Høgskulen på Vestlandet

BACHELOROPPGAVE:

BO24EH-01

Utvikle testmodul for signalføring

Martin Karlsen (Kandidatnummer: 177)
William Waage Haugen (Kandidatnummer: 179)
Kim Jone Undheim (Kandidatnummer: 185)

21. mai. 2024

Dokumentkontroll

Rapportens tittel:	*Dato/Versjon*
BO24EH-01 Utvikle testmodul for signalføring	21. mai. 2024/1.0
	Rapportnummer: B024EH-01
Forfatter(e): Martin Karlsen William Waage Haugen Kim Jone Undheim	*Studieretning:* AUTH_2021_HØST
	Antall sider m/vedlegg 93
Høgskolens veileder: Mahdi Ghane	*Gradering:* Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: AutoStore	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): Abiyu Baqala Tashoma *abiyu.baqala.tashoma@autostoresystem.com*	

Revisjon	**Dato**	**Status**	**Utført av**
0.11	02.02.24	Første utkast	Kim Jone Undheim
0.12	08.02.24	Lagt inn Innledning	Kim Jone Undheim
0.13	14.02.24	Endring av Innledning	Martin Karlsen
0.14	15.02.24	Lagt inn Kravspesifikasjon	Kim Jone Undheim
0.15	19.02.24	Lagt inn Modul Løsning 1 og 2	Martin Karlsen
0.20	17.03.24	Lagt inn Komponenter	William Waage Haugen
0.21	25.03.24	Lagt inn Tidlig testing	Kim Jone Undheim
0.30	30.04.24	Lagt inn Valg av Løsninger	Martin Karlsen
0.40	06.05.24	Lagt inn Design av fysisk produkt	Martin Karlsen
0.50	08.05.24	Lagt inn kodestandard, Forord og Retting av skrivefeil	William Waage Haugen
0.60	10.05.24	Lagt inn Appendiks B og omskriving av Valg av løsninger	William Waage Haugen

0.70	11.05.24	Skrevet om og rettet i kapittel 2,4 og 5	Martin Karlsen
0.80	13.05.24	Lagt inn Gjenstående arbeid. Fjernet alle notater i teksten.	Kim Jone Undheim
0.81	14.05.24	Lagt inn Kjente feil Oppdatering av Analyse av problemet	Kim Jone Undheim
0.82	15.05.24	Lagt inn Fysisk testing	William Waage Haugen
0.83	16.05.24	Diverse feil retting, revisjon og oppdatering av Realisering av løsning	Martin Karlsen
0.84	18.05.24	Lagt inn Appendiks C, Sammendrag og oppdateringer i alle kapittel	William Waage Haugen
0.85	19.05.24	Lagt inn Diskusjon og konklusjon Gått igjennom kapittel 1-5 Oppdatert Sammendrag	Kim Jone Undheim
0.90	20.05.24	Endelig versjon for korrektur	William Waage Haugen
0.91	20.05.24	Fullført figurer og referanser	William Waage Haugen
1.0	21.05.24	Endelig versjon for innlevering	William Waage Haugen

Forord

Denne bachelorrapporten er skrevet som den avsluttende oppgaven for vår studie i Automatisering med Robotikk ved Høyskolen på Vestlandet i Haugesund. Bacheloroppgaven er skrevet for AutoStore AS og målet er at de kan benytte arbeidet til å effektivisere testing av egne produkter.

Vi har med skrekk blandet fryd sett frem imot å avslutte studiet ved å skrive en bacheloroppgave. Oppgaven har ligget som en mørk sky i horisonten gjennom studiet, men når vi startet diskusjoner før jul viste det seg at dette kunne bli spennende og interessant. Prosessen har utfordret oss til å finne løsninger innenfor emner vi ikke hadde god kunnskap om.

Vi vil takke Høgskulen på Vestlandet i Haugesund for tilgjengelighet av elektrolabben for gjennomføring av testing.

Vi vil takke Madhi Ghane, vår veileder ved Høyskolen på Vestlandet for å ha fulgt oss opp utover semesteret og vert tilgjengelig hvis vi trengte det.

Vi takker også Abiyu Baqala Tashoma og Vegard Brekke Løvvig hos AutoStore for sin tilgjengelighet og imøtekommenhet i forbindelse med de utfordringene vi har møtt.

Sammendrag

Denne rapporten beskriver en bacheloroppgave som er skrevet for AutoStore, hvor problemstillingen er å produsere en testmodul til bruk på firmaet sine produkter. Dette verktøyet skal kunne brukes internt av test og validerings avdelingen for å kunne logge, feil søke og generere feil på produktene for å forebygge og utbedre produktfeil. Dette verktøyet skal også ha et grensesnitt for enkel konfigurering, der operatør kan enkelt for å sette opp verktøyet. Se kapittel [3] om kravspesifikasjon for grundigere oversikt.

Resultatet ble ikke som planlagt i startfasen hvor det skulle benyttes til flere produkter. Verktøyet ble nedskalert fra å kunne brukes på flere produkt til bare ett produkt som er R5 PRO roboten. R5 roboten genererer signaler med 4 enkodere for lesing av hastighet og retning, 4 tracksensorer for lesing av posisjon, 2 temperatur sensorer for lesing av temperatur på batteripolene, 2 temperatur sensorer for lesing av temperatur på batteriladekontakten, 1 heis, 1 brems og 1 nødstopp for lesing om den er aktiv eller inaktiv. Her ble det også en del utfordringer med tanke på signalbehandling, kode, fysisk enhet og tid tilgjengelig for å løse oppgaven. Se kapittel [5] for Realisering av løsning for mer informasjon.

Løsningen på verktøyet var planlagt frem til «minimum viable product». Det ble ikke en fullstendig komplett enhet og verktøyet er på «proof of concept» stadiet, der den er delt opp i flere moduler/deler for hvert signal. Grunnen er at det har vært en del endringer og usikkerheter underveis som har gjort at tiden for å fullføre et komplett produkt ikke var mulig.

De forskjellige modulene/delene ble testet og noen av dem ble fysisk koblet på roboten for verifisering og faktisk fysisk testing. Det ble funnet flere begrensninger på utførelsen av oppgaven som generering av analogt signal for temperatur sensorene, lesing og generering av korrekt tracksensor signal, fullføring av implementering av enkoder i hovedprogram, enkelt grensesnitt for konfigurering og lesing på verktøyet. Mer om disse problemene kan leses om i kapittel [6] om Testing, i kapittel [7] Diskusjon og i kapittel [8] Konklusjon.

1 Innhold

Figurliste

2 Innledning

Til bachelor oppgaven var det gjort dialog med ulike bedrifter, der valget ble å skrive for AutoStore. Grunnen til at oppgaven til AutoStore appellerte var at muligheten til å kunne utføre både praktisk og teoretisk arbeid. Programmere en mikrokontroller til testing, simulering og logging av signalerer som kan brukes til intern testing og evaluering av AutoStore produkter. Oppgaven er krevende, men planleggingen som er gjort gjør at de ønskede funksjonalitetene kan utvikles i ulike faser.

2.1 Oppdragsgiver

Ingvar Hognaland grunnla AutoStore i 1996 [1], som en løsning på lagerkapasitetsproblemene Hatteland Group opplevde ved salg og distribuering av elektrokomponenter i Åmsosen, Nedre Vats. Historien sier at Ingvar stilte seg selv og andre spørsmålet om hva som tar mest plass på et lager. Mange forslag ble nevnt, men ifølge Ingvar var det luften som opptok mest plass, og det var her ideen til AutoStore ble født.

I et tradisjonelt lager blir alt plassert i reoler som er designet for at lagerpersonell skal kunne gå mellom dem for å hente og plassere varer. Gangene mellom og høyden på reolene begrenser hvor mye areal som kan utnyttes til lagerformål. AutoStore sin løsning for å maksimere arealutnyttelsen var å eliminere reolene og lagre alt i standardiserte kasser som stables i høyden i et rutenett. På toppen av rutenettet beveger AutoStore sine egenutviklede roboter i en x-y rutenett for å hente og levere kasser som er stablet oppå hverandre. Når en kunde ønsker en vare fra systemet, henter roboten kassen og leverer den til en utleveringsstasjon. Ved å fjerne reolgangene og utnytte høyden oppnår AutoStore sitt system en arealbesparelse på fire ganger et tradisjonelt lager, samtidig som inn- og utlevering av varer effektiviseres [2].

AutoStore har hatt stor suksess med å løse sitt opprinnelige lagerplassproblem. I dag har de levert over 1250 lagersystemer i 50 forskjellige land [1], og de ble Norges første enhjørningsselskap, det vil si en privateid teknologibedrift verdt mer enn en milliard amerikanske dollar.

2.2 Beskrivelse av AutoStore systemet

AutoStore sitt lagersystem består av flere nøkkelkomponenter som arbeider sammen: gridden, kasser, roboter, porter og en kontrollenhet. Denne beskrivelsen vil gi en oversikt over hver av disse komponentene og forklare hvordan systemet er satt sammen. For å gi leseren en bedre forståelse av komponentene og begrepene som er relevante for vår oppgave.

Grid

Gridden er selve skjelettet til AutoStore-systemet, og består av et fleretasjes rutenett av aluminiumsprofiler. Kassene stables i rutene, mens robotene kjører øverst på aluminiumsprofilene i x-y-retning. Modulariteten til gridden gjør at den enkelt kan maksimere tilgjengelig areal i eksisterende bygg, siden den kan tilpasses ulike former og søyler.

Figur 1 AutoStore Systemet

Kasser

Kassene kommer i tre ulike høyder: 220 mm, 330 mm og 425 mm, alle med en grunnflate på 449 mm i lengde og 649 mm i bredde. Hver kasse kan ha en maksimal vekt på 30 kg. De laveste kassene kan stables opptil 24 stykker, mens de høyeste kan stables 14 stykker. Standardiserte mål sikrer kompatibilitet mellom nytt og eldre utstyr i AutoStore-systemet.

Figur 2 AutoStore Kasse

Roboter

Robotene har oppgaven med å hente og levere kassene mellom gridden og portene. Robotene heiser ned en plattform med samme grunnflate som kassene, låser seg fast til kassen, og heiser den opp. For å hente en kasse som er stablet lengre ned, må roboten først flytte overliggende kasser til en annen plass. Når en kasse settes tilbake, plasseres den øverst i en ledig plass. Kassene har ikke faste plasser i gridden. Dette føre til en passiv optimalisering der ofte brukte kasser forblir øverst, mens sjeldnere brukte kasser havner nederst.

Figur 3 AutoStore Robot

Porter

Portene også kjent som arbeidsstasjoner, er lagerpersonellets eneste fysiske grensesnitt mot AutoStore-systemet. Her senkes kassene ned og presenteres for personalet for plukking og innsetting av varer. Per mai 2024 leverer AutoStore syv forskjellige typer arbeidsstasjoner, designet for ulike behov som effektivitet (kasser levert per time), ergonomi og brukerbehov.

Figur 4 AutoStore Port

Kontroller

Kontrolleren er hjernen av AutoStore-systemet og spiller en avgjørende rolle i driften av hele lageret. Her er noen nøkkelfunksjoner:

- **Styring og koordinering:** Kontrolleren styrer robotene sine bevegelser, sikrer effektiv drift og optimaliserer ruter basert på sanntidsdata.
- **Lagerstyring:** Den holder oversikt over plasseringen av alle kasser, slik at ofte brukte kasser er lett tilgjengelige.
- **Ordrebehandling:** Kontrolleren prioriterer og optimaliserer plukkerekkefølgen for å sikre rask ordrebehandling.
- **Feilhåndtering:** Den overvåker systemets ytelse og håndterer eventuelle tekniske problemer ved å omdirigere oppgaver.
- **Datainnsamling:** Kontrolleren samler inn data om systemets drift for å identifisere forbedringsmuligheter og statistikk.
- **Integrasjon:** Den kan integreres med andre lagerstyringssystemer (WMS) for å automatisere inn og utlevering av varer.

Figur 5 AutoStore Grid

Figur 6 AutoStore Kontroller

Kontrolleren er derfor essensiell for å sikre at AutoStore-systemet opererer effektivt og pålitelig, og muliggjør høy grad av automatisering i lagerdriften.

2.3 Robot modeller

AutoStore har to hovedserier av grid roboter, The Red Line (R5) og The Black Line (B1). Disse robotene er laget for å hente og levere kasser fra gridden til plukkestasjoner. Begge robotene utfører oppgavene ved samme prinsipp, men tilbyr ulik formfaktor og tilleggsfunksjonalitet.

Red Line serien har vært under utvikling i 20 år og kjennetegnes i sin røde farge med overhengs design der roboten dekker to grid celler. Til tross for sitt eldre utseende, har den fått flere oppgraderinger på både elektroniske og mekaniske komponenter innvendig. Nylig lanserte AutoStore to nye modeller R5 Pro og R5+ Pro. Pro-serien innfører for første gang hurtigladning, som kan redusere ladetiden opptil 86% sammenlignet med en standard R5.

Figur 7 AutoStore R5 Robot

Lanseringen av Black Line serien (B1) bryter med AutoStore sitt tradisjonelle R5 design. B1 skiller seg ut ved at den løfter opp kassen innvendig, noe som gjør at den tar opp 25% mindre plass på gridden, og var den første til å håndtere de høyeste kassene. Den største nyheten til B1, var at de nå tilbyr batteribytte funksjon. Batteribyttet tar bare 30 sekunder, noe som gjør at B1 har mye bedre utnyttelses grad. I nyere tid har dette forspranget blitt redusert med introduksjonen av hurtigladning for R5 Pro serien.

Figur 8 AutoStore B1 Robot

2.4 Problemstilling

For at AutoStore skal opprettholde sin nåværende oppetid på 99,7% [1] tildeler de store ressurser til testing og validering av nye programvare revisjoner og nye produkter samt samhandling mellom nye og eldre produkter.
Problemstillingen er å lage ett verktøy for å analysere og reprodusere ulike signaler fra sensorer på AutoStore sine produkter. Grunnen til at bedriften ønsker ett slikt verktøy er at det kan være tidkrevende og noen ganger umulig å fremprovosere tidligere og ønskede feilsituasjoner. Dersom de skal teste om ny programvare eller fysiske endringer har utbedret feilen er det viktig at den tidligere feilsituasjonen blir gjenskapt så likt som mulig for å forsikre seg at feilen har blitt løst.

AutoStore ønsker et verktøy for logging av sensor signaler på de ulike produktene. Verktøyet skal også kunne generere signaler for å se hvordan systemet reagerer, for å forsøke å fremprovosere feilsituasjoner. Dette er i dag tidkrevende, og verktøyet kan være med på å effektivisere testing av kjente feil. Verktøyet kan i teorien også være med på å logge når en feil skjer slik at den kan gjenskapes og eller analyseres med en større nøyaktighet enn tidligere.

AutoStore ønsker et verktøy som skal plasseres mellom sensorene og prosessenheten. Her kan rå data fra sensorer logges og genereres før videresending. Verktøyet skal fysisk kobles mellom sensor og mottaker ved hjelp av kabler. For at verktøyet skal kunne brukes uavhengig av roboten og ikke la seg påvirke av potensielle feilkilder må det være selvforsynt med strøm. Verktøyet må kunne plasseres inne i et AutoStore produkt på en slik måte at det ikke skaper problemer for produktet samtidig som det skal være mulig å koble til de ønskede kablene.

2.5 Hovedidé for løsningen

For å utvikle en løsning som kan utføre disse oppgavene, er det nødvendig med en form for mikrokontroller eller PC som kan ta imot signaler og produsere disse ved behov. En PC vil være upraktisk, da den mangler integrerte løsninger for I/O, er plasskrevende, og bruker mer strøm. Derfor vil en mikrokontroller være en bedre løsning. Mikrokontrolleren må ha tilstrekkelig prosessorkraft til å generere signaler og logge samtidig, samt nok I/O-porter til å koble til alle sensorene. I tillegg bruker mikrokontrollere mindre strøm, noe som er egnet for mobile applikasjoner. Løsningen bør også kunne styres trådløst, siden en mobil robot som R5 og B1 er en av de viktigste bruksområdene.

3 Kravspesifikasjon

AutoStore ønsker en modul som kan monteres i robot, port og andre fremtidige produkter. Modulen skal kunne logge signaler uten forstyrrelser og modifisere signaler for å introdusere kontrollerte feil. For å kunne oppnå dette skal modulen kobles mellom sensor og kontrollenhet. Modulen skal være selvforsynt med strøm og ha tilstrekkelig batteri til å kunne brukes en hel arbeidsdag.

