Høgskulen på Vestlandet

BACHELOROPPGAVE:

BO21E-25 VIDEREUTVIKLING AV JERNBANEMODELL

Anne Løvfall Våge
Maren Aamodt
Mathias Duesund

26. mai. 2021

ABB

Dokumentkontroll

Rapportens tittel: BO21E-25 Videreutvikling av jernbanemodell	*Dato/Versjon* 26. mai. 2021/1.0
	Rapportnummer: BO21E-25
Forfatter(e): Anne Løvfall Våge Maren Aamodt Mathias Duesund	*Studieretning:* EAU18
	Antall sider m/vedlegg 36
Høgskolens veileder: Geir Omar Berland	*Gradering:* Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: Firma	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): Jørgen Bull Marcus Aleksander Stenhjem	

Revisjon	**Dato**	**Status**	**Utført av**
0.1	26.04.21	Utkast	Alle
0.11	28.04.21	La til kapittel 1.2	Anne
0.12	23.05.21	Ferdigstilling av tekst	Alle
0.13	25.05.21	Lagt til bilder og vedlegg	Alle
1.0	26.05.21	Endelig versjon	Alle

Forord

Den følgende rapporten er et resultat av et semester med mye lærdom og artige hverdager. De færreste kan leke med en jernbanemodell og få studiepoeng for det, noe vi har satt veldig pris på i en ellers teoretisk preget skolehverdag. Dette har etter vår mening vært den perfekte avslutning på tre spennende år ved Høgskulen på Vestlandet campus Bergen.

Vi ønsker derfor å rette en stor takk til ABB for å tilby denne oppgaven. Våre kontaktpersoner fra ABB, Jørgen Bull og Marcus Aleksander Stenhjem, fortjener en ekstra takk for å ha veiledet og motivert oss til å fullføre denne oppgaven. Vi har også hatt en god sparringspartner som vi ønsker å takke, ved HVL veileder Geir Omar Berland som har kommet med gode forslag og stilt spørsmål som har utfordret oss til å prestere bedre. Det må også rettes en takk til Mathias Duesund, som på grunn av covid-19 situasjonen, har hatt en jernbane stående i stuen sin hele våren.

Sammendrag

ABB har de siste to årene dannet bacheloroppgaver rundt en jernbanemodell, der tanken er at oppgaven skal videreutvikles hvert år. Årets oppgave er å etablere dynamiske elementer som kan påvirke togenes sikkerhet, og deretter løse sikkerhetsutfordringene knyttet til dette. Siden oppgaven er relativt åpen og kan gjennomføres på flere måter, fikk vi være med på å forme den til å bli mer konkret. Oppgaven ble å finne en løsning for å oppdage hindringer i sporet som kan utgjøre en fare for toget. I tillegg har vi sett på et par mindre utfordringer knyttet til sikkerheten.

Vår løsning involverte å montere to nye kamera med vidvinkellinse for å få kameraovervåking i svingene. Kameraene ble programmert til å sammenligne et referansebilde med et «nå»-bilde og lete etter endringer, disse vil tolkes som hindringer. Løsningen klarer å skille sporene på langsidene fra hverandre og det er dermed mulig å oppdage om hindringen dekker begge spor eller bare ett av dem. I svingene var dette ikke mulig da objekter i ytterste spor kunne skygge for objekter i innerste spor. Løsningen i svingene ble dermed å se på hele svingen som en sone og kun la ett tog kjøre gjennom om gangen. Løsningen vår oppfyller kravet om å unngå kollisjon minst ni av ti ganger, og vil oppdage hindringer på både langsidene og i svingene.

Dette har vært en spennende og lærerik oppgave. Vi har anvendt mye av kunnskapen vi har lært på HVL de siste tre årene, men også tilegnet oss ny kompetanse innen blant annet kameraprogrammering. Gruppen har opprettholdt et godt samarbeid gjennom oppgaven, og unngått å spore av. Det ble lagt en god plan i oppstartsfasen som gjorde det enkelt å holde seg i rute.

1 Innhold

Figur

Tabell

1 Innledning

1.1 Oppdragsgiver

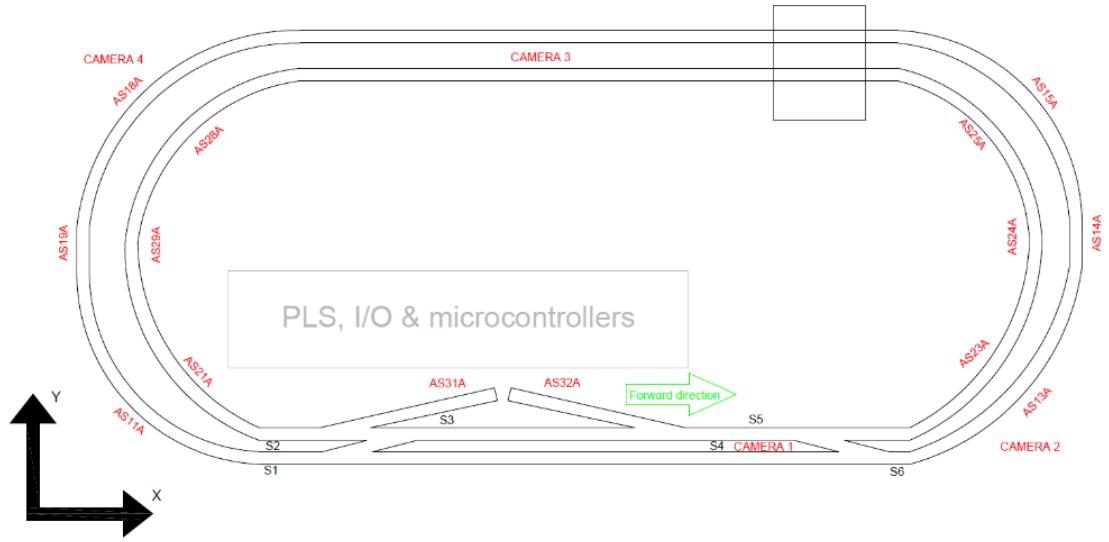
ABB, som er en forkortelse av «Asea Brown Boveri», er en sveitsisk – svensk multinasjonalt selskap med hovedkvarter i Zurich, Sveits. De jobber med programvare som kobles sammen med produkter innen elektrifisering, automasjon, robotikk, motorer og omformere. Deres fremgang bygges videre på mer enn 130 års teknologisk lederskap og de har rund 110 000 ansatte i 100 land [1].

1.2 Historikk

ABB har de siste to årene lagd bacheloroppgaver rundt en jernbanemodell, der tanken er at oppgaven skal videreutvikles og utvides hvert år.

Bacheloroppgaven første året, 2019, gikk ut på å designe og bygge jernbanemodellen, og implementere et system for å registrere posisjonen til toget. I tillegg til dette ble det laget sikkerhetssystemer for å unngå kollisjon. For å registrere posisjonen til togene ble det tatt i bruk magnetbrytere, som ble montert jevnt utover hele banen. Siden det er ABB sin oppgave, var det naturlig å bruke deres kontrollsystem 800xA. For kommunikasjon mellom PLS og magnetbryterne blir det brukt to ulike Arduinokort, UNO og mega. Det ble også montert på åtte servomotorer, seks til sporbytte og to til bommene ved sporovergangen. I ABB sitt programvare Control Builder ble det laget tre ulike sikkerhetssystem; antikollisjon, fartsredusering og sikkert sporbytte. Videre konstruerte gruppen et HMI-bilde av jernbanen og fikk markert togenes posisjon på det. [2]

Oppgaven året etter, i 2020, var å implementere analog feedback på togenes posisjon og gjøre det mulig å kjøre mer enn to tog samtidig. Gruppen bestemte seg for å bruke mikrokontrollerkamera av typen OpenMV H7. For å identifisere og måle posisjon til togene med kameraene, fikk gruppen laget AprilTags. Disse kan minne om en QR-kode. Kameraene «ser» når en AprilTag er i bildet og regner ut posisjonen til toget ut ifra x- og y-koordinatene til AprilTag. Denne typen posisjonsmåling ble tatt i bruk på langsidene på banen, men ikke i svingene der det fortsatt blir brukt magnetbrytere. I tillegg ble PLS-koden og HMI-bildet oppdatert til at tre tog kan kjøres og styres samtidig [3].



Figur 1 - Oversiktsillustrasjon fra 2020

1.3 Problemstilling

Gjennom årets bacheloroppgave ønsker ABB å videreutvikle modellen ved å etablere dynamiske elementer som kan påvirke togenes sikkerhet, og deretter løse sikkerhetsutfordringene slik at togene tar hensyn til farer i omgivelsene. I tillegg ønsket oppdragsgiver at det skulle utforskes om det var mulig å få til posisjonsmåling med kameraene i svingene, da dette ikke ble gjort i fjor.

Etter samtale med veileder fra HVL ble vi gjort oppmerksom på en annen problemstilling: hva skjer dersom man mister kontakt med et tog? Å miste kontakt med et tog defineres her som at toget ikke kan styres.

Vi hadde dermed tre problemstillinger vi ønsket å løse. Vi prioriterte problemstillingen som omhandler dynamiske elementer, siden det er oppgaven gitt av ABB. Dersom vi fikk tid ønsket vi å se om vi kunne finne en løsning på bruk av kamera i svingene for mer nøyaktig posisjonsmåling og hva som skjer om man mister kontakten med togene. Vi satte opp følgende prioriteringsliste:

1. Oppdage hindringer i sporet
2. Posisjonsmåling i svingene med kamera
3. Kommunikasjonssvikt med tog

2 Kravspesifikasjon

Oppgaven gitt av ABB var svært åpen og med kun ett krav om å etablere dynamiske elementer som kan påvirke togenes sikkerhet, og deretter løse disse sikkerhetsutfordringene. Siden dette ikke var et målbart krav, valgte vi å sette et mer konkret krav selv. Vi ønsket å danne et sikkerhetssystem som oppdaget objekter i togsporet og hindret kollisjon minst ni av ti ganger.

Videre ønsket vi å utforske mulighetene for posisjonsmåling i svingene ved bruk av kameraene. Siden det allerede var posisjonsmåling ved bruk av magnetbrytere, ønsket vi å oppnå minst like god posisjonsmåling ved bruk av kamera. For å finne ut om det i hele tatt var mulig å finne en god løsning på dette ved bruk av kamera, måtte det gjennomføres en del tester på kameraene og jernbanen.

