av hvert tårn. Det horisontale planet skjærer kulene til tre halvkuler. Posisjonen til roboten vil ligge i skjærings punktet mellom halvkulene. Radiusen til halvkulene er gitt ved lengden fra trinsen ut til senter på roboten. Lengden på linene fra trinsen til roboten er beskrevet med $\vec{l}_i$. Avstanden fra enden på robotarmen til senter av robotrammen er gitt ved $\vec{b}_i$. Radiusen til halvkulene blir dermed gitt som den Euklidske normen $r_i = \left\|\vec{l}_i + \vec{b}_i\right\|_2$.

Krumningen på linene er neglisjerbar fordi gravitasjons kraften til roboten holder linnene stramme. Dermed er radiusen til halvkulen alltid en rett linje ut ifra hvert tårn. Roboten har kun 3 graders frihet dermed trengs ikke rotasjonen å bli tatt hensyn til.

For å finne skjæringspunktet mellom halvkulene benytter vi standard likningen for en kule. Den er gitt som

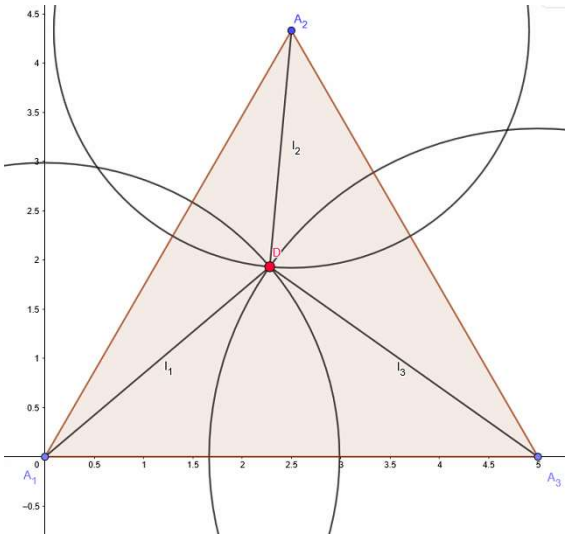
$r^2 = x^2 + y^2 + z^2$.

Likningssettene for å finne skjærings koordinater mellom kulene er

$r_1^2 = x^2 + y^2 + z^2$

$r_2^2 = (x - \frac{l_b}{2})^2 + (y - \frac{\sqrt{3}}{2} l_b)^2 + z^2$

$r_3^2 = (x - l_b)^2 + y^2 + z^2$



Figur 39 Forenklet skisse av skjæringspunkt til kulene i 2D

Ved å løse likning settene på hensyn med x, y og z finner vi punktet til roboten. I likningsett to og tre er det lagt til et offset. Dette er blitt utført for å plassere senteret til kulene på hvert tårn. For å løse likningssettet har vi benyttet MATLAB. MATLAB koden og løsningen på likningen kan finnes som vedlegg nummer 11. Utregningene som har blitt utført er basert på at tårnene er plassert som vist i figur 39. Det benyttes kun negative verdier på z-koordinatet. Grunnen til dette er at nedre dele av halvkulen benyttes og disse ligger under skjæringsplanet.

Dersom det er ønskelig å finne lengden på linene ut ifra koordinatene, settes x, y og z koordinatene inn likningssettet nevnt i på forrige side. Etter likning settet er løst er det viktig å trekke fra lengden $\vec{b}_i$ for hver line. Dette må utføres for å få roboten sentrert i forhold til linene.

11.2 Sensor posisjonering

Sensor posisjonering går ut på å finne posisjonen til roboten ved hjelp av sensorer og matematikk. Her benyttes tidligere nevnte smartsensoren til å detektere de ønskede vinklene.

11.2.1 Arbeidsområdet

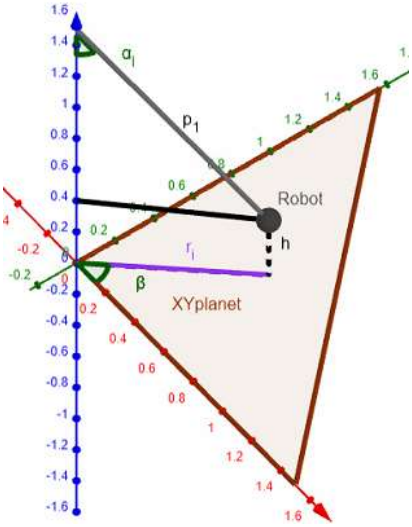
Arbeidsområdet til roboten er definert av tårnene som en likesidet trekant. Tårnenes plassering er på hjørnene til trekanten. Hjørnene defineres ved symbolet A_i. Det første tårnets plassering angir origo til det 3-dimensjonale koordinatsystemet.
Det andre tårnet blir plassert langs y-aksen. Punktet til det andre tårnet er $A_2 = (\frac{l_b}{2}, \frac{\sqrt{3}}{2} l_b, 0)$. For å finne y-koordinatet benyttes definisjonen på sinus. Definisjonen til sinus er gitt ved

$\sin(\theta) = \frac{motstående\ katet}{hypotenu}$.