Etter møte med veileder Abiyu Baqala Tashoma fra bedriften har følgende kravspesifikasjon kommet frem.

Fysisk modul som skal kobles mellom sensorer og styringsenhet, som skal ha følgende egenskaper.

- Logge og observere signaler mottatt
- Produsere støy på analoge signaler
- Forsinke signaler
- Overstyre og blokkere signaler
- Selvforsynt med strøm
- Konfigurerbar for ulik funksjonalitet
- Brukervennlighet; produktet skal kunne konfigureres av testavdelingens personell
- Modulen skal kobles på en slik måte at feilkobling ikke medfører kortslutning eller skade på utstyr og personell

Det er også planlagt en rekke funksjoner som vil være aktuelle å implementere dersom tiden tillater det

- Skjerm for avlesning av system status og innstillinger
- GUI programvare for konfigurering
- Modulen skal kunne konfigureres og sende data trådløst via WiFi gjennom en datamaskin
- Det skal være mulig å ha flere moduler i bruk samtidig med WiFi tilkobling
- Kunne sette opp nødvendige hendelser for at mikrokontrolleren skal utføre endringer på signalet(ene)

4 Analyse av problemet

4.1 Signal typer og behandling

I dette kapittelet beskrives bruksområdene til de ulike sensorene som benyttes i R5 Pro, med fokusering på hvordan signalene fra sensorene leses og brukes.

Følgende signaler har blitt identifisert som aktuelle å bearbeide med test modulen.

Robot R5 next gen:						
Nr.	Utstyr	Signal type	Spenning	Frekvens	Antall enheter	Antall ledere
1	Temperaturføler batteri	Analog	2.8V		2	2
2	Temperaturføler batterilader	Analog	2.8V		2	2
3	Nødstopp	Digital	24V		1	4
4	Enkoder	Digital	5V	0-30kHz	4	4
5	Track sensor	Digital	5V	3kHz	4	3
8	Park Brems	Digital	24V		1	2
9	Heis motor	Digital	24V		1	2

Temperaturfølere

Temperaturfølerne (sensor nr. 1 og 2) brukes til å overvåke temperaturen på batteriet og ladekontakten. Disse sensorene er en analog spenningsavlesning over en NTC termistor med målområde fra 0V til 2.8V. Disse avlesningene er ikke tidskritiske og har en relativ treghet knyttet til NTC termistoren, slik at en akseptabel forsinkelse er mindre enn 3 sekund. Ved reproduksjon av disse signalene er det viktig å verifisere at spenningen på signalet ut fra microkontrolleren er lik den spenningen avlest slik at det ikke introduseres feil i målingen.

Bruksområdet til AutoStore for disse temperatursensorene er å overvåke temperaturen på batteriet og ladekontakten. Dette er for å forsikre at temperaturen er innenfor ønsket område, som er spesielt viktig i forbindelse med hurtigladning som gir mer varme enn vanlig ladning. Målet er å logge og manipulere disse temperatursignalene, slik at man kan simulere feil situasjoner uten å fysisk tilføre mer varme på sensorene.

Enkoder

Enkoderne (sensor nr. 4) måler hastigheten og retningen på alle 8 hjulene til roboten. Signalene er et 5V høy/lav firkantbølge og med en frekvens på opptil 30 kHz. For å angi kjøre retningen mottar roboten 2 kanaler fra hver enkoder. Disse signalene er faseforskjøvet med 90 grader, og ved å registrere hvilken kanal som blir høy først kan en finne retningen til enkoderen. I tillegg til retningen kan AutoStore også detektere om hjulspinn og motorproblemer ved å sammenligne rotasjonshastighet med de andre aktive enkoderne. Derfor er det ønskelig å kunne logge og manipulere enkoder signalene slik at det kan skape en rekke ulike kontrollerte feilsituasjoner.

Tracksensor

Tracksensoren (sensor nr. 5) benyttes til å verifisere posisjonen til roboten mens de beveger seg på gridden. Tracksensoren registrerer når en robot passerer et krysningspunkt i rutenettet ved hjelp av en infrarød sensor og sender firkantbølge med en 90% duty cycle når sensoren er i åpen tilstand (ikke passerer krysningspunkt) og en konstant 5V når den er i lukket tilstand (passerer krysningspunkt).

Figur 9 Track

Dataene fra tracksensoren benyttes sammen med enkoder signalene til å verifisere målt avstand mellom hvert krysningspunkt. Roboten bruker også tracksensorene til å verifisere om roboten er hevet eller senket i forbindelse med heising av kasse og endre kjørebane i x-y retning.

Nødstopp, heis motor og park brems

Disse sensorene bruker digitale signaler på 24V for å indikere status. Signalene brukes til:

- **Nødstopp:** Et digitalt signal på 24V brukes til å aktivere nødstopp-funksjonen, som umiddelbart stopper alle operasjoner. Dette signalet er kritisk for sikkerheten og sløyfen måles kontinuerlig av kontrollenheten.
- **Heis motor og park brems:** Disse sensorene bruker også digitale 24V signaler for å indikere status og kontrollere operasjonene til heismotoren og parkbremsen. Signalene leses av kontrollenheten for å styre bevegelsene og sikre korrekt drift.

Analyse

Ved å se på typen signaler er det mulig å avgjøre om en mikrokontroller er egnet til å behandle disse signalene og vil gi grunnlag for å velge rett kontroller til å håndtere disse signalene. Ulike signalspenninger vil også være avgjørende for ulike spenningsnivåer som må være tilgjengelig i modulen. Dette medfører at det må finnes ekstra komponenter som kan regulere spenningene til de nødvendige nivåene for kompatibilitet med mikrokontrolleren. Derfor er det et behov for nedskalering av signaler inn til kontrolleren og oppskalering av signaler ut av kontrolleren.

Tracksensoren leverer en firkantbølge med 90% duty cycle med en 3kHz frekvens for LAV og 5V HØY. Det vil si at lav må kunne leses på 4.5V og det trengs en ekstern krets for å kunne foreta avlesning. Når sensor leser, har den duty cycle på 90% på 3kHz og ved ingen lesing er den konstant 5V. På R5 Pro er det 4 tracksensorer, en på hver side av roboten. Tracksensoren registrerer hver gang roboten passerer en ny rekke på gridden, ved krysningspunktet er det flatt slik at sensoren registrerer høy. Roboten bruker både track- og enkoder signalene til å beregne hvor langt den har kjørt for å vite hvor den er i gridden.

Enkoder og track sensor er satt opp som et par for hver kjøre retning. Ved senere behovsklarering kan det vurderes om det er nødvendig å kunne introdusere feil på hver enkelt eller om det er tilstrekkelig å behandle disse parvis.

4.2 Microkontroller: Analyse av Valg av mikrokontroller.

4.2.1 Kriterier for valg av mikrokontroller

Ved valg av mikrokontroller ble det sett på en rekke ulike modeller.
Et av hovedkriteriene for å velge kontroller baseres på programmeringsspråk på grunn av eksisterende kompetanse. C++ eller C# vil være mest produktivt siden kunnskapsnivået på disse språkene er god.

Mikrokontrolleren må også ha et tilstrekkelig antall inn- og utganger for å kunne håndtere minst 15 ulike signaler. Hvert signal må ha inngang for å kunne leses av og en utgang for å kunne levere et generert signal. Det ble også belyst i signal analysen at det kan være nødvendig å håndtere logikk for introduksjon av egenproduserte signaler som betyr at det vil være nødvendig med et eget styre signal for å kunne overkjøre eksisterende signal. Dette vil i effekt triple behovet for antall utganger og bringer behovet opp til minimum 45 innganger og utganger.
For å kunne håndtere de analoge signaler er det også et krav at det er tilgjengelig et tilstrekkelig antall analoge utganger. Det kan også være et alternativ med digitale PWM utganger.

For å velge en kontroller som kan utføre oppgavene er det behov for tilstrekkelig prosessor kraft. Dette for å sikre at responstiden ikke er en begrensing for de oppgavene som skal utføres i kontrolleren.

For hele modulen er det nødvendig at den er av en størrelse som gjør at det lar seg gjøre å montere i forbindelse med testing uten at den skaper problemer. I forbindelse med portene er det god nok plass til at det ikke setter noen begrensinger.

Som begrensning må modulen la seg montere i forbindelse med B1, da denne er roboten med minst plass. Størrelsen er likevel ingen stor begrensning siden det vil la seg gjøre å montere den øverst på roboten, derimot må det sikres at høyden ikke overskrider maksimal høyde i slusen for innsetting av robot til grid. Dersom den ikke kan monteres på innsiden av robotene vil det medføre noe ekstra arbeid i forbindelse med kabelføring. Det er for eksempel ikke aktuelt å montere noe på sidene av robotene på grunn av klaring til andre roboter på gridden.

4.3 Løsnings analyse

4.3.1 Modul Løsning 1

Mikrokontrolleren blir koblet direkte mellom utstyr og hovedkortet. Den får signalene direkte inn på inngangene fra utstyret og den genererer alle signaler direkte til hovedkortet. Mikrokontrolleren må derfor generere signalet uavhengig om det er en gjengivelse av signalet eller om det er et egengenerert signal. Den er også montert med egen strømforsyning for å unngå signal komplikasjoner.

4.3.2 Modul løsning 2

En løsning der det brukes logikk for signalbehandling og styring mellom utstyrssignal og mikrokontroller. Logikken gjør det slik at det blir minimalt med forsinkelser mellom signal og hovedkort siden mikrokontrolleren ikke trenger bruke prosess kraft for å videresende et uendret signal. Dette gjør det mulig å kunne lese et signal med større spenning enn hva mikrokontrolleren kan lese. Det blir montert oppladbare batterier som er enkle å bytte ut ved utlading.

4.3.3 Vurderinger ved valg av utstyr

4.3.3.1 Mikrokontroller

Blant mikrokontrollere er det flere alternativer som kan utføre oppgaven:

Raspberry Pi [3]: En mikroprosessor som kan kjøre Linux. En Raspberry Pi kan fungere som en datamaskin i microformat, men har begrenset antall I/O tilkoblings muligheter uten utvidelseskort. Med en begrenset erfaring med Linux så vil det ikke være en god plan å benytte denne.

Ved å velge et utviklerkort (development board) så er det enkel mulighet til å koble til sensorer for testing i oppstartsfasen. Det kan også bli aktuelt å produsere et shield for å kunne montere kontakter til sensorene.
Det er mange ulike muligheter innenfor utvikler kort.

Arduino [4]: En familie av åpen kilde kode kort som er laget for prototyping og undervisning. På grunn av utredning så er tilgjengeligheten på kunnskap stor. Tidligere varianter som UNO og MEGA har vært lenge på markedet, men det gjør også at de er basert på utgående maskinvare. GIGA R1 ble lansert i 2023.

Nucleo [5]: Et alternativ til Arduino produsert av ST Microelectrics basert på deres ST32 chip. Utfordringer som er funnet her er at Nucleo programmeres i C og at den mangler EEPROM for lagring av permanente variabler. EEPROM problematikken kan løses med å koble til et shield, men det er en ytterligere hindring for en smidig oppstart av prosjektet.

4.3.3.2 Strømforsyning

For å benytte en mikrokontroller kreves det en strømforsyning som kan dekke kravene satt i oppgaven.

Som et alternativ er det mulig å hente strøm ved å koble seg til strømforsyningen til utstyret som skal tilkobles for logging.

- En løsning for å forsyne mikrokontrolleren for strøm kan være en batteribank, siden det er en løsning som kun krever tilkobling med en USB-C kabel til Arduino GIGA.
- En egen komponert strømforsyning basert på eksisterende produkt, men settes sammen i forhold til oppgavens behov.

4.3.3.3 Spennings kontroll

I det første planstadiet var det ikke kjent at det var signaler med ulik spenning. Når oversikt over signal ble kjent ble det derfor nødvendig å se etter løsninger for å skalere spenning fra sensor til mikrokontroller. Ved generering av signaler må også signalet skaleres til rette spenningsområde.

For analoge signaler kan en benytte spenningsdeler for nedskalering og en op-amp for oppskalering av signalene.

For digitale signaler kan det være mulig å kjøre direkte 3.3V ut dersom denne spenningen er over terskelen for 5V høy-signal for mottaker. Som alternativ finnes det også ferdig produserte logic level converters som kan gi ønsket spenning.

Som andre muligheter finnes det også ferdig produserte løsninger basert på optocoupler, logiske IC moduler og rele.

4.3.3.4 Lagrings medie for logging

Når data skal lagres er det flere muligheter.

- Intern lagring på mikrokontroller er en mulighet dersom mikrokontrolleren har tilstrekkelig lagringskapasitet.
- Lagring på USB minnepenn er også en mulighet dersom mikrokontrolleren har USB kontakt og det finnes styringsfunksjoner i mikrokontrolleren som tillater dette
- Lagring på SD kort ved å benytte en ekstern minnekortleser koblet til mikrokontroller.

4.3.3.5 Signal styring

For signalstyring kan det benyttes å gjøre all styring og konvertering internt på mikrokontrolleren. Da vil modulen bli mye mindre, og det blir mindre fysiske komponenter.

- Det kan kobles opp logikk porter til å lage switcher for å velge mellom flere signal som egengenerert og originalt signal.
- Bruke IC chip med alle de logiske portene sin funksjon og dette tar mindre plass.

4.3.3.6 Ekstra utstyr

- Skjerm som gjør det mulig å ha et enkelt grensesnitt for definering av moduser og enkelt kunne sjekke status på modulen.
- Mulighet for strømmåling for analyse og overvåking av modulens strømforbruk i ulike driftssituasjoner.

4.4 Valg av løsninger

4.4.1 Mikrokontroller

Ved å benytte en programmerbar mikrokontroller som er av kjent virkemåte med programmering av Arduino med C++ vil prosjektet ta nytte av den eksisterende kompetansen i prosjektgruppen. Valget er å benytte en Arduino GIGA R1 WiFi siden denne har WiFi integrert og har tilkobling for ekstern antenne som øker WIFI rekkevidden. Ved å velge en løsning som har mulighet for ekstern antenne blir problemer i forbindelse med at den skal monteres inne i et metall kabinett som kan forstyrre signaler bli utelukket, siden dette gir mulighet for å forlenge antennen til utsiden av kabinettet.

Arduino GIGA har 76 digitale input output tilkoblinger, 12 analoge innganger, 12 utganger med PWM og 2 analoge utganger (DAC). Med så mange tilkoblingsmuligheter vil det gi tilstrekkelig mulighet for å håndtere den nødvendige mengden tilkoblinger, det er også muligheter for å tilføye flere sensorer i fremtiden dersom det er ønskelig. Arduino GIGA R1 ble først utgitt Q1 2023 og har derfor mulighet for videre støtte i flere år fremover.

4.4.2 Batteri

Et av kravene for testmodulen var at den skulle være selvforsynt med strøm gjennom en egen batteriløsning. Selv om det er mulig å koble seg til utstyrets strømforsyning ville dette ha introdusert en mulig feilkilde dersom utstyrets strømforsyning ikke virker som den skal. Ved å benytte en separat batteriløsning oppnås en enklere montering og bedre kompatibilitet med ulike AutoStore-produkter. Og det det må ikke tas hensyn til ulike spenningsnivåer i utstyret som skal tilkobles for logging.

På grunn av behovet for flere spenninger som 2.8V, 5V og 24V, ble det besluttet å utvikle en egen batteriløsning. Et problem med kommersielle batteribanker er at de primært er designet for å lade USB-enheter. Dette krever en USB Power Delivery protokoll, hvor enhetene kommuniserer for å sikre korrekt spenning ved oppstart. En adapter som utfører denne USB-PD håndteringen kunne ha blitt benyttet, men en egen løsning som gir større fleksibilitet ble valgt.

Mot slutten av 2022 utvidet AutoStore installasjonsmulighetene til også å omfatte fryserom. Dette har ført til videreført grundig testing og validering i disse miljøene, da miljøet fører til andre feilsituasjoner enn vanlige temperaturmiljøer. Det var nødvendig å finne en batteritype som kunne operere ved 2 grader, som er temperaturen utenfor gridden i kjølte miljøer. Vanlige Li-ion batterier presterer dårlig i slike forhold, noe som reduserer batteriets levetid. Etter undersøkelser ble LiFePO4 funnet som et godt alternativ. Disse batteriene har en helt annen kjemi som gjør at de er sikrere da de ikke selvantenner ved skade og kan opereres ned mot -20 grader. Ulempen er at LiPO4 har lavere energitetthet, noe som betyr at batteriene blir fysisk større for samme kapasitet. Begge batterityper ble derfor kjøpt inn for testing i ulike temperaturmiljøer, med mål om å finne en balanse mellom energitetthet og temperaturytelse.