Vi ble gjort oppmerksom på at det ikke var spesifisert hva som skjer dersom man mister kontakt med ett eller flere av togene. Dette ønsket vi å se nærmere på og etablere et sikkerhetssystem som sørger for at ingen tog kolliderer.

3 Analyse av problemet

Som introdusert i kapittel 1.3 delte vi oppgaven inn i tre ulike deler: oppdage hindringer i sporet, posisjonsmåling i sving, og kommunikasjonssvikt med tog. I dette kapittelet vil alle problemstillinger presenteres og mulige løsninger bli reflektert over.

3.1 Problemstillingene

3.1.1 Oppdage hindringer i sporet

Jernbanen er formet som en ellipse og kan deles inn i to langsider og to svinger. De to delene byr på ulike utfordringer, og potensielt ulike løsninger. På hver av langsidene er det montert to kameraer som kan brukes. Magnetbryterne som ble montert første året på langsidene, er delvis fjernet og flyttet til svingene for mer nøyaktig posisjonsmåling. Dermed er det to kameraer som kan brukes på hver

langside, i motsetning til i svingene der det kun er magnetbrytere. Å oppdage hindringer i svingene vil kreve en mer kompleks løsning som tar hensyn til krumningen til skinnene.

3.1.2 Posisjonsmåling i sving

Å måle posisjon i svingene med kamera er en utfordring. Dette ble ikke gjennomført i fjor, da løsningen deres krevde flere kameraer. Denne delen av oppgaven ble dermed lagt til sides [3]. Et viktig punkt å notere seg er at de kameraene som allerede er monter har en synsvinkel på 70,8°. Dette vil si at dersom vi skal ha kameradekning gjennom hele svingen med denne typen kameralinse, trengs det flere kameraer. OpenMV selger vidvinkellinser som har synsvinkel på 99° som kan være et alternativ å ta i bruk, da vil vi kun trenge ett kamera per sving. Et annet aspekt av denne problemstillingen er hvor kameraet(ene) skal plasseres for best mulig oversikt over begge sporene gjennom svingen.

Figur 2 – Standardlinse

Figur 3 - Vidvinkellinse

3.1.3 Kommunikasjonssvikt med tog

Det er kritisk at man alltid vet hvor alle togene står plassert på banen for å unngå kollisjon. Men hva skjer om man mister kommunikasjonen mellom tog og skinner? Dette kan føre til at et tog ikke kjører fremover selv om det får beskjed om å gjøre det. I så tilfelle er det viktig at man har et sikkerhetssystem som registrerer hvor det står tog, og hindrer kollisjon. Her kan det være aktuelt å ta i bruk informasjonen fra kameraene, som forteller om det er en AprilTag i bildet. I koden til kontrolleren har hvert tog variablene «posisjon» og «neste posisjon», som kan brukes for å registrere sist kjente posisjon til et tog.

3.2 Utforming av mulige løsninger

3.2.1 Oppdage hindringer i sporet

Fra oppdragsgiver ble det lagt frem et forslag om å bruke en bil som hindring i sporet. Selv ønsket vi å lage en universell løsning som ville reagere på alle gjenstander som er store nok til å påføre skade på togene. Vi prioriterte å lage en god løsning som detekterte objekter i begge togsporene til å begynne med, etterpå kunne vi se på løsninger der vi detekterte hvilket spor det dreier seg om.

3.2.1.1 Løsningsforslag 1 – Sporing av merker

Ved å benytte kameraene som allerede er installert på modellen er det mulig å ikke bare observere togene med AprilTags, men likeens andre selvdefinerte «merker». Et slikt merke kan for eksempel være en lyskilde, da dette vil skille seg ut på kameraet som en hvit prikk. Ved å plassere disse merkene

på enden av banen, i sikt for et kamera, kan det detekteres om noe blokkerer sikten til merket. Vi kan fortelle kameraet hvor mange merker det skal se, og dersom kameraet ikke ser alle kan det antas at det er objekter i togsporene som kan skape en fare for togene.

Per i dag er det ikke kamera i svingene på banen. Ved å montere et kamera med en vidvinkel linse kan vi få oversikt over sporet i hele svingen.

3.2.1.2 Løsningsforslag 2 – Sammenligne bilder

Kameraene som er montert kan sammenligne bilder. Det vil si at kameraet kan finne differansen mellom to forskjellige bilder, dette konseptet kalles «frame differencing». Ved å benytte oss av sammenligning har kameraet mulighet til å oppdage forandringer eller bevegelse i omgivelsene. Det blir dermed mulig til å oppdage objekter i togsporene og varsle om dette. Det er usikkert hvor følsom denne sammenligningen er, og om den er sterkt påvirket av endring i lysintensitet eller bakgrunnsstøy.

3.2.2 Posisjonsmåling i sving

Modellen har allerede posisjonsmåling med kameraer på langsidene, i tillegg er det fortsatt måling med magnetbrytere i svingene.

3.2.2.1 Løsningsforslag 1 – Vidvinkel kameralinse

Vi ønsket å utforske videre fjorårets løsningsforslag om å bruke kameraer. Ved å montere ett nytt kamera i hver sving med en vidvinkellinse, åpner vi opp for å teste dette ut. Det ble nevnt innledningsvis at det er flere detaljer man må bestemme som kan være kritisk for at dette skal lykkes. En av disse er plassering av kamera på banen og i høyden. Noe annet som må testes ut er om det er mulig å se AprilTags til togene gjennom hele svingen, da AprilTags må leses av nokså vinkelrett. Når vi skal programmere sving-kameraene må oppløsningen på bildet bestemmes. Dersom den blir for stor kan det bli problematisk å lese av AprilTags.

3.2.2.2 Løsningsforslag 2 – Estimering av posisjon

Ved å beregne hvor fort togene går og hvor de er plassert, har vi mulighet til å estimere en mer nøyaktig posisjonsmåling. Dette kan være et godt alternativ til kameraovervåking, da det i de fleste tilfeller vil være godt nok. For at dette skal være et alternativ er vi avhengig av at den fysiske farten til togene er lik når de settes til lik fart i kontrolleren, men dette må testes ut.

3.2.3 Kommunikasjonssvikt med tog

3.2.3.1 Løsningsforslag 1 – Full stopp

Dersom vi mister kontakt med ett tog vil de resterende togene som befinner seg på banen stoppe, for å forsikre at det ikke forekommer en kollisjon. Dette er en drastisk løsning som tilbyr lite fleksibilitet, men ved å bruke denne minimeres sjansen for kollisjon og sikkerheten blir ivaretatt.

3.2.3.2 Løsningsforslag 2 – Trygg plassering

En annen løsning er å finne ut hvor vi mistet kontakten med toget, og deretter bestemme hvor det vil være trygt for de gjenværende togene å stoppe. Denne løsningen er mer elegant og vil gi et mer realistisk resultat enn full stopp. Med denne løsningen må vi vite at toget vi mistet kontakt med har stoppet, men dette er informasjon vi ikke nødvendigvis har. Dersom dette viser seg å være vanskelig å

finne ut, kan man se på løsninger som innebærer å kjøre togene inn på ulike spor og isolere toget vi mistet kontakt med i eget spor.

3.3 Vurderinger i forhold til verktøy og HW/SW komponenter

Siden jernbanemodellen bygger på to tidligere bacheloroppgaver, eksisterer det allerede et utvalg av komponenter som er montert og integrert.

Selve hjernen til hele jernbanemodellen er ABB sin kontroller AC800M med tilhørende programvare 800xA. Disse hadde vi lite forkunnskaper om, men alle gruppemedlemmene var kjent med kontrollsystemer fra andre produsenter, i tillegg ville det bli utlevert dokumentasjon om den av ABB. Gruppene første og andre året brukte mellom 20 og 25 timer i kursing av 800xA, og vi estimerte dermed å bruke like lang tid.

Fra tidligere gjennomførte emner i utdanningen hadde vi opparbeidet god kjennskap til Arduino og dens applikasjoner. Arduino er en sentral del av denne oppgaven og utfordringen her var å sette seg inn i koden som allerede er programmert.

Kameraene som allerede var montert på modellen er av typen OpenMV H7, og blir programmert i MicroPython. Dette programmeringsspråket var nytt for alle tre, etter som det kun er C++ og C# vi har forkunnskaper i. Å lære seg MicroPython ble sett på som en gøy utfordring og vi regnet med det tok cirka to dager å lære seg syntaks og hvordan kameraene fungerer. Denne estimeringen ble tatt på bakgrunn av tilgjengelige datablad, OpenMV sine godt dokumenterte nettsider, og de utallige nettforumene som finnes på kameraene.

Disse komponentene, samt en del andre, vil bli gått dypere inn på i kapittel 4, Maskinvare og programvare. Her blir i tillegg funksjonene deres forklart.

3.4 Konklusjon

En av fellesfaktor for løsningene er at det trengs to nye kameraer til svingene, og da gjerne med en vidvinkellinse. Kostnadene vil dermed bli noe det samme uavhengig av hvilke løsninger vi velger.

3.4.1 Oppdage hindringer i sporet

For denne problemstillingen var det som nevnt tidligere to mulige løsninger som ble vurdert, sporing av merker og sammenligning av bilder. Det var usikkert hvor godt sammenligning av bilder fungerte, og hvor lett påvirkelig den var av ytreforstyrrelser, som for eksempel endring i lysforhold. Mikrokontrolleren med kameraet hadde ikke tilstrekkelig lagringsplass til å kunne foreta sammenligning av bilder, men det var en inngang til SD-minnekort. Med andre ord for å kunne ta i bruk løsningsforslag 2, å sammenligne bilder, måtte det kjøpes inn seks minnekort i tillegg til to nye kameraer med vidvinkellinse. For løsningsforslag 1, sporing av merker, måtte det kjøpes inn en del ekstra utstyr, i tillegg til kameraer med vidvinkellinse.

Det var med tanke på dette det ble bestemt å utforske sammenligning av bilder videre og teste ut konseptet på de eksisterende kameraene. Dersom dette ikke ga et godt nok resultat, ville vi gå over til å bruke sporing av merker i bildene. Denne konklusjonen ble støttet av oppdragsgiver, ABB.