Hypotenusen til trekanten er gitt som lengden til den likesidete trekanten l_b. Y-koordinatet er gitt som den motstående kateten. Dermed får vi utrykket $\sin(\theta) = \frac{y}{l_b}$. I en likesidet trekant er vinkelen $\theta = \frac{\pi}{3}$. Ved å sette inn vinkelen θ i sinus funksjonen og løse med hensyn på y-koordinaten får vi $y = \frac{\sqrt{3}}{2} l_b$. X-koordinatet er gitt som halve siden til trianglet langs x-aksen. Dermed er koordinatet gitt som $x = \frac{l_b}{2}$. Det tredje tårnet ligger langs x-aksen. Dette punktet er gitt som $A_3 = (l_b, 0,0)$.

11.2.2 Algoritmen

For å finne punktet til roboten benyttes sfæriske koordinater. Lengden på linen til hvert tårn gitt ved symbolet $\vec{l}_i$. Lengden på linen er avstanden mellom ytterpunktet på trinsen og enden på robot rammen. Her legges også til halve robot rammen, som er gitt ved symbolet $\vec{b}_i$. Den totale avstanden mellom tårnet og senter på roboten er gitt ved den Euklidske normen $\rho_{i=} \left\| \vec{l}_i + \vec{b}_i \right\|_2$.



Figur 40 Matematisk skisse av roboten i rommet

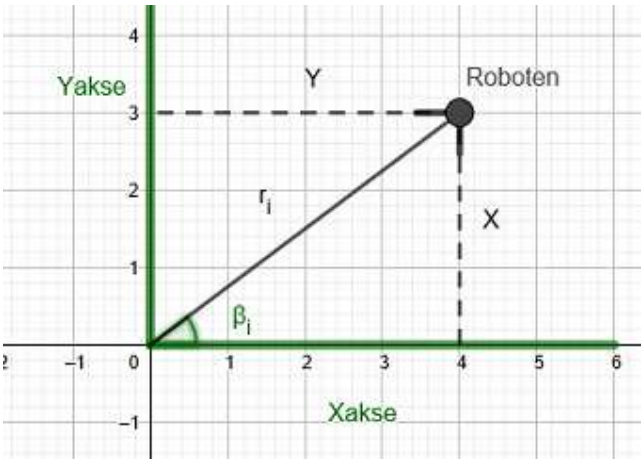
For å finne koordinatene til roboten benyttes vinkelen sendt fra smartsensoren. Den første vinkelen er mellom linen og tårnet. Denne vinkelen er gitt med symbolet α_i. For å finne vinkelen α_i må vi trekke fra 90°. Vinklene på 90° må trekkes fra ettersom sensoren som benyttes måler den øvre vinkelen mellom tårn og trinsen som ligger i åttende oktant. Denne vinkelen er gitt som symbolet β_i . Dermed blir utrykket til $\alpha_i = \frac{\pi}{2} - \beta_i$.

Den andre vinkelen som benyttes er rotasjons vinkelen til tårnet i xy-planet. Vinkelen er gitt med symbolet η_i.

Projeksjonen til p_i finnes i xy-planet. Projeksjonen til p_i i xy-planet er gitt ved r_i. For å finne projeksjonen til r_i i xy-planet benyttes definisjonen av sinus. Dermed kan r_i bli utrykt som $r_i = p_i \cdot \sin(\alpha_i)$. Ved å benytte de trigonometriske likhetene til sinus og cosinus kan vi skrive $r_i = p_i \cdot \cos(\alpha_i)$.

Når lengden r_i er funnet kan man enkelt finne x og y koordinatene. Dette gjøres ved å benytte polare koordinater. I polare koordinater er x og y koordinatene gitt ved følgende formler:

$x = r_i \cdot \cos(\eta_i) = p_i \cos(\alpha_i) \cos(\eta_i)$
$y = r_i \cdot \sin(\eta_i) = p_i \cos(\alpha_i) \sin(\eta_i)$



Figur 41 Projeksjon av roboten i rommet

Den andre delen går ut på å finne robotens z-koordinat. Her benyttes lengden p_i. Utrykket for å finne z-koordinaten til roboten er gitt ved:

$$z = h - p_i \cdot \cos\left(\frac{\pi}{2} - \alpha_i\right) = p_i \cdot \sin(\alpha_i).$$

Der h definere den totale høyden på tårnet. Denne matematiske metoden kan kun benyttes til å finne koordinatene til roboten. Grunnen til dette er at det er robotens bevegelse som angir vinklene på sensorene. Det er mulig å benytte utrykkene til å sjekke om systemet er redundant.

12 Styring

For at roboten skal kunne styres må den kabeldrevet roboten ha en styringsenhet. Videre må det utvikles programvare som kommuniserer med alle styringsenheter til roboten. Det ble valgt å dele styringen inn i tre hoveddeler: Håndkontroller, som er den fysiske enheten som roboten styres med, programmene og bibliotekene knytter det hele sammen.

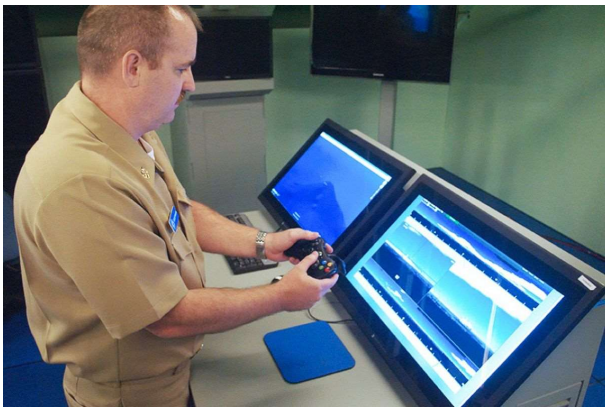
12.1 Håndkontroller

Et av målene som ble satt på prototypen var at den skulle ha to styresystemer. Der den ene innebar en spillkontroller som skulle brukes til å styre roboten. Håndkontrollerstyringen satt vi disse kravene til:

- Styringen skal ha rask respons
- Lett å bruke for en eventuell operatør
- Ergonomisk
- Nøyaktig styring

12.1.1 Ergonomi

I industrien er ikke spillkontrollere så populært. De blir sett som lite ergonomiske, og dyrere alternativer blir derfor ofte brukt. Den siste tiden har det derimot blitt sett flere eksempler på at industrien har åpnet seg mer til bruk av spillkontrollere. Den amerikanske marinen har blant annet byttet ut periskopskontrolleren sin med en Xbox 360 kontroller. Kontrolleren som ble byttet ut hadde en pris på 38 000$ mens Xbox 360 kontrolleren kostet 20$. Den amerikanske marinen sier selv at de bruker nå minutter på opplæring mens med den gamle kontrolleren brukte de flere timer på [24]. Reservedeler er også mer tilgjengelig.



Figur 42 Amerikanske marinen bruk av X-box 360 kontroller [24]

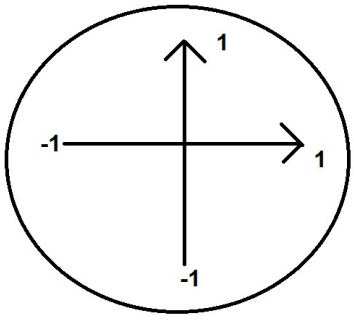
12.1.2 Fra kontroller til Windows

Ettersom hovedstyringsenheten ikke har Bluetooth ble det valgte å koble til spillkontrolleren med USB. Dette ble utført for å få en god overføring av utgangssignalet. For at spillkontrolleren skulle kommunisere med Windows, var det nødvendig med en driver. Valget falt på en applikasjon som heter DS4 Windows, som er en åpen kilde C# applikasjon [25]. Denne applikasjonen simulerer en Xbox kontroller i Windows. Etter å ha installert DS4, fungerer Windows og spillkontrolleren gunstig.

12.1.3 Fra Windows til TwinCAT 3

C# er et høynivå språk som blir kompilert. Dette betyr at kildekoden blir oversatt til maskinkode av en kompilator. C# er et objektorientert språk, og har røtter fra C familien og syntaksene deler flere likheter. I likhet med JAVA har C# en virtuell maskin som leser den kompilerte koden. Dette gjør C# mulig å bruke på andre plattformer [26]. I denne oppgaven benyttes C# til å få utgangssignaler fra en Playstation kontroller opp til TwinCAT 3 ved hjelp av ADS serveren [27]. Biblioteket som ble valgt for å kommunisere med spillkontrolleren, heter Gaming og kommer fra Windows universal pakken. Bibliotek fra Windows Universal pakken kan kun brukes i Windows Universal applikasjoner. Disse applikasjonen kan bare kjøres med operativsystemer fra Microsoft. Dette ser ikke prosjektgruppen som et problem fordi Windows 10 er installert på hovedstyringsenheten.

Programvaren som er utviklet har muligheten til å sende utgangssignalet til ADS serveren som består av en løkke, som kjøres med et brukervalgt intervall. Når løkken kjøres blir utgangssignalet til kontrolleren lagret i en strøm og sendt til ADS serveren. ADS serveren består av en datastruktur som er knyttet til hovedprogrammet på software PLSen. Det er viktig å spesifisere antall bytes som sendes i koden.
Den analoge spaken gir ut to utgangssignaler en for x-aksen og en for y-aksen. Begge disse utgangssignalene gir ut en verdi fra -1 til 1. De analoge signalene fra spaken kan tolkes som vektorer. Knappen gir ut en binær verdi, som kan bli sendt trådløst via Bluetooth eller ved hjelp av USB. Koden som ble utviklet for Dualshock 4 kontrolleren er vedlagt i vedlegg 24.



Figur 43 Joystick akser

12.2 Program

12.2.1 Programmeringsspråk

Det anbefales En To Tre Teknikk å benytte et standardisert programmeringsspråk til styringen av den kabeldrevne roboten. Ved å utvikle den logiske styringen i et standardisert språk har En To Tre Teknikk mulighet til å endre PLS leverandør. Standarden på programmeringsspråket anbefales å følge IEC 61131-3 som støttes av organisasjonen PLC Open [22]. Software PLSen til TwinCAT 3 støtter de fem programmeringsspråkene til standarden IEC 61131-3 [23]. De fem programmeringsspråkene er

- Strukturert tekst
- Instruksjons Liste
- Ladder diagram
- Funksjonsblokkdiagram

- Flytdiagram

Vi har valgt å benytte oss av strukturert tekst for som programmerings språk. Strukturert tekst ble valgt fordi dette ble benyttet i faget prosesstyring EL2-202, og fra oppfordring fra styringsgruppen.

12.2.2 Valg av Applikasjons bibliotek

TwinCAT 3 har flere typer bevegelse bibliotek for programmering av den numeriske kontrolleren ved hjelp av software PLSen. De utvalgte applikasjonsbibliotekene kan benyttes til styringen av roboten. Disse er basert på PLC Opens spesifikasjoner [28]. Eksempler på applikasjons bibliotek som kan benyttes på den kabledrevne roboten er:

- TF5000 NC PTP 10 Axes
- TF5113 Kinematic Transformation L4
- TF5420 Motion Pick-and-Place

De overnevnte applikasjonsbibliotekene har blitt installert på TwinCAT 3. Under utviklingen av styringen valgte vi kun å benytte biblioteket TF5000 NC PTP 10 Axes. Grunnet tidsrammen til prosjektet var det ikke mulig å teste alle bibliotekene.