4.4.3 Styring av spenning

For å sikre en stabil spenning fra batteriet, ble det valgt å bruke DC-DC omformeren DFR0753 i strømforsyningen. Denne omformeren kan ta imot en variabel inngangsspenning fra 6 til 14V og konvertere denne til en stabil utgangsspenning på 5V, med en kapasitet på opptil 8A. Dette sikrer at strømforsyningen får en stabil inngangsspenning uavhengig av batteritype og spenning, så lenge input spenning er innenfor området 6 til 14V.

For å kunne variere mellom forskjellige spenningskretser ble det valgt to typer DC-DC Adjustable Buck Converter. DFR0379 blir brukt til å steppe ned fra 5V til 2.8V spenning og DFR0123 blir brukt til å steppe opp fra 5V til 24V spenning.

4.4.4 Løsning for logging

Ved logging til internt minne vil kapasitet bli en begrensing siden de fleste mikrokontrollere ikke har stort internminne. Dersom mikrokontrolleren har flashminne vil det være ugunstig å benytte dette til lagring siden flashminne har begrenset antall skrive sykluser. Det vil si at ved hyppig skriving til minnet vil sektorer i minnet bli ubrukelig etter et antall skrive sykluser. Det vil være upraktisk å benytte internminne siden hele microkontrolleren må byttes dersom internminnet er defekt.

Siden Arduino GIGA R1 WiFi ble valgt så er en USB minnepenn et alternativ som lar seg gjennomføre fordi GIGA kortet har mulighet til å benytte biblioteker for bruk av USB lagring. Formfaktoren til USB minnepenn er som kjent utstikkende fra USB porten, og i en liten mobil enhet er det en bekymring for at minnepennen og USB porten skal ta skade som følge av fysisk belastning. En design løsning som tar høyde for beskyttelse av USB minnepenn som er satt i vil også bli komplisert. Selv om en USB-minnepenn er en meget kostnadseffektiv løsning vil en potensiell skade på mikrokontrolleren høyst sannsynlig bety at mikrokontrolleren byttes ut.

Derfor ble det bestemt at loggeløsningen skal bestå av et SD kort i en ekstern leser. Siden SD kortleseren kobler til GIGA kortet med kabel så gir det større fleksibilitet for plassering for tilkomst til SD kortet. SD kortet vil heller ikke være så utstikkende, som gjør at å lage en design som beskytter kortet mot fysisk belastning ikke er komplisert. Kostnadsmessig er SD kort en rimelig løsning som vil gjøre at utskiftning av minnekortet ikke er av betydning. Derfor vil problematikk med flashminne og skrivesykler omgås ved å se på SD kort som en slitedel som må byttes i intervaller.

4.4.5 Løsning for signal styring

For å signalbehandling og styring ble logikk IC chipper valgt for å spare plass og minimere forsinkelser mellom utstyr og hovedkort. Disse IC chippene vil fungere som en switch mellom to innganger sik at det kan byttes mellom for eksempel originalt eller egengenerert signal. Det er også valgt å bruke Logic Level Converter for å enklere steppe opp og ned signaler til ønsket måleområde.

4.4.6 Ekstra utstyr som ble valgt

INA169 Analog DC Current Sensor Breakout vil komme til nytte for å måle strømforbruket til modulen. Denne kan ta målinger som kan analyseres om det er nødvendig med et bedre batteri ved forskjellige driftssituasjoner [6].

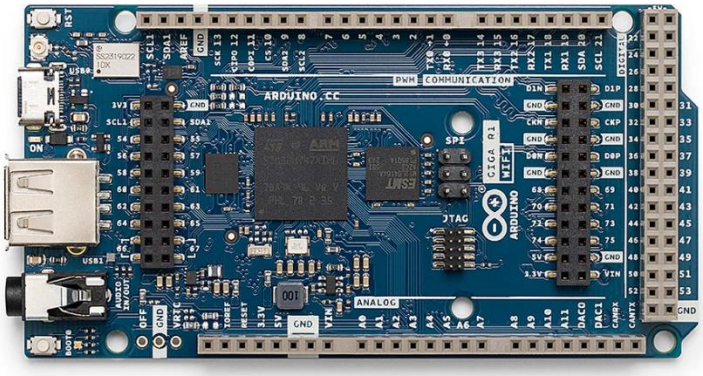
Arduino GIGA DISPLAY Shield ble valgt fordi den kunne bli koblet direkte på Arduino GIGA sin bakside og det ble mulighet med et enkelt Arduino grensesnitt med berøringsskjerm [7].

5 Realisering av løsning

5.1 Komponenter

Se vedlegg [2] Deleliste.

5.1.1 Arduino GIGA R1 WiFi



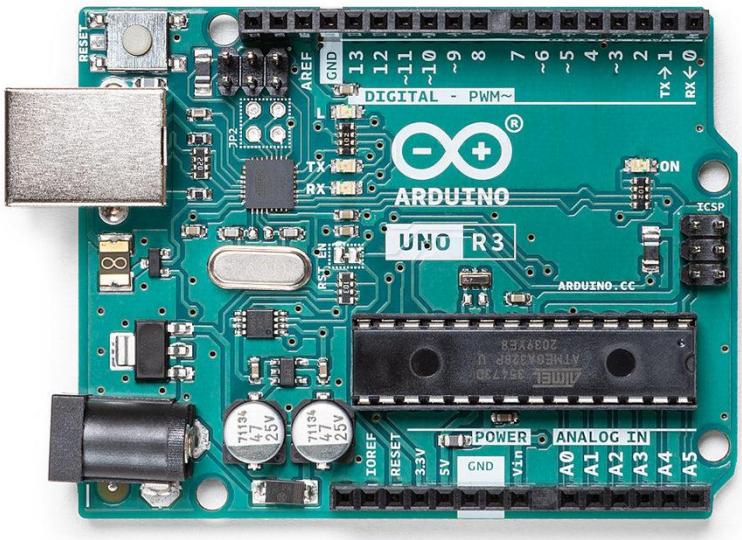
Figur 10 Arduino GIGA R1 WiFi

Et kort basert på mikrokontrolleren STM32H747XI [4]. Denne mikrokontrolleren har to kjerner en Cortex®-M7 som opererer på 480 Mhz og en Cortex®-M4 som opererer på 240 Mhz.

Noen av egenskapene til kortet inkluderer WiFi, 76 digitale inn- og utganger, 12 analoge innganger og 12 PWM utganger. Det store antallet innganger og utganger gjør at dette kortet er egnet til å håndtere mange signaler. Med 2 kjerner kan man dele oppgaver mellom hoved- og sekundærkjernen. GIGA kortet er også kompatibel med Arduino Cloud, som er en IoT plattform som lar deg koble til og styre en Arduino ved hjelp av skyløsningen. Kortet kan forsynes med 6 til 24 Volt.

Se vedlegg [3] ARDUINO GIGA R1 WIFI og vedlegg [4] ARDUINO GIGA R1 WIFI FULL PINOUT

5.1.2 Arduino UNO R3



Figur 11 Arduino UNO R3

UNO R3 er basert på mikrokontrolleren ATmega328P [7]. Den har 14 digitale pinner som kan bli brukt som innganger eller utganger, 6 analoge innganger, 16 MHz keramisk resonator, USB tilkobling, strøminntak, ICSP og en resett knapp.

UNO R3 kan 6 av de digitale pinnene bli brukt som PWM, og kortet i seg selv har en operasjonsspenning på 5V og tilførselsspenning på 7-12V, men maksimum og minimum grensen er 6-20V.

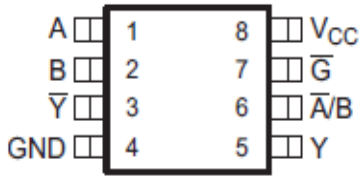
Se vedlegg [5] ARDUINO UNO R3.

5.1.3 Single 2-Line to 1-Line data selector multiplexer SN74LVC2G157 SSOP:

Dette er en integrated circuit chip som har et Signal Switch sett. Her velger man hvilken av 2 inngangssignaler som skal gå ut på utgangen. Hvilken inngang som velges er avhengig av om inngangsvelgeren er HØY eller LAV.

Se vedlegg [6] SN74LVC2G157.

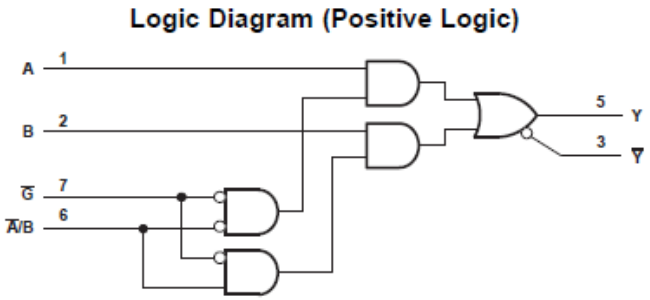
Chip inputs/outputs:



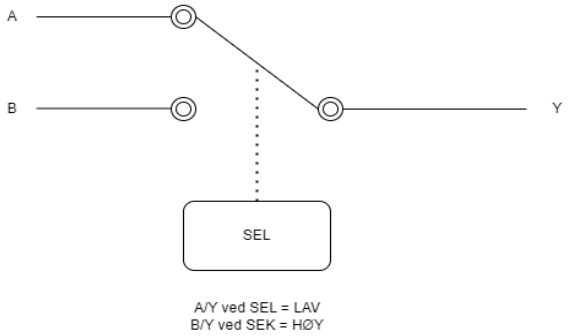
Figur 12 SN74LVC2G157 I/O

A	Inngang 1, spenninger fra 0V til VCC
B	Inngang 2, spenninger fra 0V til VCC
$\overline{Y}$	Negativ verdi av utgangssignalet, Denne er ikke i bruk
GND	0V Jord
VCC	Tilførsel spenning 0V-5V
$\overline{G}$	Inngang for aktivering/deaktivering av Inngangsvelger, Ved ($\overline{G} = GND$) har $\overline{A}$/B lov til styre inngang. Ved ($\overline{G} = VCC$) har $\overline{A}$/B Ikke lov til styre inngang.
$\overline{A}$/B	Inngangsvelger, Ved ($\overline{A}$/B = 3.3V) er Inngang 1 Aktiv mot Utgangen. Ved ($\overline{A}$/B = 0V) er Inngang 2 Aktiv mot Utgangen.
Y	Utgangssignal, Spenning lik som Inngangen som er mellom (GND-VCC).

Logikk Diagram for SN74LVC2G157:



Figur 13 SN74LVC2G157 Logikk Diagram



Figur 14 SN74LVC2G157 Switch Diagram

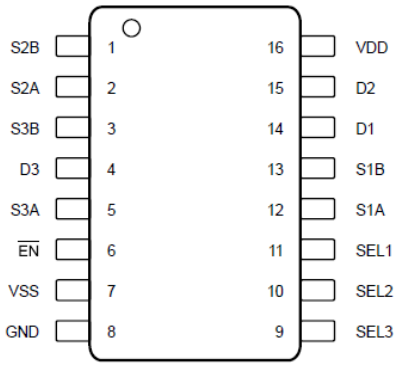
På figur 13 ser man logikken til multiplexeren. Det hadde vært en mer avansert og plass krevende produkt ved å koble opp disse logiske portene enn å bruke IC chipen SN74LVC2G157. Enkelt på figur 14 ser man hvordan switchen fungerer ved Høy og LAV signal.

5.1.4 2:1, 3-Channel Multiplexers with 1.8-V Logic TMUX4053 TSSOP:

Dette er en integrated circuit chip som har tre separate Signal Switch sett. Her velger man hvilken av 2 inngangssignaler som skal gå ut på utgangen for hvert sett. Hvilken inngang som velges er avhengig av om inngangsvelgeren er HØY eller LAV.

Se vedlegg [7] TMUX4053.

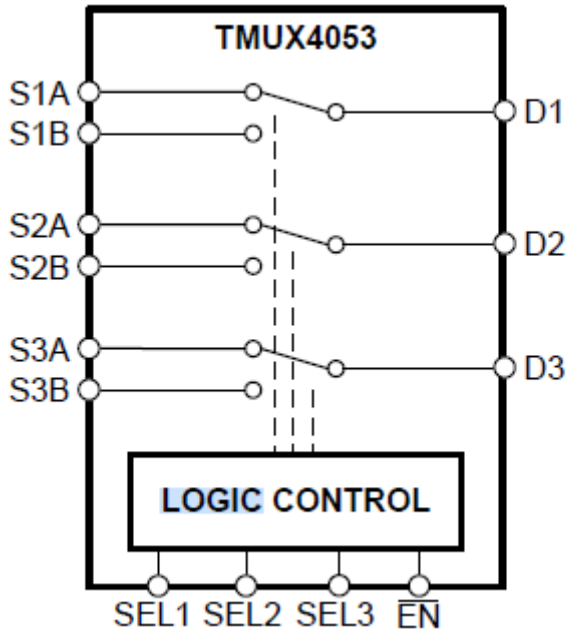
Chip inputs/outputs:



Figur 15 TMUX4053 I/O

SxA, Der x er enten 1,2 eller 3.	Inngang 1, spenninger fra 0V til VDD
SxB, Der x er enten 1,2 eller 3.	Inngang 2, spenninger fra 0V til VDD
GND	0V Jord
VDD	Tilførsel spenning 5V-24V
$\overline{E}$	Inngang for aktivering/deaktivering av alle inngangsvelgere på likt, Ved ($\overline{E} = GND$) har SELx lov til styre inngang. Ved ($\overline{E} = VDD$) har SELx Ikke lov til styre inngang.
SELx, Der x er enten 1,2 eller 3.	Inngangsvelger, Ved (SELx = 3.3V) er Inngang 1 Aktiv mot Utgangen. Ved (SELx= 0V) er Inngan2 Aktiv mot Utgangen.
Dx, Der x er enten 1,2 eller 3.	Utgangssignal, Spenning lik som Inngangen som er mellom (GND-VDD).
VSS	Negativ Tilførsel, Ikke i bruk

Switch Diagram TMUX4053:



Figur 16 TMUX4053 Switch Diagram

Her er en oversikt over hvert Switch sett på figur 16. $\overline{EN}$ er en aktiv LAV logikk styring, det vil si at uten spenning på $\overline{EN}$ styrer SEL hvilken av inngangene, A eller B som skal ut på utgangen D. Ved å tilføre spenning på $\overline{EN}$ vil alle brytere være av.

SEL1 styrer inngangene S1A eller S1B til utgangen D1.
SEL2 styrer inngangene S2A eller S2B til utgangen D2.
SEL3 styrer inngangene S3A eller S3B til utgangen D3.

5.1.5 Bi-Directional Logic Level Converter BOB-12009

En Bi-Directional Logic Level Converter ble valgt til å steppe opp og ned spenninger mellom 0-5V. I systemet behøves en rekke ulike spenninger på de ulike signalene som enten trenger å bli steppet opp eller steppet ned.

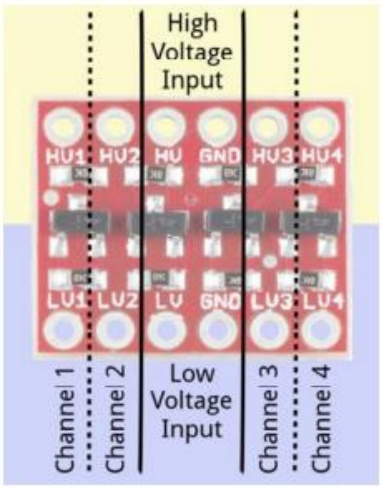
Arduino GIGA: 0V-3.3V:
Temperatur Sensor: 0V-2.8V
Heis, Brems og Nødstopp: 0-24V
Track Sensor: 0-5V

Eksempel: Et signal med spenningsområde 0-5V som har signal verdi på 3V vil gi et signal med verdi på 1.98V ved ned stepping til et spenningsområde på 0-3.3V.

Formel: $Signal\ 2 = \frac{Signal\ 1}{Tilførsel\ 1} * Tilførsel\ 2, \quad Signal\ 2 = \frac{3V}{5V} * 3.3V = 1.98V$

Se vedlegg [8] Bi-Directional Logic Level Converter.