3.4.2 Posisjonsmåling i sving

Med tanke på valgt løsningsforslag til problemstilling 1 og innkjøp som følger denne, ville vi prøve å bruke kamera(er) med vidvinkellinse. Som nevnt i 3.2.2.1, Løsningsforslag 1 – Vidvinkel kameralinse, kreves det mye vellykket testing for å kunne ta i bruk dette forslaget. Dersom dette viste seg å være vanskeligere enn først antatt, var Løsningsforslag 2 – Estimering av posisjon en god reserve. Dette alternativet har i tillegg et par testforutsetninger som var kritiske. Det ble vurdert underveis om det ville gi et godt nok resultat i forhold til posisjonsmålingen som allerede var etablert i svingene med magnetbryterne.

3.4.3 Kommunikasjonssvikt med tog

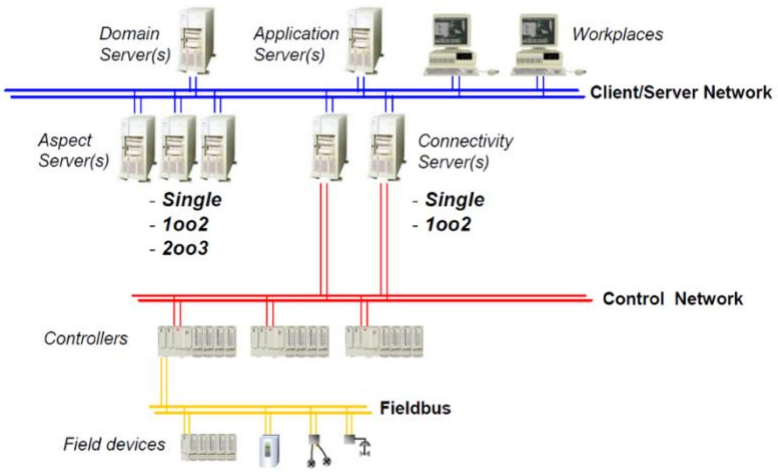
Denne problemstillingen er svært kritisk, og løsningsalternativene er ulike. Den sikreste løsningen er å stoppe alle tog på banen. Å kjøre togene til en trygg posisjon er det beste alternativet, men vil skape flere farlige situasjoner. Dermed vil vi ikke komme med en konkret konklusjon helt enda. Vi ønsker å kjøre tester og utspille ulike senarioer på jernbanemodellen for å se hvilket løsningsforslag som bidrar til høyest sikkerhet og hva som passer best inn med løsningen til de andre problemstillingene.

4 Maskinvare og programvare

I dette kapittelet kan man lese om maskinvare og programvare som er brukt under oppgaven. Det vil også være en forklaring på funksjon og bruksområde.

4.1 Nettverks Topologi

Som nevnt i kapittel 3.3, bruker jernbanen 800xA fra ABB som kontrollsystem. Et typisk 800xA system består av ulike typer utstyr og nettverk. Øverste del av nettverket består av servere og datamaskiner og blir kalt klient-/servernettverk. I denne delen av nettverket finner man domeneservere, applikasjonsserver og klienter. Videre er det koblet opp mot et eller flere kontrollnettverk via en tilkoblingsserver. I kontrollnettverk finner man kontrolleren, som i dette tilfelle er en AC 800M PM864. Hver kontroller har en egen feltbuss som igjen har flere enheter som styres.



Figur 4 - Nettverks topologi

4.1.1 Elementer av 800xA

4.1.1.1 Domeneserver

Domeneserveren blir brukt til å konfigurere sikkerheten for systemet. Denne blir viktigere i større system der flere klienter er koblet til, og flere brukere har tilgang til systemet. Serveren holder kontroll på alle brukere og kontrollerer tilgangene deres og hva som blir endret av hver enkelt bruker.

4.1.1.2 Applikasjonsserver

Applikasjonsserveren lagrer alle applikasjonene som blir brukt i systemet.

4.1.1.3 Aspektserver

Aspektserveren inneholder objekter som blir brukt i hele systemet. Den inneholder blant annet informasjon om systemets struktur, samt informasjon om utstyret det er assosiert med. I aspektserveren blir filer som datablad, brukermanualer, HMI skjermer og alarmlister lagret. Objekter blir også lagret her og kan bli åpnet av brukere av systemet.

4.1.1.4 Tilkoblingsserver

Tilkoblingsserveren er en server som gir tilgang til kontrollere og andre kilder for sanntidsdata, historiske data, hendelsesdata og alarmer. Denne står for kommunikasjonen mellom klient-/servernettverket og kontrolleren.

4.2 Maskinvare

I årets oppgave er det kun lagt til to kamera med vidvinkellinse, som kan leses mer om i kapittel 4.2.6 OpenMV H7. Utenom kameraene, er det skrevet en kort forklaring om utstyr som er montert i forbindelse med tidligere oppgaver for å skape et godt helhetsbilde.

4.2.1 ABB AC800M

Kontrolleren som er brukt på banen er en programmerbar kontroller av typen ABB AC800M. Den er i hovedsak produsert for bruk i prosessindustri og har derfor mange muligheter for bruk og tilkobling av ønskelig utstyr.

Figur 5 - Kontroller med I/O moduler

4.2.2 S800 I/O

I/O modulen som er brukt er av typen S800 I/O. Den passer for et bredt spekter av prosesskontroller fra ABB, men har også mulighet til å bruke utstyr fra andre leverandører på grunn av de gode tilkoblingsmulighetene.

Figur 6 - I/O moduler

4.2.3 ABB optisk modul buss

TB840 er en optisk modulbuss med fiberoptisk grensesnitt. Den blir brukt til å skape kommunikasjon mellom kontrolleren og lokale I/O kort. Modulen har et elektrisk og et optisk grensesnitt, der det elektriske blir brukt til kommunikasjonen med I/O kortene og den optiske brukes for kommunikasjonen med kontrolleren.

Figur 7 - Optisk modul buss

4.2.4 Arduino UNO og Mega

Arduino er en mikrokontroller som i dette prosjektet blir brukt til å sende signal mellom kameraene, magnetbryterne og PLSen. Arduino er en open-source basert maskinvare som bruker programmeringsspråket C++. Hovedforskjellen mellom UNO og Mega kortet er at det er flere inn- og utganger på Mega kortet, samt at det har en kraftigere prosessor og mer minne.

Figur 8 - Arduino UNO

4.2.5 Velleman motor- og strømmodul

Modulen er nødvendig for å drive et rele eller en stegmotor med et Arduino UNO kort. Den består av to kanaler som har output på opp til 2,5 ampere når man bruker internstrøm fra Arduinoen, eller 6 ampere dersom man bruker en ekstern strømkilde [4].

Figur 9 - Velleman motor- og strømvern

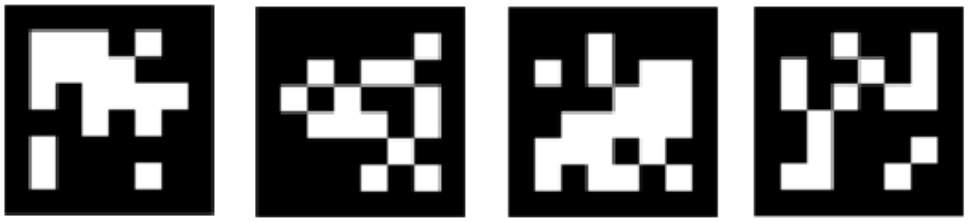
4.2.6 OpenMV H7

For å registrere tog ble det i fjor montert fire kameraer av typen OpenMV H7. Dette er et lite mikrokontrollerkort som krever lav spenning som blir programmert i MicroPython. I år er det blitt montert to ekstra kameraer for å kontrollere sporene i svingene. For å få bilde av hele svingen med et kamera, per sving, er det montert vidvinkellinser på kameraene. Disse gir en synsvinkel på 99°, og sørger derfor for god dekning i svingene.

Figur 10 - OpenMV H7

4.2.7 AprilTag

I fjorårets oppgave introduserte de AprilTags for å merke de ulike togene slik at kameraene registrerte forskjell og kunne melde fra hvilket tog som var hvor på banen. Denne type merking er mye brukt innenfor robotikk og kamera kalibrering. AprilTag biblioteket er bygd opp i C, uten ekstern avhengighet, og er lagd for å enkelt kunne inkluderes i andre applikasjoner.



Figur 11 - AprilTags

4.3 Programvare

I dette kapittelet vil det foreligge en kort beskrivelse av programvaren som er blitt brukt i forbindelse med programmeringen av komponentene på banen.

4.3.1 800xA

800xA er ABBs eget system, og det brukes til å overvåke og kontrollere et prosessanlegg. Operatøren har tilgang til informasjon via «operator workplace», som er operatørens grensesnitt til 800xA funksjonene. Her er ofte de tradisjonelle operatørfunksjonene som prosessgrafikk, alarm- og hendelseslister, trendvisninger og historikklogger tilgjengelige. I tillegg til dette tillater System 800xA operatøren å ha direkte tilgang til dokumentasjon, standard operasjonsprosedyrer, og tegninger, avhengig av hvordan prosessapplikasjonen er konfigurert [5].

4.3.2 Control Builder M

Control builder M blir brukt til å lage kontrolløsninger, biblioteker som kan brukes om igjen for AC800M kontrolleren. Den støtter alle fem programmeringsstrukturene i henhold til IEC 61131-3 [6], som består av:

- Instruksjonsliste
- Strukturert tekst
- Funksjonsblokkdiagram
- Sekvensiell funksjonsoversikt
- Stigediagram

4.3.3 Arduino IDE

Arduino IDE er programvaren som blir brukt for å styre de ulike arduino kortene. Dette er en gratis programvare som vi har fått erfaring med fra tidligere fag på studiet. Programmeringsspråket som blir brukt er C/C++ og programvaren er veldig fleksibel i forhold til andre mikrokontrollere når det kommer til operativsystem. De fleste maskinvarene er begrenset til kun Windows, men Arduino IDE kan brukes på Windows, MacIntosh OSX og Linux [7].