TF5000 NC PTP 10 Axes inneholder alle de grunnleggende funksjonsblokkene for punkt til punkt bevegelse av roboten. Dette biblioteket er en videre utvikling på Tc2_MC2. Funksjonsblokkene i biblioteket er organisert i organisasjons funksjonsblokker og bevegelse funksjonsblokker.
Organisasjonsblokkene brukes til å klargjøre aksene på den numeriske kontrolleren. Bevegelsesblokkene benyttes til å kjøre roboten.

MC_Reset

MC_Reset benyttes til å resette en akse på den numeriske kontrolleren [28]. På styringen er funksjonsblokken benyttet ved omstart og oppstart av hovedprogrammet.

Figur 44 Funksjonsblokken MC_Reset [28]

MC_Power

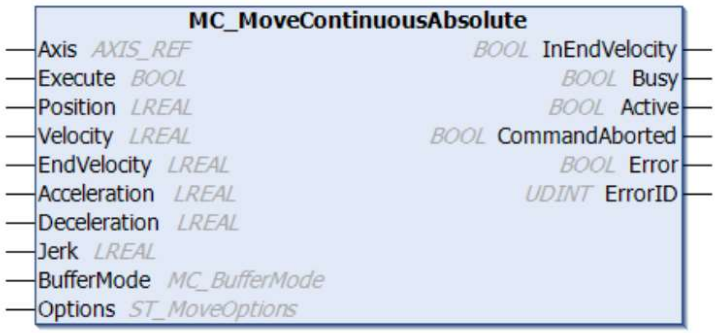
MC_Power benyttes for å aktivere ønskede akser. Her er det mulig å sette bevegelse retningen på steppermotoren. Det er også mulig å sette den maksimale overstyringen til bevegelse kommandoene som påføres steppermotor driveren [29]. Funksjonsblokken har blitt benyttet i oppstartfasen for å definere bevegelse retningen til steppermotoren. Bevegelse retningene som er blitt aktivert på aksene er med og mot klokka. Den maksimale overstyringen er satt til 100%. Dette medfører at den nyeste bevegelses kommandoen overstyrer alle andre påførte bevegelses kommandoer på EL7031.

Figur 45 Funksjonsblokken MC_Power [29]

Det har blitt benyttet flere av de tilgjengelige bevegelse funksjonsblokkene. Disse blir nevnt i kommende avsnitt.

MC_MoveContinuousAbsolute

MC_MoveContinuousAbsolute benyttes til å forflytte steppermotoren en bestemt distanse og fart [30]. På styringen benyttes MC_MoveContinuousAbsolute til å bevege steppermotoren. Hastigheten er satt til 30mm/s. Posisjonen til steppermotoren er kontinuerlig oppdatert. Dette er blitt utført for å få en kontinuerlig forflytning av roboten til ønsket posisjon.



Figur 46 Funksjonsblokken MC_MoveContinuousAbsolute [30]

Akselerasjon, ned akselerasjon og napp beskriver rampingen til steppermotoren på funksjonsblokken. Disse har ikke blitt initialisert med noen verdier. Dette medfører at verdiene blir initialisert fra steppermotor drivernes parametere.

Buffermode inngangen blir også benyttet. Denne inngangen har blitt initialisert med funksjonsblokken MC_Next. MC_Next sørger for at den nyeste bevegelse kommandoen blir kjørt på aksen [31].

MC_Stop

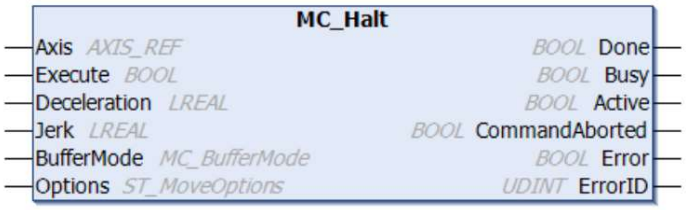
MC_Stop benyttes til å stoppe steppermotoren med spesifisert ramping fra siste kjørte bevegelse kommando. Stoppen som blir utført kan ikke avbrytes av andre bevegelse kommandoer [32].

Funksjonsblokken blir benyttet som nødstopp på styringen. Nødstoppen utføres dersom operatøren ser en farlig situasjon under kjøringen av roboten.

Figur 47 Funksjonsblokken MC_Stop [32]

MC_Halt

MC_Halt benyttes til også til å stoppe steppermotoren med spesifisert ramping fra siste kjørte bevegelse kommando. Stoppen som utføres kan avbrytes av andre bevegelse kommandoer [33].
På styringen benyttes MC_Halt til å stoppe bevegelsen av roboten.



Figur 48 Funksjonsblokken MC_Halt [33]

12.2.3 Kodearkitektur

For å programmere den kabeldrevne roboten må kode arkitekturen være strukturert. Det er viktig å få koden strukturert slik at den blir eksekvert korrekt. Det er også viktig å benytte seg av beskrivende variabelt navn. Dette gjør det enklere og følge koden, samt andre kan lettere bygge videre på koden.