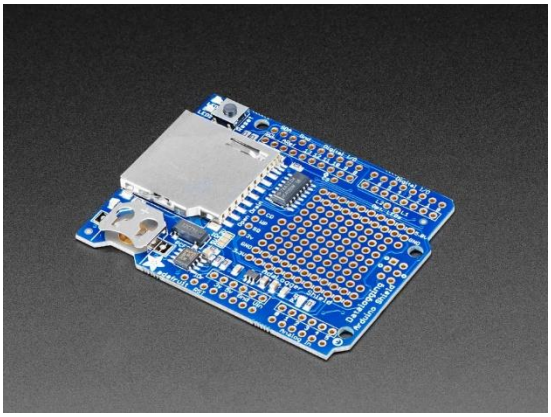
Kort Inputs/Outputs:



Figur 17 Bi-Directional Logic Level Converter I/O

HV	Største Signal sin tilførselsspenning (0V-5V)
LV	Minste Signal sin tilførselsspenning (0V-5V)
GND	0V Jord
HV1	Største Signal Kanal 1
HV2	Største Signal Kanal 2
HV3	Største Signal Kanal 3
HV4	Største Signal Kanal 4
LV1	Minste Signal Kanal 1
LV2	Minste Signal Kanal 2
LV3	Minste Signal Kanal 3
LV4	Minste Signal Kanal 4

5.1.6 Adafruit Data Logging Shield



Figur 18 Adafruit Data Logging Shield

Adafruit Data Logging Shield er et utvidelses skjold til Arduino UNO familien. Et utvidelses skjold er et kort som monteres over Arduino kortet. Skjoldet er designet for logging og har en integrert RTC sammen med en SD kort leser. Kortet har et prototypefelt hvor man kan bygge egne kretser. Kortet har også egen 3.3V regulator for å kunne levere pålitelig strøm til SD kortet. SD kortet må formateres med FAT16 eller FAT32 for å kunne brukes. Kortet har også plass til et knappcellebatteri som gjør at klokke funksjonen fortsetter uten annen strøm kilde. Kortet kobles til en mikrokontroller ved hjelp av I2C eller SPI.

Se vedlegg [9] Adafruit Data Logging Shield.

5.1.7 Arduino GIGA Display Shield

Figur 19 Arduino GIGA Display Shield

For å øke brukervennligheten til test modulen var det tenkt å implementere en HMI skjerm, noe som det ikke ble tid til. Det var et ønske å bruke skjermen til å se test modulens gjenværende batteriprosent, oppsett av nettverk, signaler i sanntid og utføre enkle signalbehandlinger. En av grunnene for at Arduino GIGA R1 ble valgt som hovedkontroller var på grunn av at den er kompatibel med Arduino GIGA Display Shield [7]. Dette gjør at skjermen kan feste direkte på baksiden av Arduino GIGA uten å bruke eksterne kabler.

Se vedlegg [10] Arduino GIGA Display Shield.

5.1.8 Batteri komponenter

5.1.8.1 Spektrum 7.4V 5000mAh LiPo HC G2 IC3

Spektrum 7.4V 5000mAh LiPo HC G2 IC3 [9] er et høyytelse litium-polymerbatteri, med en kapasitet på 5000 mAh som gir lang driftstid mellom ladninger.

Spesifikasjoner og egenskaper:

- **Kapasitet:** 5000 mAh
- **Spenning:** 7.4V, som oppnås ved hjelp av to celler i serie.
- **Vekt:** 315 gram, som gjør det relativt lett og enkelt å håndtere.
- **Kontakttype:** IC3
- **Anbefalt ladestyrke:** 7.5A, som er trygg for regelmessig lading.
- **Maks ladestyrke:** 25A, som muliggjør raskere lading ved behov.
- **Kabeltype:** 12AWG, som er en tykk og robust kabel, egnet for høye strømstyrker.
- **Temperaturområde:** Driftstemperatur mellom -20°C og 60°C, lagringstemperatur mellom 0°C og 40°C.

Sikkerhet og effektivitet:

Batteriet har innebygd balanseringskontakt som sørger for jevn lading av cellene, noe som bidrar til lengre levetid, bedre ytelse og sikkerhet. IC3-kontakten er designet for å minimere motstand og maksimere effektiviteten, noe som resulterer i mindre varmeutvikling og høyere strømlevering.

Konklusjon:

Spektrum 7.4V 5000mAh LiPo HC G2 IC3 er et robust og effektivt batteri som gir god ytelse og lang driftstid. Med sine sikre og effektive ladeegenskaper, er det et ideelt valg for å drifte en testmodul bestående av en Arduino mikrokontroller, spesielt i miljøer hvor stabil strømforsyning er kritisk. Det er viktig å merke seg at dette LiPo-batteriet presterer best i moderate temperaturer og kan ha redusert ytelse i svært kalde forhold.

Figur 20 Spektrum 7.4V 5000mAh LiPo HC G2 IC3

5.1.8.2 Spektrum 9.9V 3200mAh LiFe IC3

Spektrum 9.9V 3200mAh LiFe IC3 [10] er et høyytelse litium-polymerbatteri, med en kapasitet på 3200 mAh som gir lang driftstid mellom ladninger.

Spesifikasjoner og egenskaper:

- **Kapasitet:** 3200 mAh
- **Spenning:** 9.9V, som oppnås ved hjelp av tre celler i serie.
- **Vekt:** 245 gram, som gjør det relativt lett og enkelt å håndtere.
- **Kontakttype:** IC3
- **Anbefalt ladestyrke:** 3,2A, som er trygg for regelmessig lading.
- **Maks ladestyrke:** 4.8A, som muliggjør raskere lading ved behov.
- **Kabeltype:** 13AWG, som er en tykk og robust kabel, egnet for høye strømstyrker.
- **Temperaturområde:** Driftstemperatur mellom -10°C og 60°C, lagringstemperatur mellom -20°C og 40°C.

Sikkerhet og effektivitet:

Batteriet er utstyrt med IC3-kontakt. Dette sørger for effektiv strømoverføring med minimalt varmetap. LiFe-teknologien er kjent for sin sikkerhet og lange levetid sammenlignet med andre batterityper. En stor fordel med LiFePO4-batterier er den har bedre ytelse i kjøligere temperaturer sammenlignet med LiPo-batterier, noe som gjør dem mer pålitelige i kalde omgivelser.

Konklusjon:

Spektrum 9.9V 3200mAh LiFe HC G2 IC3 er et pålitelig og effektivt batteri som tilbyr en god kombinasjon av kapasitet, vekt og ytelse. Med sikre ladestyrker og robust tilkobling, er det et ideelt valg for å drifte en testmodul bestående av en Arduino mikrokontroller, spesielt i miljøer hvor både vekt, sikkerhet og evne til å operere i kjølige temperaturer er viktige faktorer.

Figur 21 Spektrum 9.9V 3200 mAH LiFe HC G2 IC3

5.1.8.3 Spektrum S100 USB-C Smart Charger 100W

Spektrum S100 USB-C Smart Charger 100W [11] er en lader utviklet for å levere pålitelig og effektiv lading til et bredt spekter av batterityper, inkludert LiPo og LiFePO4, som de nevnte Spektrum 7.4V 5000mAh LiPo og Spektrum 9.9V 3200mAh LiFe batteriene. Med en maksimal effekt på 100 watt, er denne laderen ideell til bruk i oppgaven for en rask og sikker ladning av batteriene.

Spesifikasjoner og egenskaper:

- **Maksimal effekt:** 100W, med justerbar ladehastighet på 1A / 2A / 3A /6A.
- **Tilkobling:** USB-C
- **Ladekompatibilitet:** Støtter ulike batterityper, inkludert LiPo, LiFePO4, NiMH, og Pb (blybatterier).
- **Sikkerhetsfunksjoner:** Innebygd beskyttelse mot overlading, overstrøm, kortslutning, og overoppheting.
- **Brukergrensesnitt:** LCD-display som gir enkel tilgang til ladestatus og innstillinger.
- **Smart teknologi:** Automatisk gjenkjenning av batteritype og justering av ladeparametere

Kompatibilitet med valgte batterier:

Spektrum S100 USB-C Smart Charger er tilpasset for å lade Spektrum 7.4V 5000mAh LiPo HC G2 IC3 og Spektrum 9.9V 3200mAh LiFe HC G2 IC3 batteriene. Laderen justerer automatisk ladestrømmen basert på C-rating og batterikapasitet angitt av batteriets innebygget styring:

- **For 7.4V 5000mAh LiPo:** Anbefalt ladestrøm er 5A (1C), maks ladestrøm er 25A (5C).
- **For 9.9V 3200mAh LiFe:** Anbefalt ladestrøm er 3,2A (1C), maks ladestrøm er 4,8A (1,5C).

Konklusjon:

Spektrum S100 USB-C Smart Charger er ideell til de valgte batteriene på grunn av den smarte ladefunksjoner, sikkerhetsmekanismer, og brukervennlighet. Dette fører til sikrere bruk der en ungår å lade batteriene utenfor anbefalt ladehastigheter. Selv om batteriet ikke kan lade LiPO batteriet med maks anbefalt ladehastighet på 25A, er 6A rask nok til vårt bruk. Dersom det er behov for raskere ladning kan laderen byttes ut med en som er kraftigere, eller gå i innkjøp av ekstra batterier for å bytte mellom.

Figur 22 Spektrum S100 USB-C Smart Charger

5.1.8.4 Spektrum Kontakt – IC3 Male

Spektrum IC3 Connector [12] er en pålitelig og høy ytelses kontakt designet for å sikre effektive og trygge forbindelser i elektriske systemer, spesielt egnet for batterier og ladere i radiostyrte modeller og andre elektroniske applikasjoner. Denne kontakten er kjent for sin evne til å håndtere høye strømstyrker med minimal motstand og varmeutvikling.

Spesifikasjoner og egenskaper:

- **Strømkapasitet:** Kan håndtere høye strømstyrker opptil 60A kontinuerlig.
- **Design:** Ergonomisk utforming for enkel tilkobling og frakobling.
- **Sikkerhet:** Innebygd polaritetssikring for å forhindre feil tilkobling og sikre trygg bruk.

Bruksområder:

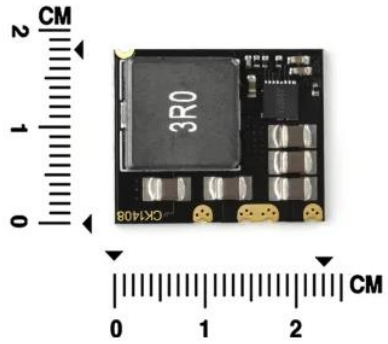
IC3-kontakten er ideell for bruk med høy ytelses LiPo- og LiFe-batterier, ladere, ESC-er (Electronic Speed Controllers), og andre applikasjoner som krever sikre og effektive strømforbindelser.

Figur 23 Spektrum IC3 Connector

5.1.9 DC-DC BUCK CONVERTER DFR0753

Siden batteriets spenning reduseres ved utladning, ble det funnet et behov for å kontrollere spenningen. Hadde DC-DC omformere blitt koblet direkte til batteriet og innstilt omformeren til 2,8 V mens batteriet var fulladet, ville spenningen falle etter hvert som batteriet blir utladet. Dette ville kunne føre til problemer med de elektroniske kretsene, som tilsynelatende ikke hadde noen kilde. Det kunne også forårsaket problemer for de elektroniske komponentene.

For å sikre en stabil spenning fra batteriet, ble det valgt å bruke DC-DC omformeren DFR0753 [13] i strømforsyningen. Denne omformeren kan ta imot en variabel inngangsspenning fra 6 til 14V og konvertere denne til en stabil utgangsspenning på 5V, med en kapasitet på opptil 8A. Dette sikrer at strømforsyningen får en stabil inngangsspenning uavhengig av batteritype og spenning, så lenge det er innenfor området 6 til 14V.



Figur 24 DFR0753

5.1.10 Adjustable Buck Converter

Etter å ha sikret at strømforsyningen gav en fast spenning på 5V, ble det også nødvendig å fremstille 2.8V og 24V. For å løse dette på enklest mulig måte ble det brukt to justerbare dc-dc transformatorer også kjent som «buck convertere». En for å øke spenningen, STEP-UP og en for å redusere spenningen, STEP-DOWN. I en kommersiell sammenheng ville det vært best og funnet komponenter med en fast justering for å redusere størrelsen og være mer energieffektiv. En fordel med omformerne som er valgt er at de har en digital skjerm som viser utgangspenningen, noe som gjør det lettere å justere utgangs spenningen og verifisere i ettertid at den er korrekt.

5.1.10.1 DC-DC Adjustable Step-Down Buck Converter DFR0379

For å redusere spenningen fra 5V til 2.8V ble DFR0379 [14] valgt. Denne kan redusere spenninger opp til 40v med en nøyaktighet på ± 0.05V og strøm opp til 3A. Se vedlegg [11] DFR0379.

Figur 25 DFR0379

5.1.10.2 DC-DC Adjustable Step-Up Buck Converter DFR0123

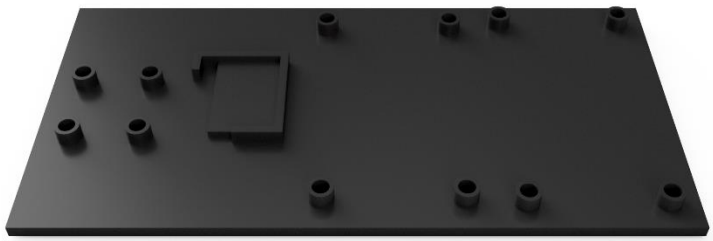
For å øke spenningen fra 5V til 24V ble det valgt en lignende løsning ved å bruke DFR0123 [15]. Denne har et inngangsområde fra 3v til 32v og ett utgangsområde på 5v til 35v opp til 4A med samme nøyaktighet på ± 0.05V. Se vedlegg [12] DFR0123.

Figur 26 DFR0123

5.2 Design av fysisk produkt

AutoStore ønsket at testmodulen skal kunne brukes på alle produktene. For å oppnå disse kravene ble det undersøkt hva som måtte til for å gjøre testmodulen kompatibel med alle robotene og portene. Det ble raskt oppdaget at de ulike produktene ikke hadde like koblinger og kabling. For at produktet skulle bli kompatibelt med nåværende og fremtidige produkter ble et modulært design valgt. Med et slikt design kan et adapterkort designes spesifikt mot hvert produkt. Adapterkortet er utstyrt med de riktige koblingene for sensorkablene og elektronikken som gjør signalene kompatible med en Arduino GIGA. Fordelen med dette er at test modulen slipper å ha alle ulike koblingene for produktene på én og samme plass, og siden flere har like koblinger, men med ulik kabling, kan dette føre til feil som kan forårsake kortslutning ved eventuelle feilkoblinger. Den største fordelen med modularitet er at det gjør det mulig å oppgradere testmodulen med nye adapterkort for fremtidige produkter og bytte mellom strømforsyninger. Det vil derfor være viktig å lage en standardisering med en omfattende dokumentasjon for at nye og reviderte moduler skal bli kompatible med Arduino-hovedkortet.

5.2.1 3D-Print design strømforsyning



Figur 27 Strømforsyning 3D-Print Design

Ettersom strømforsyningen bestod av 3 stykk dc-dc transformatorer og muligheten for strømmåler så ble det nødvendig å lage en monteringsplate. Dette gjør det lettere å flytte på strømforsyningen og øker sikkerheten betraktelig da belastningen fra koblingspunktene blir redusert. En monterings plate ble benyttet som en tidligfase prototype for montering av disse komponentene.

Tidligere erfaring med 3D-printing, Autodesk Fusion 360 og tilgang på flere 3D-printere hos AutoStore gjorde at det ble valgt å starte design prosessen. Etter å ha målt alle komponentene ble braketten designet i Fusion 360. De justerbare transformatorene og strømmåleren hadde hull til skruer så braketten ble designet til å bruke gjengeinnsatser som er hevet opp fra platen. Gjengeinnsatsene smeltes inn ved bruk av en loddebolt. 5v transformatoren hadde ingen skrufester da den er laget for å monteres direkte på ett kretskort, og det måtte derfor finnes en annen løsning på å montere denne på.