4.3.4 OpenMV IDE

OpenMV IDE er programvaren som brukes for å programmere kameraene. Dette er også en gratis programvare som ligger lett tilgjengelig på internett og er kompatibel med alle de mest vanlige operativsystemene som er Windows, Mac og Linux. OpenMV IDE bruker programmeringsspråket MicroPython.

4.3.5 DCC++

DCC++ står for Digital Command Control og blir brukt til å sende kommandoer til modelltog. DCC++ bruker pulsbreddemodulasjon til å sende signal til de ulike togene, som har dekodere som tolker informasjonen den mottar fra kontrollstasjonen som i dette tilfellet består av Arduino UNO kortet og Vellerman motor skjoldet. Programvaren brukt her er fra tidligere oppgaver og de har oppgitt at de har hentet denne fra GitHub og at den er skrevet av Gregg E. Berman, men at de har modifisert den med noen mindre endringer.

4.4 Kommunikasjon

Dette kapittelet vil sette søkelys på kommunikasjonsprotokollene som er blitt brukt i prosjektet for å sende informasjon mellom de ulike komponentene. De ulike protokollene ble tatt i bruk i fjorårets oppgave, slik at årets oppgave kun har videreutviklet de nødvendige delene av de ulike protokollene.

4.4.1 UART

UART er forkortelsen for Universal Asynchronous Receiver Transmitter. Dette er ikke en kommunikasjonsprotokoll, men en integrertkrets i mikrokontrollere. En UARTs hovedoppgave er å overføre og motta seriell data. Dette gjør den med å motta data fra en databuss. Siden den er asynkron så er det ingen klokkesignal som kan brukes til å si når en datapakke starter og slutter. Derfor mottar den datapakker som består av et startbit, selve dataen og til slutt et stoppbit. Når mottakerdelen mottar datapakken, konverterer den dataen. Til slutt blir datapakken fraktet videre fra UARTen til databussen for videre prosessering [8].

4.4.2 RS232

RS232 er en standard for seriell kommunikasjon mellom en DTE, en dataterminal, og en DCE, et datakretsløpterminerende utstyr for eksempel et modem. RS232 er en toveis kommunikasjon som kan sende og motta data begge veier samtidig. Dette er fordi datasendingen og datamottak er to ulike kretser. Det er også mulig å sette opp RS232 systemet enten synkront eller asynkront, men det mest vanlige er å ha den asynkron. Da fungerer den på samme måte som UART og sender et start- og et stoppbit med selve dataen imellom. Det er også mulig å ha med en valgfri paritetsbit om man ønsker dette.

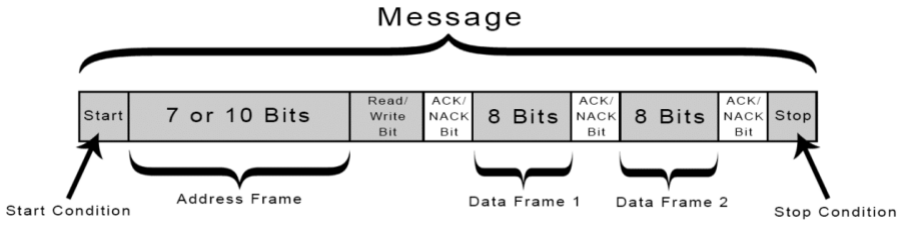
Som forklart er RS232 en toveis kommunikasjon, og i praksis betyr det at begge enhetene har TX som sender data, RX som mottar data, RTS som står for request to send og CTS som står for clear to send. I tillegg har begge enhetene jording. Enheten som ønsker å sende data sender først ut en RTS og så svarer mottakeren med en CTS dersom den er klar til å motta data, deretter er alt klart for dataoverføringen [9].

4.4.3 TTL

TTL står for Transistor – Transistor logic og består av kretser sammensatt av transistorer og motstander. Denne type kretser er til vanlig brukt i datamaskiner, instrumenter og testutstyr. TTL bruker transistorer til å utføre logiske og forsterkende operasjoner, der dette er nødvendig.

4.4.4 I²C

I²C er en kommunikasjonsprotokoll som er forkortet fra Inter-Integrated Circuit og er en seriell og synkron protokoll. I²C har mulighet til å ha både flere mastere og flere slaver. Dataen blir transportert i meldinger, som videre blir delt opp i rammer av data. Hver melding består av seks deler. Den begynner med en startilstand, deretter kommer adresserammen etterfulgt av et les-/skrivbit og et «acknowledg/no-acknowledge»-bit. Etter dette kommer selve dataen som skal overføres, før det til slutt kommer en stopptilstand [10].



Figur 12 - I2C

5 Realisering av valgt løsning

I begynnelsen brukte vi tid på å bli kjent med kameraene og mulighetene disse ga oss. Vi utforsket ideen om å benytte oss av lysdioder i enden av banen, men vi endte opp med å bruke frame differencing.

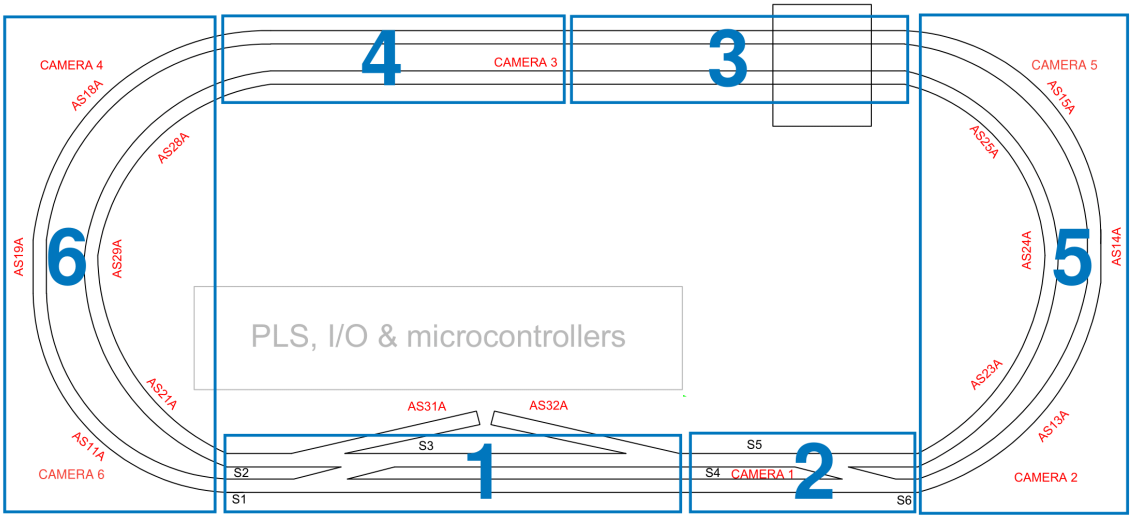
Frame differencing sammenligner et referansebilde med et «nå» - bilde. Endringer i bildet vil se ut som et invertert svart-hvitt bilde, hvor lyse områder i bildet har større endringer enn mørke områder. Ved å øke kontrasten på bildet vil skillet mellom svart/hvitt bli tydeligere. Dersom bildet ikke har endringer, vil det være helt svart. Referansebildet blir tatt ved oppstart, og over tid kan det forekomme endringer i lysforhold eller endringer i bildet generelt. For at kameraene skal kunne kompensere for dette må man ta et nytt referansebilde. Et nytt bilde vil bli tatt dersom det ikke er et tog i sonen og kameraet ser et objekt i mer enn 30 sekund. Hjørnekameraene vil derimot ta nytt referansebilde selv om det er et tog i sonen, siden de ikke «ser» AprilTag. Tiden det tar for å ta nytt referansebilde kan endres i kamerakoden, verdien på 30 sekund ble valgt da dette passet årets oppgave.

For å detektere endringene som kommer frem i frame differencing benyttes en funksjon som heter «blob detection». Denne funksjonen gjør det mulig å detektere områder i bildet som oppfyller visse krav. Et område blir definert som en «blob» og kravene man setter kan for eksempel være hvor de hvite pikslene må være, samt minimum størrelse på arealet de dekker. Dersom en blob er stor nok og er innenfor området vi har definert, vil den registrere dette som et objekt og sende informasjonen videre.

Store deler av prosessorkraften i kameraene går til frame differencing og blob detection. For å ha tilgang til referansebildet i tillegg, må dette lagres på en ekstern disk. Hvert kamera er derfor utstyrt med et mikro SD-kort, og man kan da hente ut referansebildet når det trengs.

Hvert kamera har fire punkter som tilsvarer koordinater på bildet og dersom kameraet skal oppfatte et objekt i sonen, må objektet være innenfor området av disse fire punktene. Kameraet tegner selv hvite streker mellom disse fire punktene for å gjøre det lettere å se hvor området er for brukeren. Et problem vi hadde var at kameraene kunne se ting utenfor banen, dette skapte problemer/forstyrrelser i bildet når vi beveget oss rundt banen. For å fjerne denne støykilden modifiserer vi bildet til frame differencing. Dette gjør vi med å tegne svarte streker rundt banen, noe som vil fungere som skylapper for kameraet og gjøre at det kun ser selve banen.

ABB800xA har kontroll over sonene på banen. En sone tilhører ett kamera og det er totalt 6 soner, illustrert på Figur 13. Hver sone inneholder «ObstacleData» som består av en boolsk verdi for hvert spor og en stoppeklokke som sier hvor lenge siden verdiene ble oppdatert. Dersom verdiene ikke er oppdatert på ca. 1 sekund vil all data resettes, ettersom kameraene kun sender data dersom de ser et tog og/eller objekter i togsporene. Sonene er lagret som en array og kamera-id tilsvarer indeksen til sonen. Det er dermed enkelt å utvide modellen med flere soner og kamera.