Kodearkitekturen og navn syntaksen som benyttes på styringen kan ikke benyttes på en kabeldrevet robot til En To Tre Teknikk. Grunnene til dette er at den fullskalerte kabledrevet roboten til En To Tre Teknikk vil ha en større og mer kompleks kodearkitektur.

Det har blitt utviklet egne funksjonsblokker for å strukturere de benyttede funksjonsblokkene fra TF5000 NC PTP 10 Axes biblioteket. Utviklede funksjonsblokker på prototypen kan finnes i vedlegg nummer 42.

Strukturen som er blitt benyttet for å sikre korrekt eksekvering av koden er tilstandsmaskin. På en tilstandsmaskin defineres flere tilstander. Hver tilstand beskriver en oppgave på styringen. For å kunne gå mellom tilstanden må en eller flere betingelser være oppfylt. På styringen benyttes det 8 tilstander som blir beskrevet i de neste avsnittene. Tilstandsmaskinens diagram vedlagt som vedlegg nr. 10.

Idle

Dette er startpunktet på tilstandsmaskinen. Her eksekveres kode som skal benyttes i oppstart fasen av hovedprogrammet.

Denne tilstanden er ikke blitt benyttet på styringen til prototypen. Grunnen til dette er prototypen ikke har kode som skal eksekveres under oppstarten av hovedprogrammet. Tilstanden kan benyttes ved videre utvikling av styringen.

Reset

Tilstanden Reset benyttes til å kjøre omstart på motor driveriene. Det må utføres en omstart på motor driverne dersom en feil oppstår eller en nødstopp er utført. Når det er blitt utført en omstart på motor driverne er alle feilmeldinger nullstilt.
Tilstanden blir benyttet for å resette EL7031 ved oppstart av steppermotoren etter utført nødstopp. På styringen benyttes ikke tilstanden reset dersom det oppstår feil under kjøringen av roboten. Dette kan videre utvikles på styringen.

Power

Tilstanden Power skal klargjøre motoren for forflytning av roboten. I tilstanden aktiveres motor driverne. Bevegelse retningen på motorene defineres også her. Begge retninger må brukes for å forflytte roboten til ønsket posisjon.
På protypen benyttes tilstanden Power til å aktivere steppermotor driverne.

Stop

I tilstanden Stop skal robotens bevegelse stoppe. Det er to mulige stopp funksjoner. Disse er nødstopp og normal stop. En nødstopp utføres dersom operatøren ser en farlig situasjon under forflytningen av roboten. Dersom operatøren utfører en nødstopp kan ikke andre bevegelse kommandoer utføres til forflytning av roboten. Den normale stoppen skal utføres når roboten har kommet til en ønsket posisjon. Denne stoppen kan avbrytes dersom operatøren ønsker å forflytte roboten med en ny bevegelse kommando.

På prototypen er det blitt valgt å benytte kontrollerens firkant knapp til normal stopp og sirkel til nødstopp.

Homing

Denne tilstanden benyttes på en oppstart fase på en kabeldrevet robot. Tilstandens oppgave er å kjøre roboten til fast bestemte punkt under oppstart for kalibrering. Dette må utføres for å få kalibrert posisjonen til roboten.

På prototypen var det planlagt å benytte ultralydsensorene på tårnene til å definere fastbestemte hjem punkt. Roboten skulle kjøre mot hvert tårn og ultralydsensoren. Ultralydsensoren skulle måle avstanden mellom roboten og tårnet. Når roboten hadde nådd den bestemte avstanden skulle enkoderen nullstilles. Dette viste seg å være utfordrende, ettersom indikert avstand på ultralydsensor var støyete. Tiden strakk ikke til å lage et filter for å redusere støyen på ultralydesensoren. Dette kan utføres ved en videreutvikling av styringen.

Calculate

Her utføres beregninger til posisjoneringsalgoritmene. Utregninger for de neste forflytning blir også utført i denne tilstanden.

Vi lyktes ikke i å få korrekt posisjonering av roboten. Grunnen til dette var at den indikerte lengden på linene i programmet var feil i forhold til målt lengde. På prototypen er den indikert lengde på linene essensiell for å oppnå korrekt posisjonering. Dersom lengden på linene er feil blir løsningen på likningen som benyttes for å finne punktet til roboten feil.

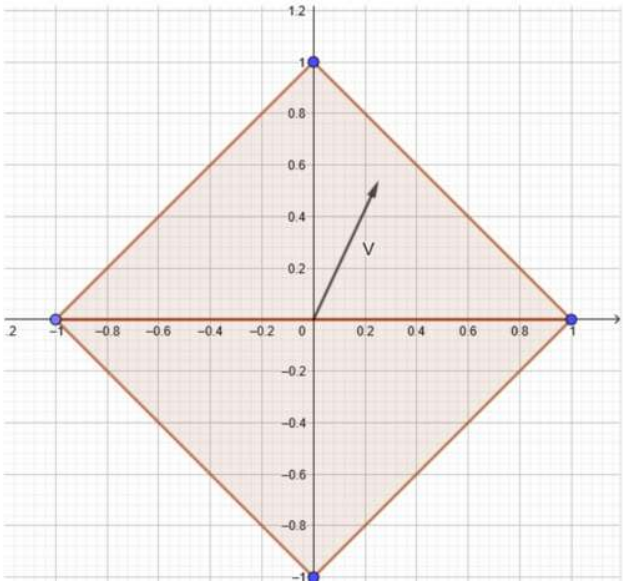
Move

I Move tilstanden skal robotens forflytning utføres. Forflytningen av roboten kan enten bli utført ved direkte innskrivning av posisjon eller manuelt. Før roboten kan forflyttes må operatøren benytte en knapp til aktivering av forflytningen. Dette sikrer roboten mot uønsket forflytning.