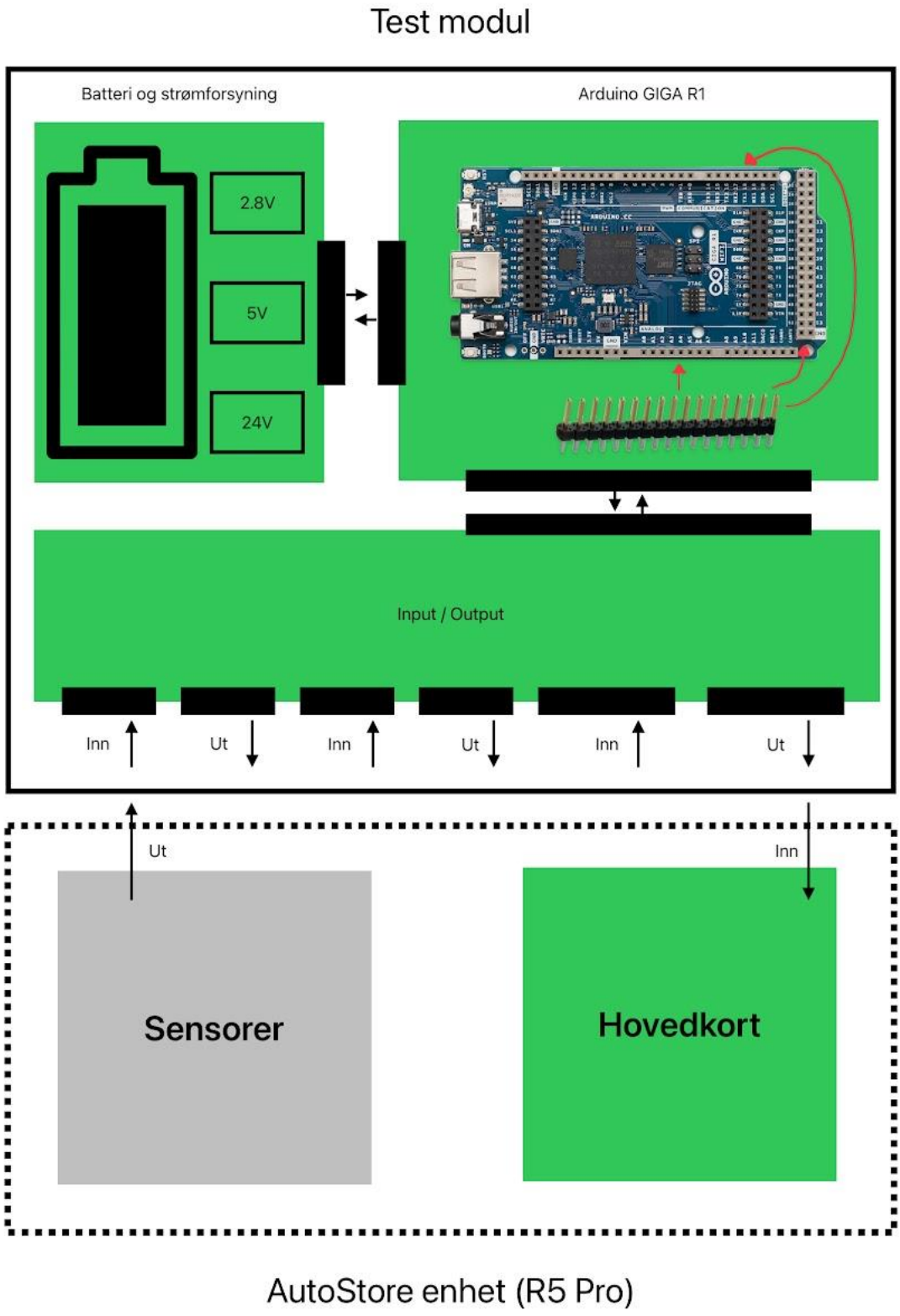
Løsningen ble å lage en hevet plattform som gjør at lodding og kabler ikke kommer i konflikt. For å holde den på plass ble det laget en ramme som dekker hver side, med unntak av ett hjørne som gjør at komponenten enkelt kan presses på plass og fjernes. Etter å ha 3D printer første versjon ble det oppdaget at det var feil på målene på 5V transformator festene. Det tok derimot kort tid å fikse opp i feilen og lage en ny versjon med oppdaterte mål. Dette er en av fordelene med å ha tilgang på 3D-printer der en enkelt og kjapt kan lage ett produkt ut ifra et 3D-design og nye revisjoner kan skrives ut over natten. Som verktøy for prototyping er 3D printing en revolusjon som gjør at alle enkelt kan lage enkle prototyper. Det var plan å få 3D printet en modulær innkapsling for modulen slik at kontrollenheten med tilkoblinger til sensorer var i et eget kabinett. I tillegg var det tenkt et eget kabinett til strømforsyningen slik at den ble enkel å bytte ut og frakte med seg uten at andre komponenter vil bli berørt. Det ble kun laget enkle skisser for dette før det ble lagt til siden til fordel for mer kritiske oppgaver.

5.2.2 PCB Design

Etter testing og utvikling av elektronikken var målet å lage et kretskort med samme kretsene som var blitt lagd og koblinger til R5-Pro. Designprosessen startet i EaseEDA, men på grunn av begrenset erfaring oppstod flere utfordringer. Planen var å designe kretskortet slik at det kunne bestilles ferdig montert med alle komponentene fra PCBWay. For å oppnå dette måtte kompatible komponenter og koblinger med R5Pro identifiseres.

Etter å ha funnet de nødvendige komponentene, oppstod problemer med å integrere dem i PCB design-programmet. Hver komponent krever tre filer: 3D-design, PCB-design og koblingspunkter. Hvis en av disse mangler, kan ikke komponenten brukes i programmet, da alle er essensielle. Etter å ha testet flere ulike programmer, ble det konkludert at det ikke var forsvarlig å bruke mer tid på dette, grunnet leveringstiden på kretskortene og den høye sannsynligheten for designfeil på første revisjon.

Derfor ble det besluttet å beholde de eksisterende kretsene på breadboard, da dette var mer tidseffektivt og reduserte risikoen for ytterligere forsinkelser.



Figur 28 PCB Design

5.3 Koblingsskjema

Det er laget et koblingsskjema for hver signalbehandling og tilkoblede komponenter. Det vises enkelt hva så går til hva og det brukes samme navn som i koden for enkelhets skyld. Dette gjør det lettere å sjekke for feil og få god oversikt over komponentene.

For design av koblingsskjema ble det brukt et Online program som heter TinyCad [16] og dette var veldig greit å bruke fordi det hadde innebygde funksjoner for komponent tilkobling. Det har vært litt utfordrende med plass på hver skjemaside, men det er tydelig å se hva som blir koblet til hvor. Det hadde vært gunstigere med et skikkelig designprogram i forhold til utseende og tidsbruk.

Se vedlegg [13] AutoStore Test Module Connection Diagram.

5.4 Kodestandard:

En kodestandard er en beskrivelse av regler og konvensjoner som kode skal følge for å holde koden uniform. Hensikten er at koden skal skrives likt av alle deltagere i prosjektet, dette øker lesbarhet og forståelse for koden for alle som skal jobbe på koden.

Nye deltagere i prosjektet har også en plass å starte for å kunne forstå hvordan koden er skrevet. Ved å etablere en standard for koden vil gjenbruk og testing av koden være mer smidig.

Engelsk benyttes som basespråk for kommentarer, variable og dokumentasjon. Kodestandarden blir derfor skrevet på engelsk. Dette valget er gjort på grunn av at AutoStore er et internasjonalt selskap med mange ulike nasjonaliteter.

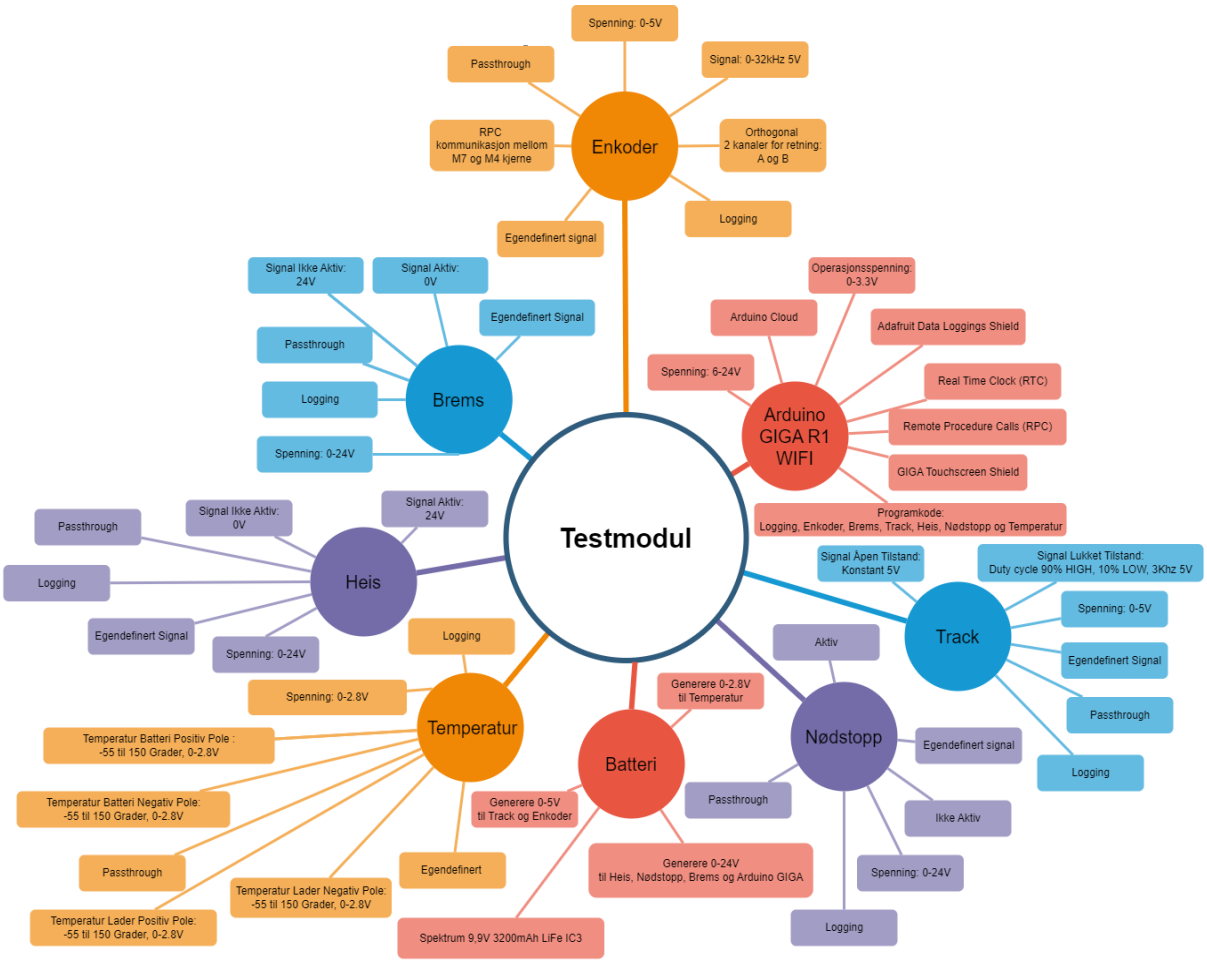
Å benytte en kodestandard er god praksis i et prosjekt hvor flere jobber med samme kode, utfordringen er å få utvikler til å følge standarden.

Se vedlegg [14] CodeStandardASTestModule.

5.5 System

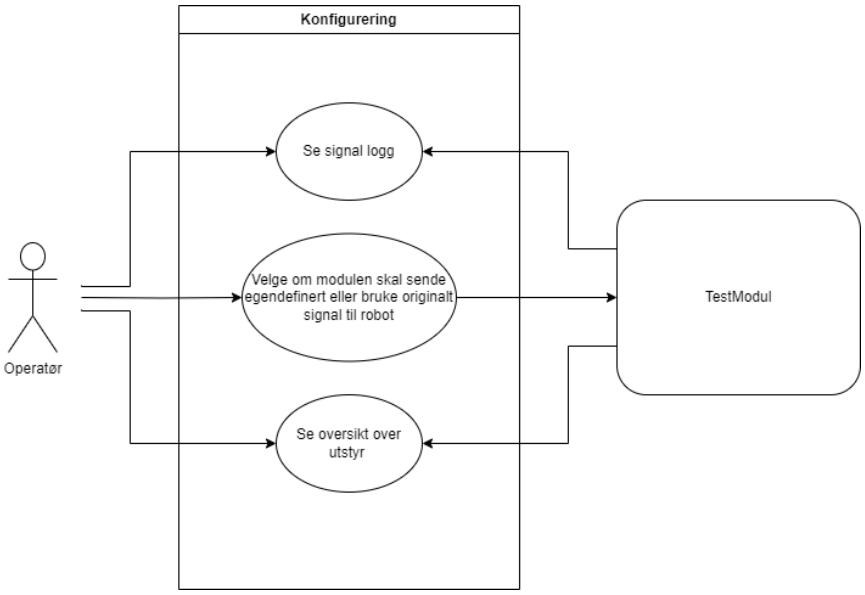
Programmet Drawio [17] ble brukt for produsering av diagram.

5.5.1 System Oversikt



Figur 29 Systemoversikt

5.5.2 Use Case Diagram



Figur 30 Use Case Diagram

6 Testing

6.1 Signal Testing

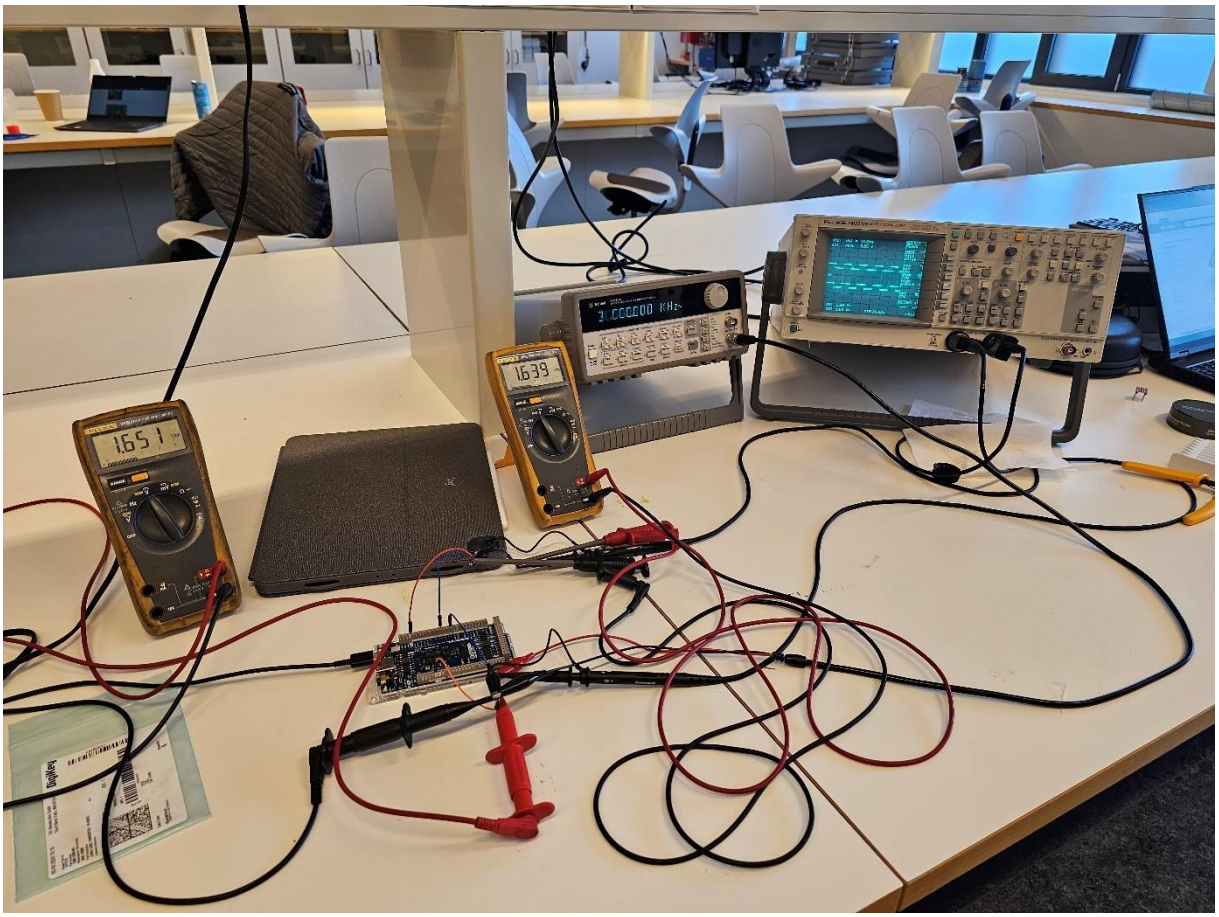
6.1.1 Tidlig testing på laboratoriet

Etter at oppgaven ble satt laget AutoStore en oversikt som beskriver sensorene. Basert på denne oversikten så ble det funnet at enkoder signalet hadde størst usikkerhet knyttet til seg. Med frekvens opp mot 30 kHz oppsto det usikkerhet knyttet til mikrokontrollerens evne til å lese og reprodusere signalet. Det ble derfor opp 3 problemstillinger som måtte testes og besvares.