Figur 13 - Kamerasoner

5.1 Hjørnekamera

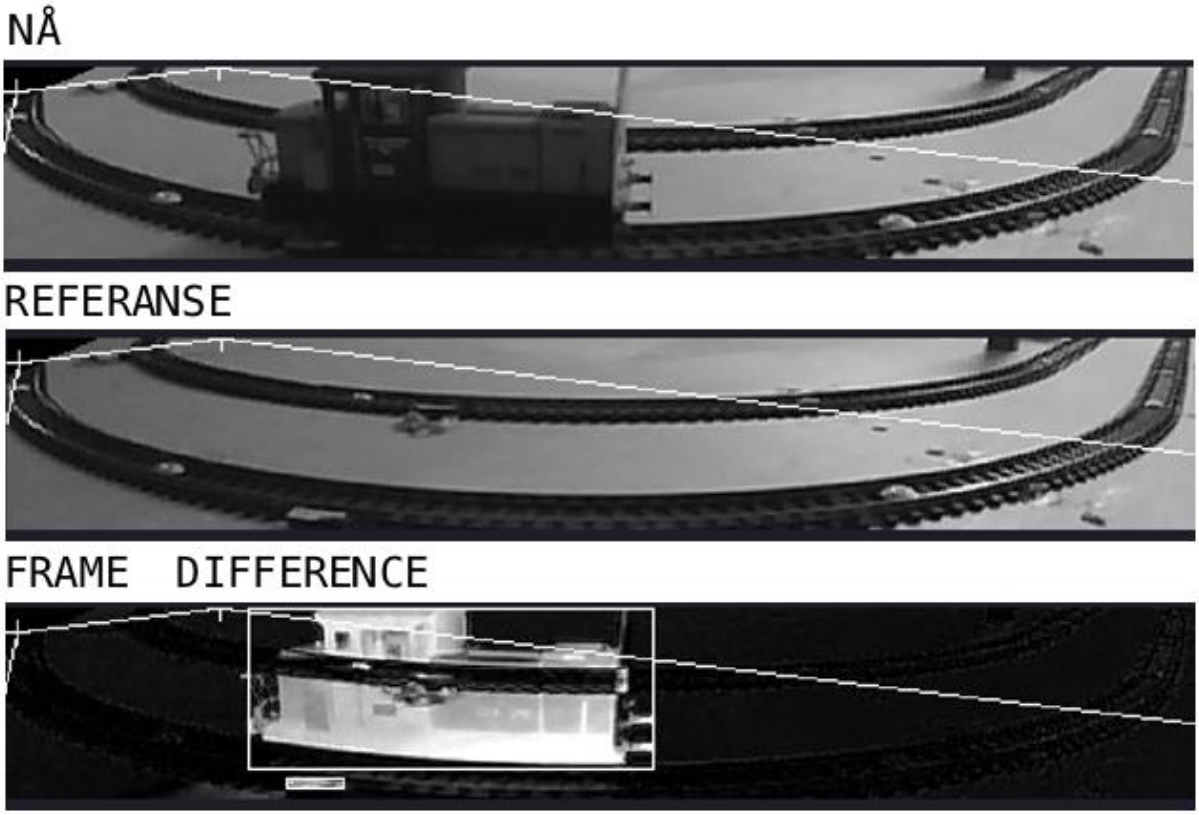
På grunn av høy oppløsning på hjørnekameraene vil de ikke klare å lese togposisjon. Dette kommer av at funksjonen kun klarer å lese AprilTags på et begrenset område.

I tillegg til dette er ikke taggene synlige for kameraene i svingene, noe som kan bli observert under «nå» bildet på Figur 14.

På Figur 14 kan man se hvordan frame differencing i sone 5 fungerer. Funksjonen kan sammenlignes med denne formelen: $NÅ - REFERANSE = FRAME\ DIFFERENCE$

De hvite strekene på bildene tilsvarer det egendefinerte området endringer i bildet må skje innenfor. Endringer utenfor disse strekene vil ikke bli markert som et objekt. Som man ser på «nå» bildet vil det innerste sporet være ute av synet dersom noe befinner seg i det ytterste sporet. Dette kan være en sikkerhetsrisiko for andre tog. For å løse dette problemet vil hjørnekameraene si at det alltid befinner seg objekter i begge sporene, selv om den kun ser ett objekt. Det gjør at det kun kan kjøre ett tog i svingene om gangen så lenge objektdeteksjon er aktivert. Som man ser på «frame difference» bildet på Figur 14 vil toget ha en hvit firkant rundt seg, dette er for å indikere for brukeren at kameraet har detektert dette som et objekt.

Ettersom modellen ikke har analog lesing av posisjon i svingene, vil ikke togene bytte sone før de kjører over en magnetbryter for den aktuelle svingen. Man er dermed nødt til å korte inn «synsfeltet» til kameraene, siden de ikke skal oppdage objekter utenfor den sin sone. Dette kan skape problemer med soneskifte for togene. På grunn av dette vil vi få dødsoner inn i og ut av svingene.

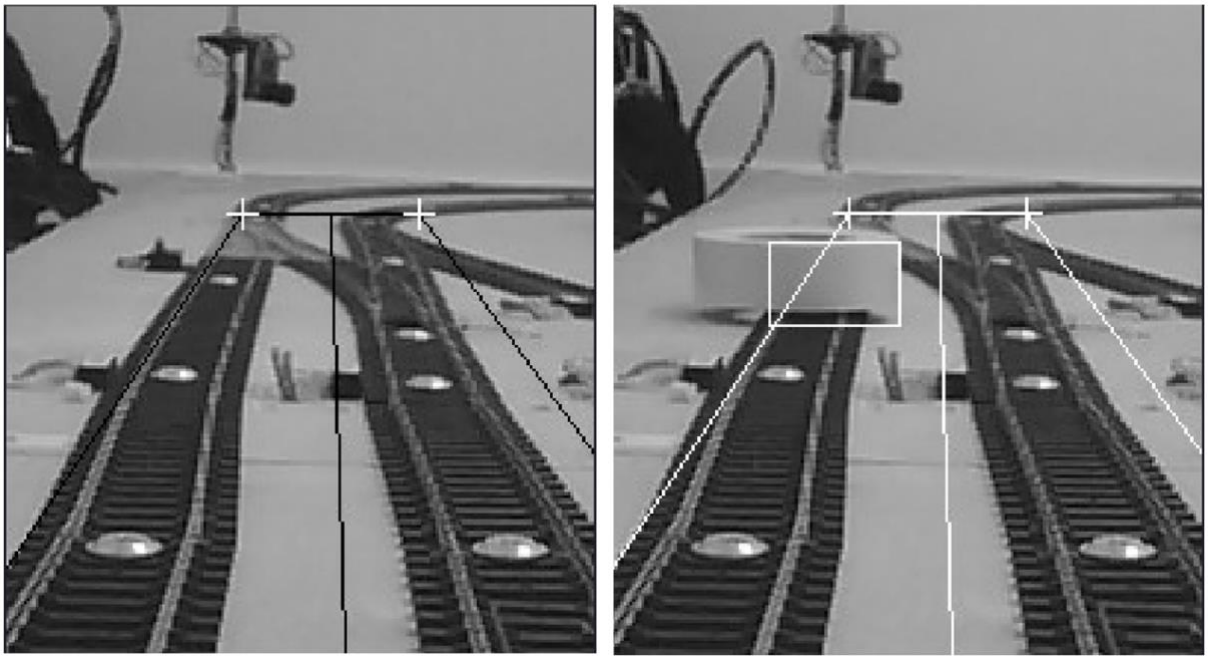


Figur 14 - Frame difference

5.2 Langsidekamera

Kameraene på langsidene er programmert litt annerledes enn for hjørnekameraene. Dette er hovedsakelig på grunn av oppløsningen på hjørnekameraene er større og tar mer prosessorkraft enn langsidene. Med mindre oppløsning på bildene kan vi behandle to versjoner av bildet samtidig, en for frame differencing og en for AprilTags, noe som gjør at man kan vise selve «nå» bildet i OpenMV IDE istedenfor «frame difference» bildet.

Kameraene på langsidene har et område objektene må være innenfor, likt som for hjørnekameraene, men den har også en ekstra funksjon som skiller mellom venstre og høyre spor. Figur 15 viser hvordan det ser ut dersom et objekt er detektert, i dette tilfelle i venstre spor. Man ser at linjene blir hvite dersom kameraet oppdager et objekt, og dette er kun en visuell funksjon for brukeren. Det er selvfølgelig mulig at et objekt dekker begge sporene, noe kameraene vil oppdage helt selv og sende informasjonen videre.



Figur 15 - Langsidekamera objekt

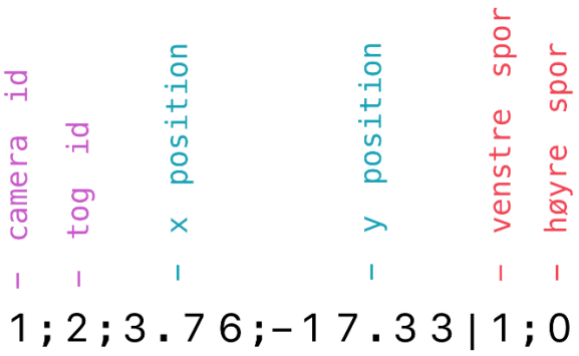
5.3 Fra kamera til Arduino

Modellen har to Arduinoer som mottar seriell data fra kameraene som er plassert på banen. Arduino UNO har én innebygd seriell port, og denne porten blir brukt til å kommunisere med ABB 800xA. Arduino Mega har fire innebygde serielle porter og de blir brukt til å kommunisere med kamera 1-4. For å kunne bruke to nye kameraer med seriell kommunikasjon ble «SoftwareSerial» benyttet. Dette er et bibliotek i Arduino IDE som kan simulere én seriell port per Arduinokort. Ettersom vi har to Arduinoer er det mulig å simulere to nye serielle porter, noe som ble brukt for kamera 5 og 6.

For å ikke sende unødvendig mye data vil kameraene kun sende seriell data dersom de ser et tog eller et objekt. Dette gjør de i form av en streng, noe som gjør det lett for PLS koden å dekode.

Figur 16 viser hvordan den serielle dataen er formatert.

Ettersom fjorårets oppgave benyttet seg av tilsvarende format, var det mulig å bygge videre på deres data. Det ble derfor valgt å skille tog-data og objekt-data, noe som ble gjort med å bruke '|' som et nytt skilletegn for objekt-dataen. Noe som ble diskutert var å kryptere dataen som sendes fra kameraene, dette hadde økt sikkerheten ettersom det ville være svært vanskelig å lese og modifisere dataen for uautorisert personell. På grunn av manglende kunnskap om kryptering av data, samt programmering i 800xA falt valget på å sende ukryptert data.