En direkte forflytning av roboten kan bli utført ved hjelp av håndkontrollerens joystick. Når operatøren forflytter joysticken skal roboten forflytte seg etter bevegelsen. Det er to metoder for å forflytte roboten ved hjelp av joysticken. Den første metoden går ut

på å forfølge robotens posisjon i forhold til joysticken. Her vil joysticken angi et mønster til roboten. Forflytningsmønsteret blir forfulgt av forflytningsvektoren fra kontrolleren. Den andre metoden går ut på å endre hastighetene og retningene på motoren. Retningen og hastighetene blir gitt ut ifra forflytningsvektoren på joysticken.

Det har blitt utviklet en direkte styring ved hjelp av kontrollens analoge spak. De analoge verdiene fra spaken angir retningene og hastighetene på steppermotorene. For å finne retningen til forflytningsvektoren benyttes det to trekanter. Grunnlinjen til trekantene er plassert langs x-aksen på håndkontrollerens koordinatsystem. Høyden til trekantene ligger langs den positive og negative y-aksen se figur 49



Figur 49 Plassering av trekantene i koordinatsystemet til håndkontrolleren

Dette danner fire kvadranter som forflytningsvektoren kan være innenfor. Kvadrantene forteller retningene på steppermotoren. Dersom forflytningsvektoren er i første eller andre kvadrant beveger roboten seg vekk fra operatøren. Om forflytningsvektoren befinner seg i tredje eller fjerde kvadrant beveger roboten seg mot operatøren.

Det er også blitt utviklet en vertikal styring ved hjelp av to knapper. Knappen som benyttes er L2 og R2. Dersom R2 knappen blir aktivert forflytter roboten seg oppover. For å forflytte roboten nedover benyttes L2 knappen. Før forflyttingen kan utføres må L1 knappen på kontrolleren trykkes. Dette forhindrer en uønsket forflytning av roboten. Kommentert kildekode og programvare tilknyttet styring ligger i vedlegg 42.

Error

Tilstanden Error benyttes til å håndtere feil under driften av den kabledrevne roboten. Noen eksempler på feil kan være

- Posisjonerings feil av roboten
- Kommunikasjons svikt mellom komponenter
- Feil på motor driver

Etter en feil har oppstått må systemet kjøre en omstart. Omstarten som blir utført på systemet avhenger av hvor kritiske feilene er på systemet. Operatøren må også motta en feilmelding i form av alarm på SCADA applikasjonen. Når operatøren har rettet feilmeldingen må den bli kvittert ut.

På prototypen er ikke feil som kan oppstå håndtert. Dette fikk vi ikke tid til, ettersom oppsette av den numeriske kontrolleren tok lengre tid enn antatt. En ny prosjektgruppe bør se nærmer på videre utvikling for håndtering av feilmeldinger.

13 Oppsummering

I dette prosjektet har oppdragsgiver satt som mål å undersøke styringen og posisjoneringen til en kabeldrevet robot. Til dette formålet er det utviklet en prototype av en kabeldrevet robot med tre vinsjtårn. Utviklingen av prototypen er en del av et større samarbeid mellom HVL og En To Tre Teknikk. Der målet er å undersøke om det er mulig å bruke en kabeldrevet robot, for å modernisere det tradisjonelle taubanesystemet. En kabeldrevet robot kan øke sikkerheten for operatøren og hogstarealet kontra, den tradisjonelle tømmerdrift ved taubane.

Prototypens mekaniske deler er blitt produsert som en del av et samarbeid mellom HVL og Øyrane VGS. Samarbeidet gikk ut på at Øyrane VGS konstruerer tårnene i stål etter bachelorgruppens skjematikk. Dette sparte bachelorgruppen for tid og samtidig sikret kvaliteten på prototypen. Produsering av diverse andre deler ble utført av en 3D-printer på skolen, etter prosjektgruppens 3D modeller. 3D printeren har gitt oss fleksibilitet når det kommer til design. Videre er prototypen satt sammen, testet og optimalisert.

Ved kalibrering av steppermotoren var ikke prosjektgruppen fornøyd med nøyaktigheten. Avviket mellom driverkortet til steppermotorens og faktisk posisjon var for stort. Vi har prøvd å kalibrere og optimalisere parameterne, uten at dette ga tilfredsstillende resultat. Dermed anbefales fremtidige prosjektgrupper som skal videreutvikle prototypen, å bytte driverkort. Pris og hvilke driverkort prosjektgruppen anbefaler ligger i vedlegg 13. Det er også mulig å bruke en roterende giver for mer nøyaktig måling av posisjon.

I rapporten er det presentert to fremgangsmåter for å styre prototypen. Første fremgangsmåten benyttes det en håndkontroller for å styre prototypen. Mens den andre styringen består av et styrings og overvåkningssystem (SCADA) som gir operatør mulighet til å angi ønsket posisjon. Videre gir SCADA applikasjonen

operatøren mulighet til å overvåke prototypen visuelt. SCADA applikasjonen vi har utviklet er ikke helt ferdigstilt og bør videreutvikles.