1. Målefeil i forbindelse med HØY frekvens og presisjonen på lesing
2. Mikrokontrollerens evne til å reprodusere et signal
3. Forsinkelsen som oppstår i forbindelse med reproduksjon og muligheten for at det ødelegger signalet.

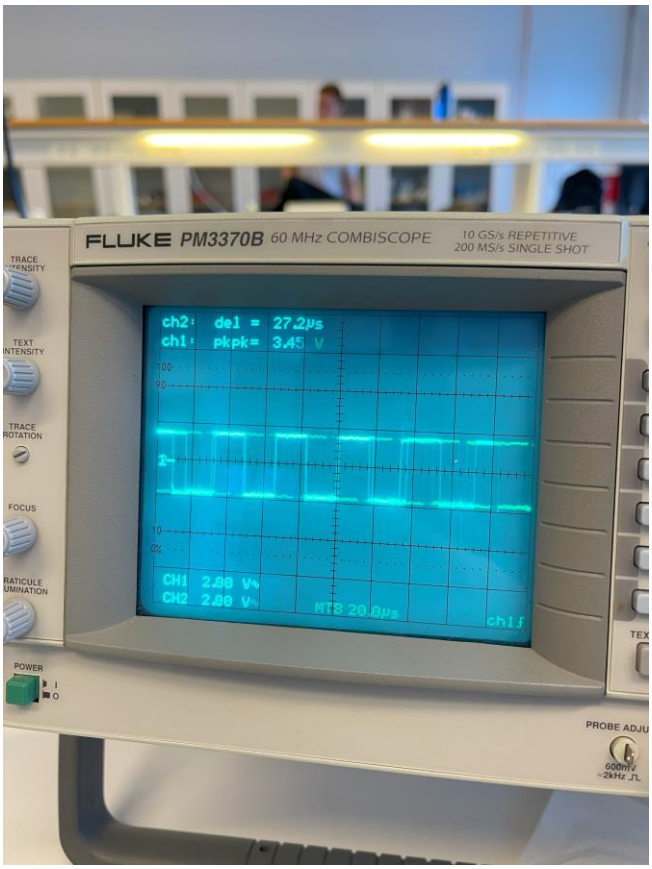
På grunn av usikkerheten om komponentene ville klare å utføre disse oppgavene ble det besluttet å prioritere testing av mikrokontrolleren. Skulle testingen få et negativt utfall så ville det være mindre arbeid som ble forkastet ved valg av en annen mikrokontroller.

Til å begynne med for å sikre at alle i gruppen brukte oscilloskopet på rett måte ble det satt opp en enkel brukerveiledning som ble gjennomgått av alle deltagerne. Se vedlegg [15] Test prosedyre for signaler til modulen.



Figur 31 Oscilloskop Test

Testen er beskrevet i vedlegget «Test prosedyre for signaler til modulen» dette ble gjort for å kunne gjenskape testen dersom det skulle oppstå behov for det senere. Kort oppsummert ble et signal generert inn til GIGA kortet ved hjelp av en bølgeformgenerator, input og output fra GIGA kortet ble så avlest ved hjelp av et oscilloskop for å kunne sammenligne signalene. Det ble teste med ulike frekvenser for å sjekke at resultatene var konsistente over hele frekvens båndet.



Figur 32 Oscilloskop Test Signal

Ved å benytte oscilloskopet til å verifisere det genererte signalet ut av mikrokontrolleren ble det bekreftet at Arduino GIGA kan reprodusere signalet. Forsinkelsen ble funnet til å ligge i området 1 til 30 mikrosekund.

Det var vanskelig å samle datapunkter som ville beskrive forsinkelsen siden kunne hoppe direkte fra 2 til 26 mikrosekund på en oppdatering på oscilloskopet. For å vise dette er det lagt ved et opptak som ble gjort i forbindelse med målingene.

Se vedlegg [16] oscilloskoptest video.

6.2 Kode testing

6.2.1 Kode

En Arduino mikrokontroller programmeres med kode skrevet i Arduino Programming Language som er basert på programmeringsspråket C++. Koden lastes vanligvis opp til et Arduino brett via en USB-kabel ved hjelp av programmet Arduino IDE.

Tidligere erfaring med Arduino UNO og NANO har vist at det er ganske omfattende med dokumentasjon og ressurser på nett. Det viser seg etter hvert at det ikke er like dekkende for Arduino GIGA. Det finnes brukbar dokumentasjon om Mbed som er operativsystemet som benyttes i Arduino GIGA, men dokumentasjonen om Arduino sin implementering av Mbed er vanskeligere å finne. Det er ikke alltid like tydelig hvilken funksjonalitet som finnes i Mbed som også er tilgjengelig i Arduino sin versjon.

Arduino IDE:

IDE står for intregrated development environment og er en betegnelse på programvare som er laget for utvikling av annen programvare. En IDE er et verktøy som benyttes for å danne kode fra et kodespråk over til maskininstruksjoner og inneholder funksjoner som redigerings verktøy for kildekode, kompilering og debugging.

Ved kompilering av koden vil programmet gi feilmelding dersom koden ikke lar seg oversette. Her vil ofte syntaks feil og glemte variable gi feilmelding.

Selv om koden lar seg kompilere og laste opp er det ikke dermed automatikk at koden utfører det man ønsker. Spesielt hvis man har gjort feil som ikke kompilering kan fange opp. Det kan enklere forklares som at koden ikke er ulovlig, men brukes på feil måte og derfor ikke gir ønsket resultat.

Eksempel på kode som er korrekt, men ikke gir ønsket resultat:

Ved å opprette en funksjon som returnerer en char* variabel så kan man ikke returnere en char eksempel[] variabel. Kompilatoren ser ingen problemer her for dette er lovlig kode. Problemet oppstår siden en char eksempel[] variabel som er opprettet inne i funksjonen kun returnerer en adressen noe er lagret på og ikke det som ble lagret på adressen. Forståelsen for dette er meget viktig og er på ingen måte intuitivt. Uten kjennskap til problemstillingen vil man kanskje aldri finne ut hvorfor det oppstår, men har man erfaring med kode så tar det bare 2-3 timer å finne ut av.

Eksempel på kode som ikke er korrekt, men lar seg kompilere:

Ved å opprette en char eksempel2[4] vil man kunne holde 3 verdier og en null for å avslutte. Dersom man så setter denne inn i en for-loop hvor man går igjennom 4 verdier (int i; i < 5; i++) så vil koden la seg laste opp og kjøre frem til den krasjer i det den forsøke å sette inn den fjerde verdien.

Se vedlegg [17] Sammenstilt Programkode.

6.2.2 Kode Biblioteker

Prinsippet med biblioteker er at utvikleren som skriver kode legger til de biblotekene som det er behov for. På grunn av begrenset kapasitet på en mikrokontroller er det nyttig å kun inkludere funksjoner som det er behov for. Dette gjøres for å spare minne og internlagring.

For å kunne benytte funksjonalitet knyttet til GIGA er det nødvendig å inkludere biblioteket «mbed.h», som er operativ systemet til GIGA. Arduino sin implementering «Arduino Core-mbed» lar brukere koble til utstyr som sensorer og ekspansjons skjold. «mbed_mktime.h» er et bibliotek for håndtering av tid og forenkler bruken av RTC.

«WiFi.h» og «WiFiUdp.h» benyttes for å håndtere tilkobling med WiFi og UDP over WiFi.

«thingProperties.h» er et bibliotek som er automatisk generert ved bruk av Arduino IoT Cloud og inneholder de deklarerte sky variablene, nettverknavn, passord og metode for tilkobling til Arduino Cloud.

6.2.3 Feilsøking i kode

I forbindelse med koding så er det nødvendig å feilsøke koden når den ikke virker som den skal. Feilsøking kan være en frustrerende og vanskelig prosess som setter programmererens mentale helse på prøve. Derfor er det nødvendig å vise litt hvordan noen av problemene har blitt løst.

I Arduino IDE er det mulig å sette opp breakpoints men nyttigheten av disse er ganske begrenset. Derfor har det blitt benyttet Serial.println(); som en indikator på hvilke punkter i koden som har blitt passert. Ved hjelp av denne metoden kan man finne hvor koden har stoppet og dermed hva som har stoppet. Da gjenstår bare hvorfor det stoppet. Alle Serial.print som er benyttet skal kommenteres ut fra koden når den er ferdig feilsøkt.

En annen utfordring med feilsøking kan også oppstå dersom feilen skyldes andre ting enn selve koden. Det kan finnes feil i kompilatoren som gjør at lovlig kode ikke kan kompileres og man må søke alternative metoder å programmere en løsning. Det er også enkelte tilfeller av hardware og firmware problemer som gjør at kode eller funksjoner ikke kan benyttes. Disse eksterne feilene skjer sjeldent, men de kan forekomme.

6.2.4 Testing av kode

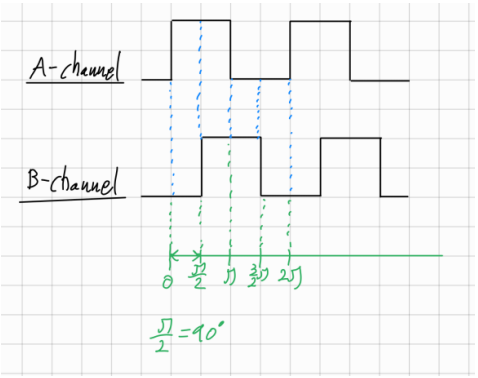
Når koden kjører uten programmeringsfeil så er det nødvendig å verifisere at koden utfører ønsket funksjonalitet. Det må også utføres verifikasjon av at programmet ikke møter på nye feil når nye tilstander introduseres til systemet. I flere tilfeller vil en endring føre til en alternativ kodebane som gjør at det er nødvendig å teste disse også. Når nye feil oppdages må ny feilsøking starte, i enkelte tilfeller kan også utbedringen av feil føre til ytterligere feil.

I verste fall må helle koden rulles tilbake til en tidligere versjon da de feilene som har oppstått er så omfattende at det er enklere å forsøke å implementere en annen løsning. Det er ingen enkel avgjørelse å forkaste flere timer med arbeid, men kan være bedre å forsøke å løse problemet fra en annen vinkel.

Når nye feil er utbedret starter testing på nytt for å verifisere ønsket resultat av koden. Dette gjør at en syklus med feilsøking og testing kan bli meget tidkrevende siden syklusen kan måtte gjentas flere ganger før ønsket resultat oppnås. Oppbygging av funksjoner som håndterer spesifikke kall kan være svært nyttig for å dele opp koden og forenkle feilsøking. Denne oppdelingen fører til at man kan teste deler av koden og eliminere feil i hver del.

Se vedlegg [17] Sammenstilt Programkode.

6.2.5 Enkoder



Figur 33 Enkoder illustrering

Enkoderen er Orthogonal som er 90 grader forskjøvet. Dette betyr at det er to kanaler som det genereres en frekvens på, der den kanalen som ikke er forskjøvet er retningen til enkoderen. 90 grader forskyvning betyr at den kanalen som er forskjøvet er 25% forsinket av perioden til den andre kanalen.

Enkoder funksjonen skal være i stand til å lese/generere frekvens og retning. Denne funksjonen er sensitiv til tidsforsinkelse ved kjøring. For å kunne lese eller genere i fred ble M4 kjernen til Arduino GIGA R1 brukt. Dette gjorde det mulig å få mer nøyaktig lesing og generering av enkoder signalet. Og ved hjelp av RPC kan det mottas og sendes beskjeder til M7-kjernen som er hovedkjernen til kontrolleren. Denne meldingen ble skrevet på dette formatet «$#ID#Direction#Control#Frequency».

ID: Enkoderen sin ID på 1bit med verdien 1-4.

Direction: Enkoder retning som er A eller B på 1 bit, A er framover og B er bakover.

Control: 1 bit verdi som er enten E, R, G. E for Error ved ugyldig verdi, R for Reading for å lese enkoder frekvens og retning, og G for generating der det genereres frekvens og velger retning.

Frequency: Frekvensen som enten blir generert eller lest, denne verdien er på 5 bit som blir 00000-99999 Hz.

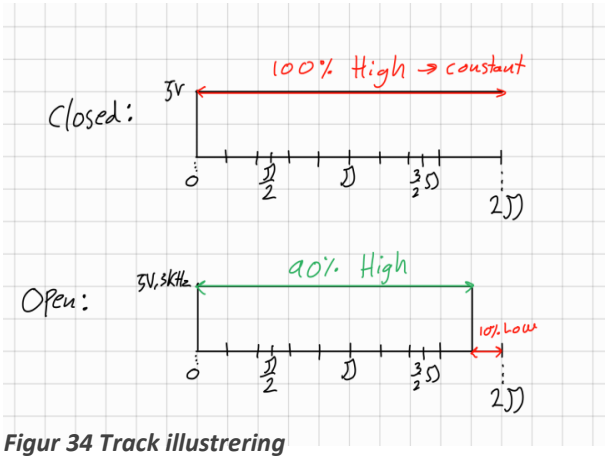
Koden for lesing av enkoder retning og frekvens var ikke lett. Her ble det først testet ved å lese HØY og LAV med flankedeteksjon. Dette virket til en grad, men det var vanskelig å finne retningen til frekvensen. Derfor ble AttachInterrupt funksjonen til Arduino brukt og denne funksjonen aktiveres hver gang inngangen ble HØY. Så ved å ha en Attachinterrupt på inngangen til både A og B -kanalen med en sjekk der det er en frekvensteller som teller hvis bare en av dem er HØY, da kan denne informasjonen brukes til å finne frekvensen og sjekke retningen. Den frekvenstelleren som er høyere enn 0 vil være retningen og verdien til frekvenstelleren er da frekvensen.

Ved generering så blir utgangen satt til HØY og LAV så mange ganger i løpet av et sekund som størrelsen på frekvensen med hjelp av flanke deteksjon. Da leses og skrives det til samme utgang for å sjekke at den faktisk har endret seg. Og da verdien til den kanalen som passer til retningen bli satt til HØY eller LAV først og deretter 25% av perioden etter på vil den andre kanalen bli endret til samme verdi. digitalRead() brukes til å lese og digitalWrite() til å skrive til utgangen.

Lese Eksempel: digitalRead(Utgang) vil gi en boolsk verdi som enten er HØY eller LAV.

Generere eksempel: digitalWrite(Utgang,HIGH) vil sette utgangen til HØY og digitalWrite(Utgang,LOW) vil sette den til LAV.

6.2.6 Track



Figur 34 Track illustrering

Track sensoren som sender en 5V frekvens på 3kHz med en duty cycle på 90% HØY og 10% LAV ved åpen tilstand og konstant 5V ved lukket tilstand.

Så her blir det brukt en funksjon med navnet GetPWM() som sjekker duty cycelen til inngangen og deretter gir oss hva HØY prosenten er. Da er sensoren 90% som åpen tilstand og er i lukket tilstand ved 100%.

Ved Lesing så vil verdien konverteres om til spenningen som blir sendt, altså 90% er 4.5V og 100% er 5V, men inngangen til GIGA er 90% 2.97V og 100% 3.3V sida kontrolleren har en logikk spenning på 0-3.3V.

Hvis det skal genereres et signal så brukes analogWrite() funksjonen som da gir ut en PWM der duty cycle defineres i parameteren til funksjonen.

Eksempel: 90% HØY 10% LAV duty cycle, analogWrite(Utgang,256*0.9). Verdien på utgangen er 0-255 og ved å multiplisere 0.9. Da genereres det en 90% PWM på utgangen.

Det ble dessverre ikke tid til å ta hensyn til frekvensen fordi enkoder koden ble prioritert siden den var mer krevende og track koden måtte ha blitt kjørt sammen med enkoder koden på M4 kjernen for å unngå forsinkelse ved lesing av frekvens.

6.2.7 Logging

For logging ble Adafruit Datalogging shield benyttet siden den ga mulighet for å benytte SD kort som lagringsmedie fordi disse er rimelige og lett tilgjengelig. Den har også en ekstern RTC som kan benyttes.

For tilkobling ble SPI benyttet da denne standarden har større båndbredde en I2C. SPI er full duplex mens I2C kun er halv duplex.

For bruk å kunne bruke SD kortet så benyttes biblioteket «SD.h». Biblioteket «mbed.h» er også nødvendig for å kunne benytte SPI tilkoblingen.