Figur 16 - Seriell data

5.4 Fra Arduino til ABB800xA

Når all seriell data er mottatt på Arduino Megaen, vil denne sende dataen videre til Arduino UNOen via I^2C bus, som deretter sender dataen til PLSen via seriell kommunikasjon.

Ved å benytte oss av fjorårets løsning for lesing av seriell data i ABB 800xA kan vi skille mellom tog- og objekt-data. Dette gjør at behandlingen av dataene skjer i to forskjellige funksjoner. Ettersom skilletegnet '|' er annerledes enn ';', kan man enkelt hoppe rett til objekt-data og behandle denne. For å hindre at årets koding kan ødelegge eller påvirke tidligere løsninger valgte vi å programmere ny uavhengig kode. Det betyr at all behandling av objekt-data skjer uavhengig av tidligere løsninger og det er kun styringen av togene som er endret.

5.5 Togstyring

For å opprettholde flyten fra tidligere oppgaver vil togene være mest mulig selvkjørende. Valget falt dermed på at togene skal stoppe og starte seg selv, så lenge objekt detektering er aktivert. Selve objektdetekteringen vil være dominant sammenlignet med tidligere sikkerhetssystem, men dette vil ha lite å si i praksis. Det var ønskelig at togene skulle bytte spor dersom de så et objekt i neste sone og sporbytte var mulig, men dette ble ikke implementert da kode fra tidligere oppgaver gjorde dette veldig komplisert.

Ved å benytte oss av togenes posisjon og neste posisjon vil vi kunne kartlegge posisjonsverdiene til sonene på banen. Det gjør at vi har togenes nåværende sone i tillegg til neste sone de skal inn i. Dersom togene leser et objekt i neste sone og i samme spor som toget er nå vil det stoppe umiddelbart. Når objektet forsvinner vil toget ikke kjøre videre før det har gått ett sekund, og denne verdien kan endres, og gjør at vi har en viss sikkerhetsmargin før togene kjører videre.

Kameraene detekterer togene som objekter, dette er gjort bevisst ettersom logikken for styringen av tog ligger i PLSen. Ved å gjøre det på denne måten vil ikke to tog være i samme spor og sone samtidig, noe som skaper en ekstra sikkerhet på banen. Togene ignorerer objekter som er i samme sone for å unngå at de stopper med å «se» seg selv.

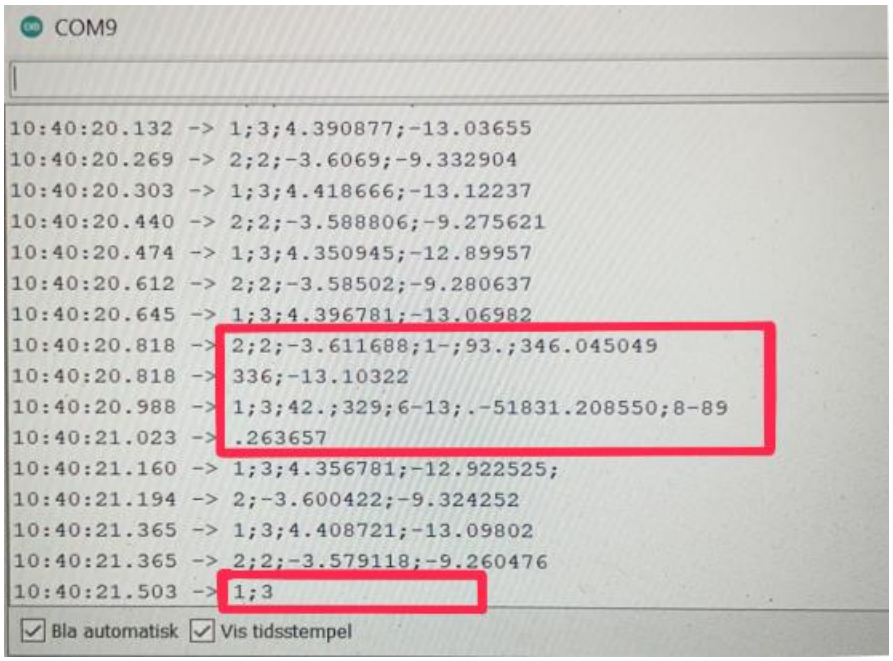
6 Testing

Ettersom oppgaven bygger på tidligere bacheloroppgaver, var det svært viktig å få en oversikt over hvordan jernbanemodellen opererer og om alt fungerte som det skulle. Testingen ble dermed delt opp i to deler: oppstartstesting og testing av ferdigstilt oppgave.

6.1 Oppstartstesting

Etter hver gang modellen har vært demontert eller avslått må det gjennomføres ulike kamerakalibreringer. Disse kalibreringsrutinene ble lagd av fjorårets gruppe. Etter montering av modellen, startet vi X- og Y-kalibrering via HMI-bildet. Kalibreringen er relativt enkel og godt dokumentert i vedlegget «Camera callibration» i Appendiks C.1 , som gruppen i fjor lagde. Tross dette, støtte vi på problemer.

I X-kalibreringen blir det registrert to togposisjoner på hvert kamera. Hver registering tok flere minutter og innimellom fikk vi inn ugyldige registeringer. Vi mistenkte at noe ikke stemte. På grunn av den lange ventetiden på hver registering sjekket vi timer funksjonen og oppdaget at det var brukt sekunder istedenfor millisekunder, noe som førte til at kameraene kun oppdaterte seg hvert 100 sekund. Deretter koblet vi oss på UNOen og overvåket de ulike dataene som ble sendt og mottatt. Her oppdaget vi at dataen som vi mottok var korrupt, og dette skjedde kun når flere kamera sendte data samtidig. Formatet vi mottok fra kameraene avviket dermed fra det formatet vi ønsket, og gjorde dataen uleselig for PLSen. Ettersom feilkilden var lokalisert, rettet vi raskt opp i denne og ga både UNO- og Mega-koden en oppgradering. Kamerakalibreringen fungerte fint etter dette. Figur 17 viser hvordan den korrupte dataen så ut.



Figur 17 - Korrupt data

Da kameraene var kalibrert var det bare å sette et tog på skinnene og teste ut de ulike eksisterende funksjonene. Da vi kjørte toget forover og bakover i ulik hastighet gikk det fint, men da det ble kjørt flere tog samtidig la vi merke til at de holdt ulik hastighet, selv om de var satt til å kjøre like fort av kontrolleren. På bakgrunn av dette kan det ikke brukes estimering av posisjon for posisjonsmåling av

tog i svingene. Sikkerhetssystemene sporbytte, antikollisjon og fartsreduksjon så ut til å fungere i de situasjonene vi klarte å simulere. HMI-bildet oppdaterte seg stort sett fint med de nye posisjonene til togene. Det eneste problemet vi fant var hvis man fysisk fjerner toget fra sporet, vil posisjonsvariabelen forbli lagret som siste registrerte posisjon til toget i PLSen. Dette førte til at toget, som ble fjernet fra banen, oppfattes at det fortsatt står i sist registrerte posisjon. Dette skapte problemer da vi skrudde på sikkerhetssystemet antikollisjon. De gjenstående togene endte opp med å stoppe for toget som var fjernet.

Videre begynte vi å studere kamerakoden fra fjoråret og prøve oss på enkle eksempeloppgaver med frame differencing og blob detection. Her opparbeidet vi raskt en god forståelse for programmeringsspråket og de ulike funksjonene som var brukt tidligere, og i tillegg til de som skulle brukes videre. Etter to nye kamera med vidvinkellinse ble kjøpt inn og montert, kunne tester relatert til problemstilling nummer to kjøres. Den viktigste testen var å kjøre tog, i begge spor, gjennom svingen og observere i hvilke deler av svingen kameraene klarte å lese AprilTag. Med denne testen fikk vi avklart flere ting. Det første var, som mistenkt, at det ikke er mulig å lese AprilTags med den oppløsningen som trengs for å oppdage hindringer i skinnene. Det andre var at det var mulig å se AprilTags i større deler av svingen enn først antatt, og at det ikke var kritisk at AprilTag ble lest av vinkelrett på. Selv om den testen kan sees på som delvis vellykket, så fikk vi kun posisjonsmålinger fra rundt en tredjedel av svingen. Dette tilsvarer et område som er mindre enn det som bruker magnetbrytere til å registrere posisjon.

6.2 Testing av ferdigstiltoppgave

Etter hvert som vi var ferdig med ulike deler av kodene ble den testet sammen med den opprinnelige koden og sett hvordan den fungerer på banen. På denne måten har vi funnet flere feil og fått utbedret disse.

Etter vi hadde programmert kamerakoden og testet den så godt som mulig i programvaren, så koblet vi oss på det ene kameraet og plasserte et tog i sporet uten å kjøre selve banen. Det ble oppdaget at blobdeteksjonen var for følsom og dette førte til at det ble registrert blobs utenfor banen som var store nok til at de påvirket systemet. I tillegg ble lysrefleksjoner på skinnene registrert som blobs. Dette ordnet vi med å legge inn spesialdesignet «skylapper» til hvert kamera, slik at de ikke så ting utenfor sine soner. Vi justerte også følsomheten for lysendringer slik at koden ble mer tolerant for refleksjoner. Skylappene består av egendefinerte svarte firkanter som markerer ut området vi ikke ønsker at kameraene skal registrere endringer i. Dette løste problemet med at personer som stod rundt banen ble registrert som blobs og det ble mulig å stå helt inntil banen for å følge med på togene uten å bli registrert som en hindring.

Når kamerakodene fungerte som ønsket, lagde vi en enkel kode i PLS programmet for å sjekke at vi fikk inn riktig data fra Arduinoen. For å kontrollere at data som arduinoen registrerte var den samme PLSen registrerte, koblet vi oss til arduino UNO kortet samtidig som vi var koblet til PLSen og leste av de ulike datastrengene. Da ble det oppdaget at kameraene av og til registrerte «spøkelsestag». Disse var ofte mye høyere tall enn 1,2 og 3, som er tagene til togene og derfor de vi forventet å få opp. Dette har vi funnet ut at er fordi kameraet ser former som kan ligne på en AprilTag, og da sender videre at det «ser» en tag. Da de fleste spøkelsestagene vi fikk opp var høyere enn 10 så har vi fikset dette

problemet med å sette at tagene som skal registreres kan være maks ha verdi 10, men dette tallet er enkelt å endre dersom det skulle være ønskelig.