Vi har delt enhetskommunikasjonen til prototypen opp i to lag. Det første laget er kommunikasjon som må forgå i sanntid. Dette er kommunikasjonen som er essensiell for systemets drift. Det første kommunikasjonslaget består av kommunikasjonen mellom steppermotorene og hovedstyringsenheten. Det andre kommunikasjonslaget er ikke essensielt for prototypens drift. Dette er blant annet OPC UA serveren og klientene. Det er også mulig for andre enheter å koble seg på det tekniske nettet og kommunisere med systemet som OPC UA klient. Begge Kommunikasjonslagene fungerer gunstig.

Bachelorgruppen har benyttet et styreskap for å beskytte de essensielle komponentene til systemet. Skapet beskytter mot støv, fukt og støt. Det øker også sikkerheten mot berøring av strømledende deler. Dette er i tråd med Direktoratet for samfunnssikkerhet og beredskap sine forskrifter om elsikkerhet [34].

Det er utviklet flere programvarer knyttet til styringen og kommunikasjon i dette prosjektet. I dette prosjektet har vi prøvd å bruke den programmeringsplattformen som er best egnet for formålet. Kommentert kildekode og prosjekt-filer tilknyttet prototypen fins i vedleggene.

Prosjektgruppen har dermed lyktes i målene om å undersøke styringen, av en kabeldrevet robot. Disse målene ble satt av styringsgruppen og oppdragsgiver. Det er også blitt levert en funksjonell prototype, som det kan forskes videre på.

14 Referanser

[1] D. Runar, «Anleggsmaskinen,» Anleggsmaskinen, 31 August 2016. [Internett]. Available: https://anleggsmaskinen.no/2016/08/tommerdrift-med-taubane/. [Funnet 17 Mai 2019].

[2] V. L. Schmidt, Modeling techniques and reliable real-time implementation of kinematic

s for cable-driven parallel robots using polymer fiber cables, Stuttgart: Franhofer Verlag, 2017.

[3] T.-E. Johnsrud, Skogsdrift og veger i bratt terreng, Biri: Skogbrukets Kursinstitutt, 2007.

[4] E. Rudeck, «Concurrent Engineering,» Concurrent Engineering, 09 mai 2013. [Internett]. Available: https://www.concurrent-engineering.co.uk/blog/blog/bid/93709/what-cad-users-need-to-know-about-direct-modeling. [Funnet 12 mai 2019].

[5] 3DInsider, «3dinsider,» 3DInsider, 2019. [Internett]. Available: https://3dinsider.com/3d-printing-materials/. [Funnet 17 April 2019].

[6] Beckhoff Automation, «Beckhoff Automation,» [Internett]. Available: https://www.beckhoff.com/twincat3/. [Funnet 11 Mai 2019].

[7] Beckhoff Automation, «Beckhoff Automation,» Beckhoff Automation, [Internett]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tf6701_tc3_iot_communication_mqtt/63050395045796619.html&id=. [Funnet 11 Mai 2019].

[8] J. Meyer, «SUPINFO,» SUPINFO International University, 6 Juni 2018. [Internett]. Available: https://www.supinfo.com/articles/single/6958-the-use-of-ide-in-software-development. [Funnet 7 May 2019].

[9] E. Graff og B. Nathan, «Micrsoft Corporation,» Micrsoft Corporation, 20 April 2017. [Internett]. Available: https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/user-mode-and-kernel-mode. [Funnet 10 Mai 2019].

[10] Beckhoff, «Beckhoff,» 28 Februar 2019. [Internett]. Available: https://download.beckhoff.com/download/document/io/ethercat-terminals/el70x1en.pdf. [Funnet 5 Mai 2019].

[11] Novotek, «Novotek,» Novotek, [Internett]. Available: https://www.novotek.com/en/solutions/kepware-communication-platform/opc-and-opc-ua-explained. [Funnet 11 Mai 2019].

[12] elprocus, «https://www.elprocus.com/,» elprocus, [Internett]. Available: https://www.elprocus.com/scada-systems-work/. [Funnet 19 05 2019].

[13] Unity, «Unity,» Unity, [Internett]. Available: https://unity3d.com/unity. [Funnet 5 Mai 2019].

[14] R. A. Knut, «Stor Norske Leksikon,» 17 April 2019. [Internett]. Available: https://snl.no/kapslingsgrad. [Funnet 11 Mai 2019].

[15] Arbeidstilsynet, «Arbeidstilsynet,» Arbeidstilsynet, [Internett]. Available: https://www.arbeidstilsynet.no/tema/byggesak/sjekkliste-for-krav-til-fysisk-arbeidsmiljo/. [Funnet 16 Mai 2019].

[16] EtherCAT Technology Group, «Ethercat,» 11 mai 2019. [Internett]. Available: https://www.ethercat.org/en/technology.html.

[17] gantner-instruments, «gantner-instruments,» gantner-instruments, 22 oktober 2015. [Internett]. Available: https://www.gantner-instruments.com/gi-magazine/ethercat-based-measurement/. [Funnet 15 mai 2019].

[18] opcfoundation, «opcfoundation,» 11 Mai 2019. [Internett]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/.

[19] Python Software Foundation, «Python Software Foundation,» Python Software Foundation, [Internett]. Available: https://www.python.org/psf/. [Funnet 17 Mai 2019].