For å kunne opprette en logge fil måtte det kodes et kall mot RTC for å hente ut dato. Denne datomerkingen benyttes for å opprette en loggfil på SD kortet. Det ble besluttet at dd/mm/yyyy var det mest egnede formatet.

En logg oppføring har formatet dd.mm.yyyy 15:12:32,T01,12345,#,

For å generere en logg oppføring kalles RTC for å generere en dato og klokkeslett med sekund.

Sammen med tids stempelet lagres sensor identifikasjon på formatet T01. Identifikatoren er satt sammen med første bit som indikerer hvilken sensor type; T (Temperatur), E (), B (Brake), R (tRack).

Andre bit angir om dette er en reel måling. 0 er en reel avlest verdi. X er indikator for at nå sendes en generert verdi, reel verdi logges samtidig med sin indikator. Y markerer at generert verdi avsluttes.

Tredje bit er sensor nummer i serien 0-9. Dersom det skulle være nødvendig kan dette feltet også benyttes med heksadesimal for serien 0-F.

Det ble kun gjort enkel testing av loggefunksjonen for å verifisere at loggeoppføringen var på rett format. Selve logge funksjonaliteten ble testet og feilsøket i forbindelse med sammenstillingen av programmet.

Det ble benyttet mye tid til å få loggefunksjonen til å fungere som tiltenkt og feilsøkingen ble ytterligere komplisert av andre feil som frem til da var ukjent og ble oppdaget i forbindelse med testing av logging.

6.2.8 Temperatur

Temperatur sensoren på batteriet er en NTC Thermistor av typen NTCLE213E3 [21] og denne sensoren kan måle fra -55 til 150 grader celsius. Ut ifra hvilken temperatur som blir målt vil sensoren generere en spenning som er mellom 0-2.8V, der 0V = -55 grader og 2.8V = 150 grader. Så ved lesing av denne verdien ble det brukt en Bi-Directional Logic Level Converter [8] til å konvertere spenningen fra 0-2.8V til 0-3.3V slik at kontrolleren kunne lese den korrekt. Da ble spenningen sendt til en analog inngang med verdien var da 0-1023. Denne verdien ble konvertert med enkel formel:

$$\text{Temperatur (Celsius)} = \frac{\text{Verdi}}{1024}((-\text{Minimum temperatur}) + (\text{Maksimum temperatur})) + (\text{Minimum Temperatur})$$

Eksempel:

En spenning på ca. 1V på sensor siden vil gi ca. 1.17V på kontrolleren sin side og 365 på den analoge inngangen.

$$\text{Temperatur} = \frac{365}{1024}(-(-55) + 150) + (-55) = 18.07 \text{ grader.}$$

Ved generering ble det antatt at PWM kunne brukes siden gjennomsnittet blir det PWM utgangen er satt til, så ved 50% hadde 1.4V spenning som ville gitt 47.5 grader. Dette var ikke tilfelle og det ble funnet at leseren var altfor god og leste bare HØY/LAV. Det må derfor brukes en DAC, men det blir mer avansert fordi kontrolleren bare har 2 DAC porter og det behovet er på 4 stk. Dette ville da introdusert mer logikk for å kunne sende ut flere signaler. Det måtte også ha blitt verifisert om hovedkortet til roboten kan lese det analoge signalet som PWM.

Ved testing ble det koblet opp et potensiometer for å justere spenninga mellom 0-2.8V og inn til kontrolleren.

6.2.9 Heis, brems og nødstopp

Signalene til heis, brems og nødstopp er av 24V HØY eller 0V LAV. Det ble valgt å bruke en TMUX4053 [7] til å velge mellom generert og originalt signal. Siden spenningen er 0-24V er dette for stor spenning for kontrolleren, derfor brukes TMUX4053 slik at original signalet blir brukt som styringssignal til to innganger med 0V eller 3.3V. Så ved originalt signal som er HØY blir 3.3V valgt og ved LAV blir 0V valgt.

Samme funksjonsmetode blir brukt ved generering av signal, men her blir signalet fra kontrolleren styringssignalet. 5V på styrings inngangen velger 24V og 0V velger 0V. Deretter er det en egen TMUX som blir brukt til å velge mellom det originale eller genererte signalet skal bli sendt videre med et styringssignal fra kontrolleren. Så her blir det totalt 3 TMUX4053 chipper der 1 for lesing, 1 for generering og 1 for valg av originalt signal eller generert signal. Hver TMUX4053 har 3 bryter sett.

6.2.10 Arduino cloud

Arduino IoT Cloud ble valgt til å håndtere kontroll av enheten, siden dette gjør det er meget enkelt å sette opp. Kontrollen i seg selv er ikke tidskritisk og logging foregår på enheten uavhengig av logging i sky funksjonen. Skyen lar deg opprette sky variabler som kan oppdateres via et Dashboard på Arduino sine hjemmesider.

På grunn av sin åpne struktur og gode dokumentasjon har Arduino vært brukt som programmerings verktøy for å lære studenter og elever om programmering. Dette var en av årsakene til at det ble valgt å jobbe med Arduino. I Arduino's IoT Cloud miljø kan det fremstå som man har gått noe vekk i fra denne filosofien til fordel for å utvikle en forretningsmodell basert på abonnering av Cloud tjenesten. Tjenesten i seg selv legger en del begrensninger på bruker på antall ting, variable og kompileringer som er tilgjengelig. Åpenheten i forhold til funksjonalitet ser også ut til å være betydelig mindre gjennomsiktig enn for tidligere modeller.

Eksempelvis er ArdruinoCloud.update() funksjonens virkemåte og hva den utfører fortsatt et mysterium. Dette til tross for at det er en funksjon som skal kalles for hver gjennomkjøring av programmet. Det er brukt tid på å gå igjennom github [18] for å forsøke å finne virkemåten til update uten hell.

Watchdog:
Dette er en timer funksjon som resetter mikrokontrolleren dersom den går over 16 sekunder [19]. Denne funksjonaliteten starter mikrokontrollen på nytt dersom den ikke kommer til ArduinoCloud.uptade() før tiden utløper. Dette har skapt en del problemer i oppstart dersom setup og WiFi tilkobling tar lengre tid. Funksjonen kan deaktiveres ved å legge til false som den andre parameteren i ArduinoCloud.begin().

Arduino Cloud har også biblioteker for å håndtere RTC. Funksjonaliteten overkjører den egen programmerte RTC funksjonaliteten ved kjøring av ArduinoCloud.update().

6.2.11 RPC

Remote Procedure Call (RPC.h) er et bibliotek i Arduino som gjør kommunikasjon mellom de to kjernene M7(Hovedskjermen) og M4(Sekundærkjernen) mulig. Denne kommunikasjonen fungerer ganske likt som seriell kommunikasjon, men bare internt mellom kjernene. Her sendes en melding i form av en streng til sekundærkjernen og bruke den til å utføre et arbeid.

Her blir det brukt kommandoen RPC.begin() for å starte opp sekundærkjernen i hovedprogrammet, og da kan du sende meldingen med RPC.println(Streng) eller lese med RPC.Read(). Ved lesing blir det bare lest en karakter om gangen og det må adderes opp til en streng for å få den fullstendige meldingen.

Det er også lurt å ta i bruk en statement der det blir sjekket om det er kommet en melding, og da kan det brukes if(RPC.Available() == true).

6.2.12 RTC

Real Time Clock (RTC) enheten på Arduino GIGA R1 har et kjent problem hvor den går for fort. For å finne ut hvor mye om ble det utført en serie tester på forskjellige GIGA kort.

Testen ble utført ved å skrive klokkeslett serielt og sammenligne opp mot timestamp det ble gitt i den serielle monitoren i Arduino IDE. Med denne metoden kan start og slutt tidspunkt sammenlignes mellom RTC i GIGA kortet og en test pc.

I testing ble det funnet at de to GIGA kortene som ble testet, går ca. 1-2 sekunder raskere per time enn test maskinen. Av data grunnlaget som er lagt ved som vedlegg er det trukket frem to av testene for å vise hvordan dette så ut.

	Arduino IDE tid:	RTC på GIGA tid:
Start	10:36:39	10:37:00
Stopp	14:45:35	14:46:00
Forløpt tid	04:08:56	04:09:00
Differanse		00:00:04

	Arduino IDE tid:	RTC på GIGA tid:
Start	12:46:11	12:46:16
Stopp	10:00:30	10:00:54
Forløpt tid	21:14:19	21:14:38
Differanse		00:00:19

Den ene testen som over 4 timer går 4 sekunder raskere enn ved starten. Den andre testen som er trukket frem er på 21 timer er 19 sekunder raskere. Alle testen gjennomført viser at den er 1 sekund raskere pr time innenfor spennet for arbeidstiden til modulen. Det ble gjort forsøk ved å koble til ekstern RTC men grunnet utfordringer med å kommunisere med denne ble det forsøkt andre løsninger.

Etter å ha gjort noen undersøkelser så ble det funnet en akseptabel løsning hvor mikrokontrolleren oppdaterer tiden ved hjelp av en Network Time Protocol (NTP) server. Ved å benytte User Datagram Protocol (UDP) til å sende en forespørsel til en NTP server på rett format vil man få tilbake en melding med tidsstempel. Formatet på meldingene er bestemt av NTP Protocol Definition Unit.

Oppdateringen gjøres hver gang modulen startes på ny, som gjør at hver gang modulen lades vil tiden igjen være korrekt.

Det har blitt vurdert at 1 sekund pr time ikke utgjør noen problem for tidtakingen så lenge avviket nullstilles minst en gang i døgnet. Dette er viktig å være klar over denne problemstillingen dersom det skulle benyttes et batteri som kan benyttes over flere døgn eller en annen permanent strømkilde tilkobles.

Klokken baserer seg på antall sekunder siden Unix time 1 januar 1970 i tidssonen Coordinated Universal Time (UTC). I flere tilfeller hvor det under testing var problemer kunne det oppdages ved at logg filen fikk navnet 01011970. En av problemstillingene som skapte dette problemet var at henting av oppdatert tidsstempel ventet på tilkobling til internett. I flere tilfeller skapte dette forvirring for prosessen kunne bli ferdig etter 1-10 program kjøringer og dermed ble det opprettet en korrekt logg fil. Dette problemet har vært fremtredende ved første gangs kjøring etter opplasting av programmet.

«mbed_mktime.h» er inkludert for å gi tilgang til en rekke funksjoner knyttet mot RTC. Dette er et bibliotek som samhandler med Mbed operativ systemet.

6.2.13 Sammenstilt program testing

Etter at alle de ulike programfunksjonene ble produsert ble alle satt sammen til et program. For å sette sammen alle program funksjonene til et program ble det satt inn en og en funksjonsblokk og funksjonstest for å verifisere at koden kunne kjøres. Til å begynne med ble hver funksjonsblokk kombinert med loggefunksjon og verifisert at korrekt hardkodet logg data ble lagret som ønsket. Strukturen ble designet rundt at signal av samme type håndteres i en for-loop i sammenheng med variable lagret i et array. Dette er en vanlig teknikk for å håndtere en identisk operasjon som skal gjennomføres flere ganger med et minimum av kode. I en tidlig fase så er det nødvendig å verifisere at samtlige gjennomkjøringer logger korrekt informasjon.

Det ble opprettet en tidsstyring for gruppen av temperatursensorer. Dette ble gjort for å redusere antall kjøringer av logging for disse sensorene. Temperatur sensorene overvåker polene til batteri og lade kontakt. I en ordinær driftsituasjon vil disse endres sakte, og behovet for logging er vurdert til en gang per 1 til 10 sekund. Det er også enkelt å justere hvor ofte disse skal logges i etterkant, dette gir en større fleksibilitet for de øvrige programfunksjonenes behov for prosessor kraft.

Heis, brems og nødstopp blir overvåket av en flankedeteksjon som logger endring av tilstand. Disse signalene er av eller på og hver gang disse endres vil det utløse logging. Dette er en meget effektiv måte å håndtere disse signalene siden ressursbruken er svært lav. Tidslogging av disse signalene er også mulig, men vil likevel være avhengig av å sjekke for endret tilstand, siden det er lite effektivt å logge at samme signal er til stede gjentatte ganger.

For track sensorene leses disse og logges hver gjennomkjøring. Disse sensorene leser en åpning i banen som roboten kjører i. Denne informasjonen benyttes for å holde kontroll på at det ikke akkumuleres feil i avlesningen av enkoder. Programmet vårt håndterer ikke denne informasjonen og logger kun hver gang en åpning i sporet registreres. Skal denne informasjonen benyttes må oppløsningen på loggingen økes til milli- eller mikrosekund nivå. Logge verdien for denne sensoren er i millivolt for å kunne skille mellom LAV signal og HØY signal. Inn på mikrokontrolleren leses disse som 2.9V og 3.3V og logges som 2900 og 3300. For å kunne gi output fra den leste verdien så må avstand mellom hvert spor i banen legges inn slik at man kan finne hastighet basert på grunnleggende fysikk.

For alle funksjonene ble det nødvendig å opprette en funksjon for generering av utgangs signaler. Disse funksjonene kunne med fordel ha blitt kodet mer generelt for å få en funksjon for generering av signal og med kode for å differensiere hva signal type som skal genereres.

Til slutt ble Arduino Cloud inkludert i skissen. Koden tok over tidligere programmert WiFi funksjonalitet med noe forenkling. Alle Cloud variable deklarert i thingProperties.h overtar nå variablene som ble opprettet tidligere. Det blir umiddelbart problemer da variablene ikke oppdateres som forventet. Dette er beskrevet i kjente feil. Etter mye feilsøking og søk på nettet kommer det frem at cloud variable ikke oppfører seg som forventet når de blir lagt inn i et array.

Gjennom realisering av programmet ved å inkludere en og en funksjon og teste disse stegvis ble feilsøkingen for hver feil så enkel som mulig. Det var likevel flere uventede feil som oppsto sammen med feil som var laget inni i koden, men ikke oppdaget.

6.3 Fysisk testing

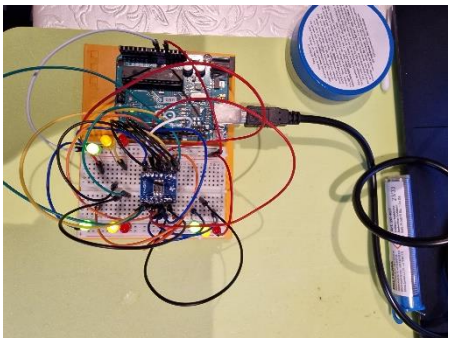
6.3.1 Realisering av signalbehandling

Før oppkobling av koblingsbrettene med de forskjellige signalbehandlingslogikkene kunne bli gjort, måtte komponentene TMUX4053 [7], SN74LVC2G157 [6] og Bi-Directional Logic Level Converter [8] bli loddet og testet. Ved lodding ble det brukt lodde tinn og flux. Ved å tilsette lodde tinn der komponenten skal bli loddet og ved hjelp av flux vil det gå ganske jevnt. Dette er fordi flux gjør slik at tinnet blir tiltrukket til metallet og den legger seg fint over begge elementene.

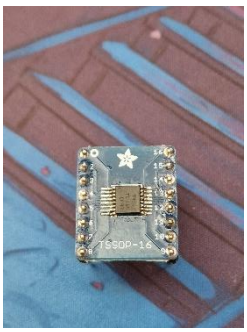
6.3.1.1 Test av TMUX4053

Siden TMUX4053 sine dimensjoner er så små tar det tid å lodde hver enkelt komponent ordentlig for å unngå kontakt mellom alle de 16 beina på breadboardet. Denne lodde jobben repeteres på 10 stykk, der 7 stykk blir brukt i logikken og 3 stykk er i reserve. Deretter blir hver enkelt chip sine 3 bryter sett testet med enkle lys der gul er utgang, grønn er inngang 1 og rød er inngang 2. Ved HØY signal lyser grønn og gul, og ved LAV signal lyser rød og gul.

Dimensjoner: Bredde: 3.36 mm, Lengde: 4.3 mm og Høyde: 1.1 mm.