Da vi utvidet PLS programmet til å inkludere vår data fra kameraene la vi merke til at HMIen registrerte de ulike togene i motsatt spor enn det de var i. Da dette kun skjedde på langsidene så konkluderte vi raskt med at det måtte være på grunn av kamerakoden. Etter litt husket vi at vi tidligere hadde snudd bildet til kamerakoden for at de skulle være rett vei selv om de er montert opp ned. Dette resulterte da i at X-verdien ble -X, og PLSen trodde toget var i motsatt spor enn det faktisk var. Dette forbedret vi med å snu X-verdien, slik at den nå er tilbake til å være korrekt med tanke på det PLSen er programmert for.

En av egenskapene banen hadde fra før var at togene kunne kjøre både fremover, i kjøreretning, og rygge, mot kjøreretning. Vi la merke til at dersom et tog rygger inn i en sone vil kameraet registrere det som en blob siden kameraet vil se bakdelen av toget før den ser en AprilTag. Dette skapte problemer for sikkerhetssystemet «objektdetektering». Sikkerhetssystemet ville stoppe toget siden det var på vei inn i en sone som inneholder et objekt i sporet, selv om objektet var toget selv. Togene kan fremdeles rygge, men da må sikkerhetssystemet for hindringer i sporet være av.

7 Diskusjon

I løpet av oppgaven har det tatt mange avgjørelser, i dette kapittelet kan man lese om intern drøfting og tankegangen bak disse.

7.1 Oppdage hindringer i spor

En stor del av oppgaven har bestått av å programmere kameraene. Noe som ble brukt en del tid på var å finjustere hvor sensitivt kameraene skulle være til lysendring og hvilken lysstyrke og størrelse en blob skulle ha. Det var flere valg som kunne påvirke hvor sensitivt kameraet skulle være til lysendringer. Når et «nå»-bilde blir tatt og brukt frame differencing på, kan man justere på blant annet kontrast og lysstyrke. Siden dette vil variere, og ikke er en «fasit», ble det brukt mye tid på tilpassingen av disse elementene for å få minst mulig utslag på skygger og lysendringer, men fortsatt oppdage alle hindringer. Ved å kombinere dette med lysstyrke og minimum størrelse på en blob, klarte vi å oppdage hindringer i sporet.

Et annet aspekt av kameraprogrammeringen var hvor ofte det skal tas et nytt referansebilde. Som nevnt i kapittel 5, er ventetiden for å ta et nytt bilde satt til 30 sekunder, med forbehold om at kameraet ikke ser en AprilTag. Hvis ventetiden var satt høyere kunne lysforskjellen mellom referansebildet og «nå»-bildet bli stor nok til at kameraet oppfattet den som en blob, og ville dermed stoppet toget. Derimot om den var satt lavere risikeres det at midlertidige endringer hadde blitt tatt med i det nye referansebildet.

For å videreformidle hindringer i sporet ble det diskutert hvilke data man skulle sende og hvordan de skulle sendes. Her var det mye å ta hensyn til siden dataene som blir sent fra kameraene blir brukt flere steder. Det ble vurdert å ha to ulike format på dataen som blir sendt fra kamera, ett format for å oppdage hindringer, og det eksisterende formatet for å måle posisjonen til togene. Dette ville skape flere utfordringer i videreføring av data, siden formatet er hardkodet der dataen skal tolkes. Det ble dermed bestemt at det skal brukes formatet vist i Figur 18.

Figur 18 - Seriell data for hindring i sporet

Da kameraene ble bestilt ble det vurdert om det var nødvendig med ett Arduinokort til, på bakgrunn av for få ledige serielle innganger for kommunikasjon med de to nye kameraene. Etter undersøking ble det oppdaget at Arduinokortene har et bibliotek «SoftwareSerial» i programvaren. Da dette ble testet første gang, var det kun ett av de nye kameraene som sendte data. Etter grundigere undersøking kom det fram at det kun går an å simulere én seriellport per kort. Ettersom det allerede var montert to Arduinokort ble ikke dette et problem.

Gjennom prosjektet har fremdriften stort sett vært etter planen. Det som skapte den største forsinkelsen var programmeringen i 800xA. Det var lagt inn god tid til å lære seg programvaren, men grunnet en misforståelse om hvordan kursingen i 800xA skulle foregå, krevde det mer selvstudie enn forventet. Dette i kombinasjon med utfordringer knyttet til eksisterende kode, førte til forsinkelser. Under tidsberegningen burde det blitt tatt hensyn til at koden i 800xA er skrevet av flere studentgrupper med ulike kunnskaper innenfor programmering. Et resultat av at ulike grupper har videreutviklet samme kode er at den fremstår mindre allsidig. På bakgrunn av dette bestemte vi å lage ny kode utenom den eksisterende, i stedet for å bygge direkte på den.

Noe som ble oppdaget under testing av ferdigstiltoppgave, var at hvis et tog kommer for nært et kamera og togets AprilTag ikke kan leses av gitt kamera, vil toget i et kort øyeblikk oppfattes som en hindring. Dette vil resultere i at eventuelle tog i kamerasonen bak vil stoppe opp maks et par sekunder før det vil kjøre videre. En mulig løsning på dette problemet kunne vært å legge inn en timer i koden, slik at når et tog kjører fra én kamerasone til en annen, vil det være en liten forsinkelse på å oppdage hindringer i første kamerasone. Grunnen til at dette ikke ble gjennomført var tidsmangel og ble dermed nedprioritert da det ikke påvirker sikkerheten på en negativ måte.

7.2 Posisjonsmåling i sving

Som avdekket i kapittel 6.1 viste det seg å ikke være mulig å gjennomføre noen av løsningsforslagene presentert i kapittel 3.2.2. Dersom vi ikke skulle oppdage hindringer i sporet, ville det vært mulig å gjennomføre Løsningsforslag 1 – Vidvinkel kameralinse, da man kunne redusert oppløsningen til det nivået som kreves for å lese AprilTags. På grunn av få AprilTag-lesninger gjennom svingen, vil man ende opp med bedre og mer nøyaktig posisjonsmåling med magnetbryterne. På bakgrunn av dette og problemene knyttet til oppløsningen var ikke løsningsforslaget mulig å gjennomføre.

Under oppstartstestene kom det frem at togene holde ulik fysisk fart, selv om de er satt til å holde lik fart av kontrolleren. Dette fører til at Løsningsforslag 2 – Estimering av posisjon heller ikke ville være aktuell å gjennomføre.

7.3 Kommunikasjonssvikt med tog

I kapittel 1.3 defineres kommunikasjonssvikt med tog som at toget ikke kan styres. I en slik situasjon kan følgende skje: manglende kontakt med et tog og det står i ro, manglende kontakt med et tog og det kjører, eller at toget er fysisk fjernet fra skinnene.

Hvis man mister kontakt med et tog på en langside og det står i ro, vil et av kameraene se toget og dens AprilTag, og dermed ikke ta et nytt referansebilde. Dette er positivt. Mister man derimot kontakt med et stoppet tog i en sving, vil det bli lagret et nytt referansebilde etter 30 sekunder. Dette medfører at etter 30 sekunder vil det stoppede toget, som til nå har blitt sett på som en hindring, blir en del av referansebildet og ikke sees på som en hindring lengre. Som nevnt i kapittel 3.1.3 har hvert tog variabelen «posisjon» lagret i kontrolleren. Det vil si at selv om det stoppede toget ikke blir sett på som en hindring, er det fortsatt registrert i kontrolleren at det befinner seg i gitt posisjon. Så lenge toget stoppet i den posisjonen som er lagret, vil antikollisjonssystemet hindre kollisjon med andre tog. Antikollisjonssystemet vil gjøre det samme hvis man mister kontakt med et tog som er i bevegelse.

Hvis man plukker opp et tog fra banen vil PLSen fortsatt ha registrert sist kjente posisjon i variabelen «posisjon». Dette medfører at antikollisjonssikkerhetssystemet vil stoppe tog som nærmer seg sist kjente posisjon til toget som ble fjernet.

7.4 Annet

Heller ikke årets oppgave kom klar av koronasituasjonen vi befinner oss i, noe som gikk ut over fremdriftsplanen i starten av oppgaven. På forhånd var gruppen klar over at vi måtte ha jernbanen hjemme hos oss da det ikke var mulig garantere at testrommet til ABB ville være åpent hele våren. Leveringen av banen skapte også noen problemer da det måtte passe både for studentgruppen og for veileder fra ABB å levere banen. Samtidig kunne ingen ha koronasymptom da ABB har svært strenge smittevernregler på dette. I det store bildet var ikke dette noe som påvirket fremdriften betraktelig, men noen mindre forsinkelser oppstod.

Foruten om en mindre misforståelse har også kommunikasjonen mellom de ulike partene involvert i oppgaven gått veldig bra. Misforståelsen oppstod tidlig i prosjektet og gikk ut på at studentgruppen trodde de skulle få et kurs i 800xA, men at kurset egentlig bestod av egenstudier av en kursmanual. Dette ble oppklart relativt raskt og førte heller ikke til noen større forsinkelse med tanke på fremdriftsplanen.

8 Konklusjon

Jernbanemodellen er en bacheloroppgave som skal bygges videre på flere år fremover, med formål om å brukes av ABB som en interessevekker ved ulike arrangementer.

Et av kravene ABB stiller for å få arbeide på modellen er fullført FSE-kurs, som vi fikk tilbud om å gjennomføre som nettkurs på grunn av covid-19. En annen vesentlig konsekvens av den globale pandemien covid-19 var at vi ikke fikk tilgang til ABB sitt testrom og måtte ha jernbanemodellen hjemme hos en av oss. Dette førte til at det var mer krevende å organisere assistanse og overlevering av modellen, og nytt utstyr krevde mer planlegging.

De fysiske endringene på modellen består av to nye kamera i hvert sitt hjørne. Disse ble montert for å løse problemstilling 1, oppdage hindringer i spor. Utgangspunktet for oppgaven gitt av ABB var å introdusere dynamiske elementer. Gruppen deltok i utformingen av en mer konkret problemstilling: å oppdage hindringer i sporet med et krav om å unngå kollisjon minst ni av ti ganger. Dette ble løst ved å sammenligne et referansebilde med et «nå»-bilde og å lete etter endringer. Løsningen vår oppfyller kravet og vil oppdage hindringer både på langsidene og i svingene.