[20] O. Roulet-Dubonnet, «https://python-opcua.readthedocs.io/en/latest/,» https://python-opcua.readthedocs.io/, 2015. [Internett]. Available: Olivier Roulet-Dubonnet. [Funnet 20 05 2019].

[21] Raspbian, «Raspbian,» 13 Mars 2019. [Internett]. Available: https://www.raspbian.org/RaspbianFAQ.

[22] ELEC Freaks, «mouser.com,» ELEC Freaks, [Internett]. Available: https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf. [Funnet 11 05 2019].

[23] MicroCHip, «Adafruit.com,» [Internett]. Available: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf. [Funnet 8 Mai 2019].

[24] F. Zahradnik, «Lifewire,» 26 November 2018. [Internett]. Available: https://www.lifewire.com/trilateration-in-gps-1683341. [Funnet 11 Mai 2019].

[25] S. Liao, «The Verge,» The Verge, 19 September 2017. [Internett]. Available: https://www.theverge.com/2017/9/19/16333376/us-navy-military-xbox-360-controller. [Funnet 2019 Mai 09].

[26] Jays2Kings, «github,» github, 9 Oktober 2016. [Internett]. Available: https://github.com/Jays2Kings/DS4Windows. [Funnet 2019 05 20].

[27] P. Hudak, «cs.yale,» Yale, 2006. [Internett]. Available: http://www.cs.yale.edu/homes/hudak/CS112F06/java.html. [Funnet 2019 05 20].

[28] Beckhoff, «Beckhoff,» Beckhoff, ads. [Internett]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcadscommon/html/tcadscommon_intro.htm&id=. [Funnet 20 05 2019].

[29] Beckhoff Automation, «Beckhoff Automation,» Beckhoff Automation, [Internett]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_mc2/45035996343852555.html&id=. [Funnet 11 Mai 2019].

[30] Beckhoff Automation, «Beckhoff Automation,» Beckhoff Automation, [Internett]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_mc2/36028797089014923.html&id=3140016792716102319. [Funnet 12 Mai 2019].

[31] Beckhoff Automation, «Beckhoff Automation,» Beckhoff Automation, [Internett]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_mc2/36028797089013387.html&id=8896480812176891133. [Funnet 12 Mai 2019].

[32] Beckhoff Automation, «Beckhoff Automation,» Beckhoff Automation, [Internett]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_mc2/54043195598549899.html&id=. [Funnet 12 Mai 2019].

[33] Beckhoff Automation, «Beckhoff Automation,» Beckhoff Automation, [Internett]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_mc2/45035996343813515.html&id=7185542268581141649. [Funnet 12 Mai 2019].

[34] Beckhoff Automation, «Beckhoff Automation,» Beckhoff Automation, [Internett]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclib_tc2_mc2/36028797089070987.html&id=4471742788717251015. [Funnet 12 Mai 2019].

[35] Direktorat for samfunnsikkerhet og beredskap, FEL, OSLO: Direktorat for samfunnsikkerhet og beredskap, 1998.

[36] Unity, «Unity.com,» Unity, u. [Internett]. Available: https://unity3d.com/legal/trademarks?_ga=2.220843800.1661326651.1558308276-105012422.1556552198. [Funnet 20 mai 2019].

15 Vedlegg

Vedlegg1: Tegning til hjul for potensiometer

Vedlegg2: Tegning topplokk

Vedlegg3: Tegning trinse

Vedlegg4: Tegning trommel

Vedlegg5: Tegning brakett ultralydsensor

Vedlegg6: Tegning vinkel for trinse

Vedlegg7: Tegning brakett til potensiometer

Vedlegg8: Tegning holder for potensiometer

Vedlegg9: Tegning arm til potensiometer

Vedlegg10: Tilstandsmaskin

Vedlegg11: Matlab kode

Vedlegg12: Datablad EK1100

Vedlegg13: Tilbud på driverkort fra bechoff

Vedlegg14: Datablad steppermotor AS1010

Vedlegg15: Skjematikk for RaspBerry PI

Vedlegg16: Datablad ZK4000-6200

Vedlegg17: Datablad driverkort for steppermotor

Vedlegg18: Datablad RaspBerry Pi

Vedlegg19: Datablad MCP3008
Vedlegg20: Datablad HCSR04-1022824
Vedlegg21: Datablad Ruter
Vedlegg22: Datablad spenningsforsyning
Vedlegg23: Datablad for hovedstyringsenheten
Vedlegg24: Kode for Dualshock4
Vedlegg25: G-kode
Vedlegg26: Modell arm til potensiometer
Vedlegg27: Modell roterende hode
Vedlegg28: Modell topplokk
Vedlegg29: Modell trommel
Vedlegg30: Modell ultralydbrakett
Vedlegg31: Modell bunnplate og tårn
Vedlegg32: Modell holder for potensiometer
Vedlegg33: Modell brakett til potensiometer
Vedlegg34: Modell hjul til potensiometer
Vedlegg35: Tegning feste til firkantrør
Vedlegg36: Kode Smartsensor
Vedlegg37: Tegning klemmhylse
Vedlegg38: Tegning snitt sammensatt roterende hode
Vedlegg39: Tegning sammensatt roterende hode
Vedlegg40: Gant diagram
Vedlegg41: Smartsensor kode
Vedlegg42: TwinCAT programvare
Vedlegg43: Regnskap
Vedlegg44: Brukernavn og passord
Vedlegg45: SCADA
Vedlegg46: SCADA ferdig kompilert
Vedlegg47: Gant forventet