Det kom frem under testingen at begge løsningsforslagene laget for problemstilling 2, posisjonsmåling i sving, ikke kunne brukes. Det ble dermed ikke prioritert å bruke noe mer tid på dette.

Den tredje problemstillingen ble presentert for oss før vi mottok jernbanemodellen. Vi hadde dermed ikke kunnskap om hvordan modellen ville handtere kommunikasjonssvikt med tog. Etter oppstartstesting ble det synlig at de etablerte sikkerhetssystemene ville hindre kollisjon, selv uten kontakt med tog.

Gruppen har tidligere arbeidet mye sammen på andre prosjekter og kjenner dermed hverandre sine sterke og svake sider. På bakgrunn av dette har samarbeidet gått fint og vi har hatt god kommunikasjon. Ett av gruppemedlemmene jobber offshore ved siden av studiet og har hatt et par turer i løpet av våren. Dette har ikke vært et problem da det ble tatt hensyn til under planleggingen.

Prosjektet har vært svært lærerik og vi har brukt mye av kunnskapen vi har tilegnet oss i løpet av de siste tre årene. I tillegg har oppgaven i stor grad omhandlet kameraprogrammering, som var nytt for hele gruppen, der vi har opparbeidet oss en god forståelse.

Kort fortalt – bacheloroppgaven har gått på skinner.

9 Bibliografi

[1] ABB, «ABB,» U.Å. [Internett]. Available: https://new.abb.com/no/om-oss. [Funnet 25 Januar 2021].

[2] M. R. Olsen og M. Vikane, «Railway Safety,» HVL, Bergen, 2019.

[3] T. S. Koløen, E. Ligård og J. Ness, «Railway Safety Upgrade,» HVL, Bergen, 2020.

[4] Velleman, «velleman.eu,» U.Å. [Internett]. Available: https://www.velleman.eu/products/view/?id=412538. [Funnet 5 Mai 2021].

[5] ABB, Juli 2015. [Internett]. Available: https://library.e.abb.com/public/9b0e967a68aa48f6bcc2ff88ff64cffb/3BSE036904-510_F_en_System_800xA_Operations_5.1.pdf. [Funnet Mai 2021].

[6] ABB, «abb.com,» U.Å. [Internett]. Available: https://new.abb.com/control-systems/system-800xa/800xa-dcs/hardware-controllers-io/control-builder-engineering-software. [Funnet Mai 2021].

[7] Arduino, «Arduino.cc,» 5 Februar 2018. [Internett]. Available: https://www.arduino.cc/en/Guide/Introduction. [Funnet 20 Mai 2021].

[8] S. Campbell, «Circuit basics,» U.Å. [Internett]. Available: https://www.circuitbasics.com/basics-uart-communication/#:~:text=UART%20stands%20for%20Universal%20Asynchronous,transmit%20and%20receive%20serial%20data.. [Funnet 21 Mai 2021].

[9] Wikipedia, «Wikipedia,» 26 April 2019. [Internett]. Available: https://no.wikipedia.org/wiki/RS-232. [Funnet 21 Mai 2021].

[10] S. Campbell, «Circuit Basics,» U.Å. [Internett]. Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/#:~:text=I2C%20is%20a%20serial%20communication,always%20controlled%20by%20the%20master.. [Funnet 21 Mai 2021].

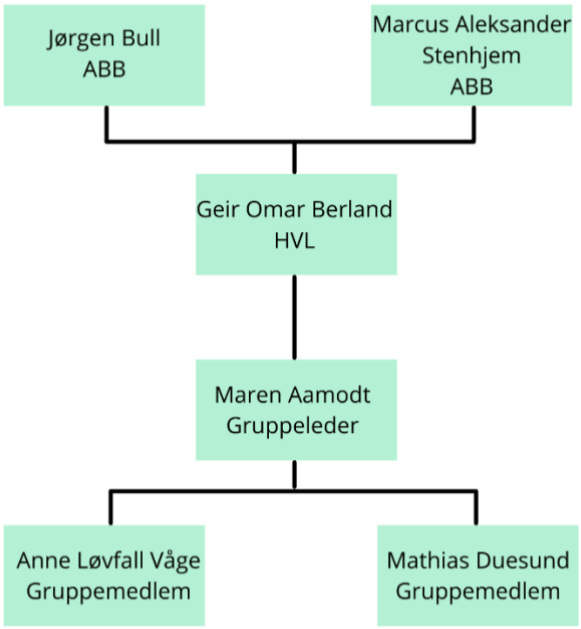
Appendiks A Forkortelser og ordforklaringer

Forkortelse	Forklaring
800xA	ABB sin programvare
AC800M	En maskinvare som inneholder kommunikasjon interface prosessor og annet utstyr som kan konfigureres via en kontroller
AC800M kontroller	En kontroller i AC8000M serien
FSE	Forskrift om sikkerhet ved elektrisk anlegg
HMI	Human machine interface
HVL	Høgskulen på Vestlandet
HW	Hardware
I/O	Input/output moduler koblet til kontroller
PLS	Programmerbar logisk styring
SW	Software
QR	Quick Response

Tabell 1 - Forkortelser

Appendiks B Prosjektledelse og styring

B.1 Prosjektorganisasjon



Figur 19 - Prosjektorganisasjon

B.2 Prosjektform

Prosjektformen vår er egendefinert og tar utgangspunkt i disse fem trinnene.

1. Oppnå forståelse for hvordan modellen fungerer
2. Planlegging og utarbeide løsningsforslag
3. Utvikle og programmer vår løsning
4. Implementere løsning i modellen
5. Teste og fikse eventuelle feil

B.3 Fremdriftsplan

Se vedlagt GANTT-diagram i Appendiks C.3 *BO21E-25 Gantt v1.0.xlsx*

B.4 Risikoliste

Nivå	Beskrivelse	Handling
1	Lav risiko. Sannsynligheten for ulykke med konsekvenser er lav.	Valgfritt
2	Middels risiko. Sannsynligheten for ulykke med konsekvenser er til stede.	Anbefalt
3	Høy risiko. Sannsynligheten for ulykke med konsekvenser er høy.	Obligatorisk

Tabell 2 - Risikonivå

Nummer	Beskrivelse	Nivå	Handling
1	Kortslutning og skade på maskinvare. Noen komponenter er sensitive til høy spenning og strøm.	**2**	Før utføring, rådfør deg med andre gruppemedlemmer og les dokumentasjonen.
2	Skade på togene.	**1**	Plasser togene på en sikker plass.
3	Strømgjennomgang i menneske. Mesteparten av maskinvaren har lav spenning (under 50V), men modellen er tilføringsstrøm på 230V.	**3**	Fjerne strømtilførsel før kobling. Kun FSE-godkjent personell har lov å gjøre endringer.
4	Underveis vil det bli brukt ulike verktøy. Dette kan føre til skader.	**1**	Bruke verneutstyr og førstehjelpsutstyr tilgjengelig.
5	Skjøre komponenter, som f.eks. kamera, kan lett bli ødelagt.	**1**	Behandle de skjøre komponentene forsiktig.

Tabell 3 – Risikoanalyse

Appendiks C Brukerdokumentasjon

C.1 Brukerdokumentasjon

- Blokkdiagram
    - *BO21E-25 Obstacle_Detection.pdf*
    - *BO21E-25 Obstacle_Train_Control.pdf*
- *BO21E-25 Kamerasoner.pdf*
- *BO21E-25 Soner for posisjonsmåling.pdf*
- *BO21E-25 Serial Communication.pdf*
- *Tilkoblingsliste.xlsx*

Dokumentasjon fra tidligere prosjekt

- Blokkdiagram
    - *BO20E-05 Next position monitoring.pdf*
    - *BO20E-05 Anti Collition.pdf*
    - *BO20E-05 Application logic diagram.pdf*
    - *BO20E-05 Speed limiting system.pdf*
    - *BO20E-05 Camera logic diagram.pdf*
    - *BO20E-05 Crossing system.pdf*
    - *BO20E-05 Track switching safety system.pdf*
    - *BO20E-05 Position monitoring.pdf*
    - *BO20E-05 Train block.pdf*
- *BO20E-05 Power drawing.pdf*

C.2 Oppstartsprotokoll

- *BO19E-01 Assembly and disassembly.pdf*
- *BO20E-05 Camera calibration.pdf*
- *BO21E-25 Kameraoppsett.pdf*

C.3 Utgifter og GANTT

- *BO21E-25 Gantt v1.0.xlsx*
- *BO21E-25 Budsjett.xlsx*

Appendiks D Programkode

Ferdig kode er lagt som vedlegg.

D.1 Arduino

- *BO21E-25 Arduino_MEGA_Communication.ino*
- *BO21E-25 Arduino_UNO_Communication.ino*

Kode fra tidligere prosjekt

- *Arduino UNO DCC_Vellman_shield*

D.2 OpenMV

- *BO21E-25 Cam_1.py*
- *BO21E-25 Cam_2.py*
- *BO21E-25 Cam_3.py*
- *BO21E-25 Cam_4.py*
- *BO21E-25 Cam_5_corner_detection.py*
- *BO21E-25 Cam_6_corner_detection.py*
- *AprilTag*

D.3 800xA

- *BO21E-25 Func_PositionToZone.pdf*
- *BO21E-25 Func_ReadCameraObstacleData.pdf*
- *BO21E-25 Func_ReadCameraID.pdf*
- *BO21E-25 Obstacle_Detection.pdf*
- *BO21E-25 Obstacle_Train_Control.pdf*
- *BO21E-25 Train_Control.pdf*
-

Kode fra tidligere prosjekt

- *Anti-Collision-Code-pdf*
- *Camera-Calibration-Code.pdf*
- *Camera-Positioning-Code.pdf*
- *Crossing-and-lights-code part1.pdf*
- *Crossing-and-lights-code part2.pdf*
- *Next-Position-Code.pdf*
- *Position-Monitoring-Code.pdf*
- *Speed-Limiting-Code.pdf*
- *Track-Switch-Safety-Code.pdf*