Figur 36 TMUX4053 Test



Figur 35 TMUX4053 Chip Fremside

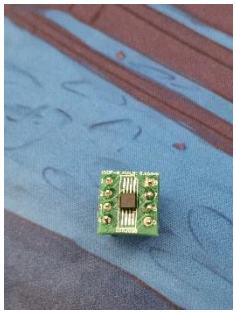
Figur 37 Tmux4053 Chip Side

6.3.1.2 Test av SN74LVC2G157

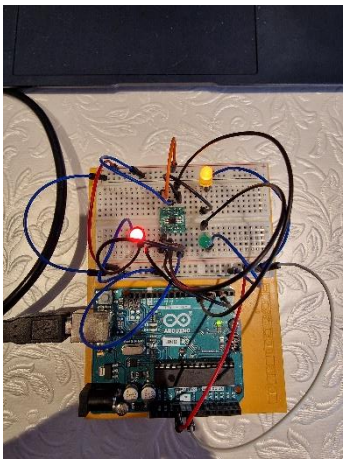
SN74LVC2G157 er enda mindre enn TMUX4053, men med bare 8 bein. Denne tok også mye tid å lodde ordentlig. Det må loddes 16 stykk, der 8 stykk blir brukt i logikken og 8 stykk i reserve. Alle disse 16 blir testet på samme metode som TMUX4053 med enkle lys, der gul er utgang, grønn er inngang 1 og rød er inngang 2. Hvis signalet er HØY lyser grønn og gul, og hvis signalet er LAV lyser rød og gul.

Dimensjoner: Bredde: 3.2 mm, Lengde: 2.1 mm og Høyde: 0.9 mm

Figur 38 SN74LVC2G157 Chip Side



Figur 39 SN74LVC2G157 Chip Fremside



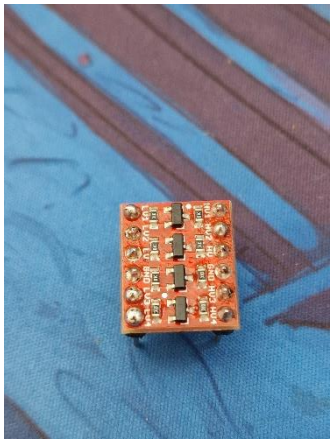
Figur 40 SN74LVC2G157 Test

6.3.1.3 Test av Bi-Directional Logic Level Converter

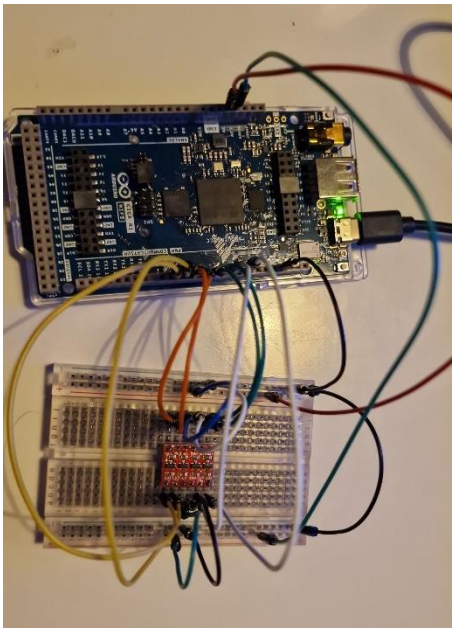
Bi-Directional Logic Level Converter chippen var ferdig loddet på breadboardet, men det måtte loddes koblingspinner. Her måtte 12 pinner loddes på 16 stykk, der 8 blir brukt i signalbehandlingen og 8 stykk er i reserve. Her ble det testet ved hjelp av PWM, der ble det sendt inn en spenning for å konvertere denne til en annen spenning for å verifisere funksjonen.

Dimensjoner: Bredde: 13 mm, Lengde: 17 mm og Høyde: 2 mm

Figur 41 Bi-Directional Logic Level Converter Chip Side



Figur 42 Bi-Directional Logic Level Converter Chip Fremside



Figur 43 Bi-Directional Logic Level Converter Test

6.3.1.4 Signalbehandlingsbrett

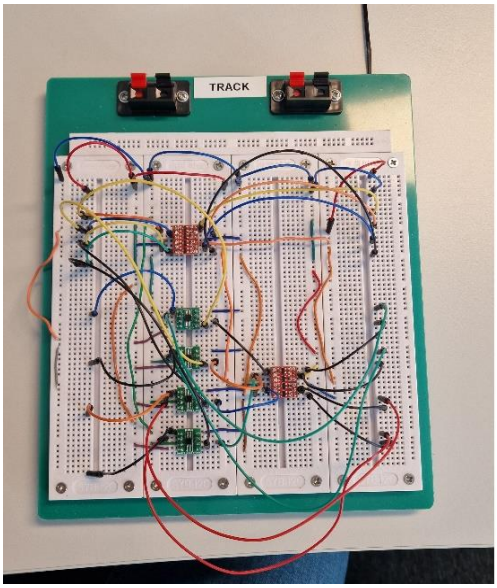
Etter at chippene var ferdig testet og klare til å bli brukt. Ble det laget signalbehandlingsbrett for hver type signalbehandling. Da er det snakk om Track, Temperatur, Enkoder, (Heis, Brems og Nødstopp). Dette ble gjort for å forenkle prosessen ved feilsøking, få god oversikt og fordi det ikke var tid eller mulighet til å lage et PCB kort med alt samlet.

6.3.1.4.1 Track

Track brettet består av 4 stykk av SN74LVC2G157 for valg av originalt track sensor signal eller egendefinert track sensor signal fra mikrokontrolleren og 2 stykk av Bi-Directional Logic Converter for å skalere ned og opp signalet mellom 0-5V hos track sensoren og 0-3.3V hos mikrokontrolleren.

Venstre side er track side med spenning på 0-5V og høyre side er mikrokontroller side med spenning på 0-3.3V.

Se vedlegget [13] AutoStore Test Module Connection Diagram.



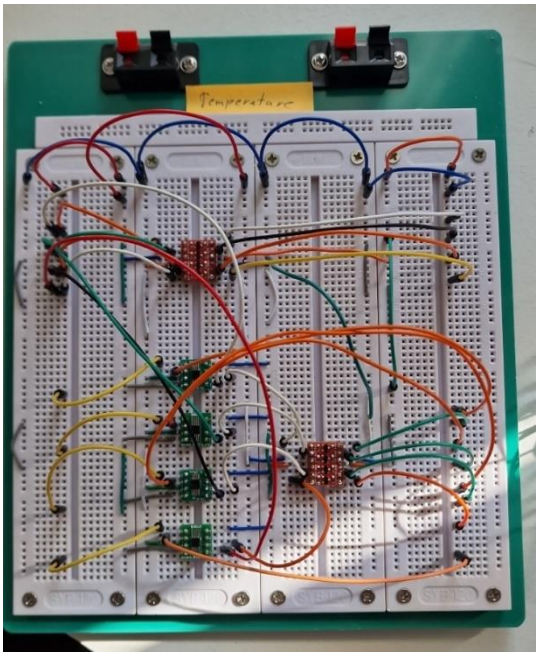
Figur 44 Track Signalbehandlingsbrett

6.3.1.4.2 Temperatur

Temperatur brettet er koblet opp av samme komponentene som Track brett med 4 stykk av SN74LVC2G157 og 2 stykk av Bi-Directional Logic Converter. Eneste forskjellen er at spenningen til signalet er annerledes. Forskjellen er at her blir det skalert opp og ned mellom 0-2.8V hos temperatur sensoren og 0-3.3V hos mikrokontrolleren.

Venstre side er temperatur side med spenning på 0-2.8V og høyre side er mikrokontroller side med spenning på 0-3.3V.

Se vedlegget [13] AutoStore Test Module Connection Diagram.



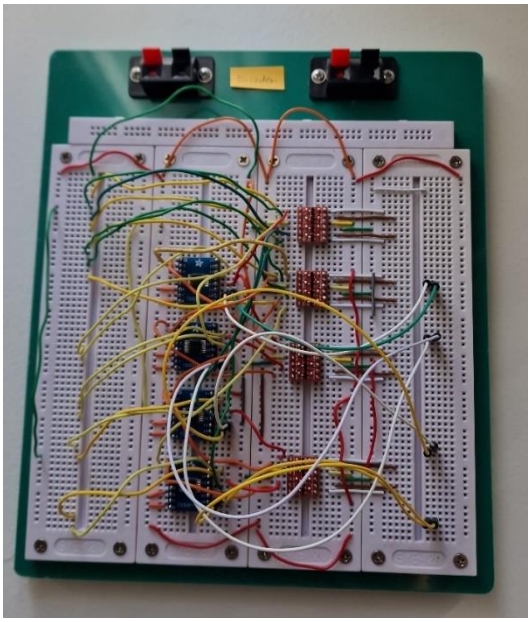
Figur 45 Temperatur Signalbehandlingsbrett

6.3.1.4.3 Enkoder

Enkoder brettet består av 4 stykk av TMUX4053 for å velge mellom originale og egendefinerte enkoder signal og det er 4 stykk av Bi-Directional Logic Converter for å skalere ned og opp signalet mellom 0-5V hos enkoderen og 0-3.3V hos mikrokontrolleren.

Venstre side er enkoder side med spenning på 0-5V og høyre side er mikrokontroller side med spenning på 0-3.3V.

Se vedlegget [13] AutoStore Test Module Connection Diagram.



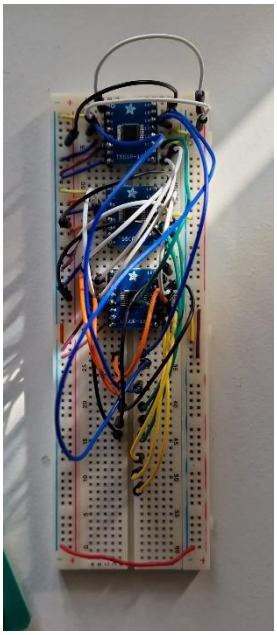
Figur 46 Enkoder Signalbehandlingsbrett

6.3.1.4.4 Heis, Brems og Nødstopp

Heis, brems og Nødstopp brettet består av 3 stykk av TMUX4053 for diverse valg av signaler for både egendefinering og valg mellom originalt eller egendefinert signal. Her brukes det ingen skalering fordi det bare kommer inn signaler som er LAV eller HØY. Her kommer det inn LAV signal på 0V eller HØY signal på 24V. Disse blir konvertert med hjelp av TMUX4053 for å generere samme signalet i spenningsområdet 0-3.3V.

Venstre side er utstyr side med spenning på 0-24V og høyre side er mikrokontroller side med spenning på 0-3.3V.

Se vedlegget [13] AutoStore Test Module Connection Diagram.

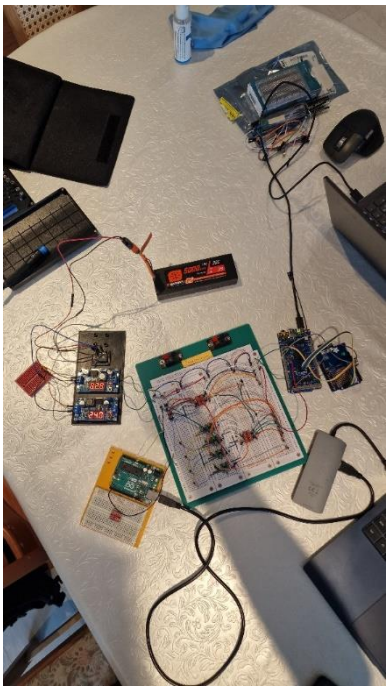


Figur 47 Heis, Brems og Nødstopp Signalbehandlingsbrett

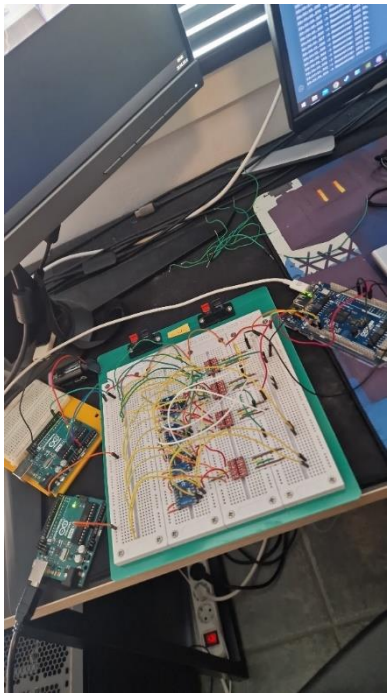
6.3.2 Funksjonstesting med UNO og GIGA

Brettene for signalbehandling ble testet ved å bruke en Arduino UNO R3 til å generere et sensorsignal som skulle være så lik som mulig det fysiske sensor signalet. Her ble det generert både 5V konstant signal og et 0-5V PWM med 90%,10% duty cycle for Track simulering. For Enkoder simulering ble det generert en frekvens mellom 0-30000 Hz med en 25% periode forsinkelse mellom signal kanalene A og B, der den så ikke er forsinket er retningen A = fram og B = bakover. Det ble laget et PWM signal med ønsket gjennomsnitt spenning mellom 0-2.8V for å simulere et analogt signal og deretter dekodet dette for å få temperaturen den simulerte temperaturen. Ved 24V kretsen til Heis, brems og Nødstopp brettet ble det brukt et batteri til å simulere HØY og LAV signal inn på brettet.

Ved alle tilfellene ble det testet å velge mellom originalt eller egendefinert signal og om lesingen av originalt signal funket slik som det skulle.



Figur 48 Test Simulering Track



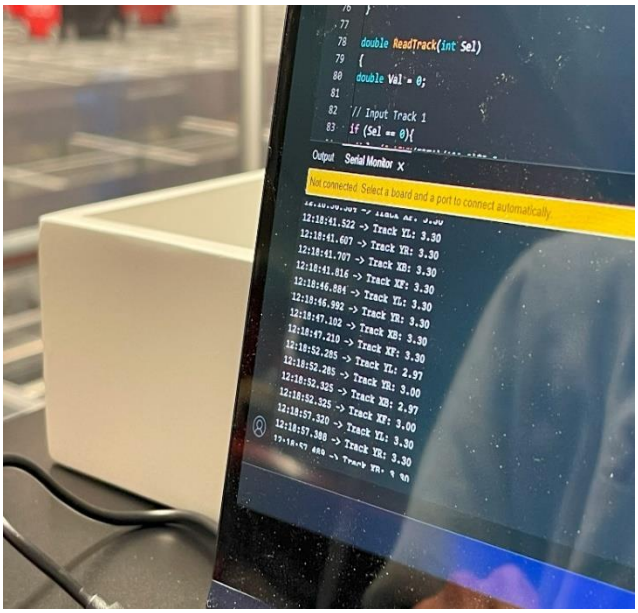
Figur 49 Test Simulering Enkoder

6.3.3 Fysisk Funksjonstesting

Da det var på tide med fysisk test var tiden før innlevering begynt å bli liten og det så kunne bli testet måtte bli testet. Det ble laget tilkoblingskabler med hurtigplugger som passet til robotens tilkoblinger og enkle pinner på motsatt side som ble koblet til signalbehandlingsbrettet. Ved testinga ble det ble fokusert å få testet enkoder og track først, siden disse to er mer de mer kompliserte signalene. Her ble brettet koblet opp og en pc ble brukt til å lese verdiene lest i sanntid. Mikrokontrolleren ble koblet opp med både hovedprogram og loggekort, her ble det komplikasjoner med hovedprogrammet. Koden klarte ikke å kompilere for å laste opp og få sanntidslesing, da ble avgjørelsen det at en enkel lese kode ble brukt.

Figur 50 Tilkoblingskabler

Det ble derfor ikke tid til testing av egendefinert signal for track som ønsket. Heldigvis klarte en enkel test kode å lese signalet. Det ble testet ved å kjøre roboten manuelt opp og ned for å aktivere track sensoren. Siden roboten ikke fikk feilkode så funket signalbehandlingsbrettet som var i passthrough modus ved testing.



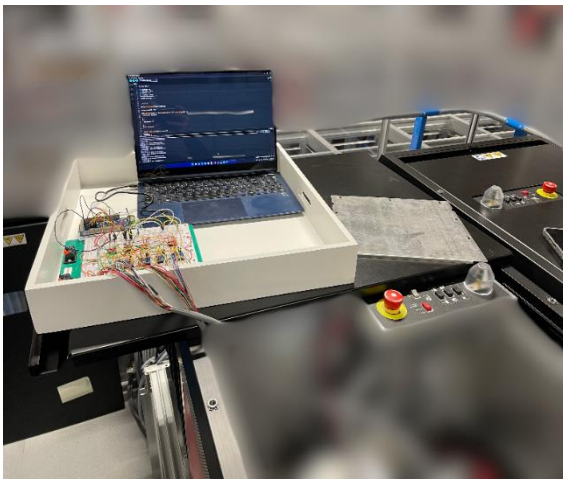
Figur 51 Fysisk Test Track Resultat

Lesing av verdiene for de forskjellige track sensorene er vellykket. Her er 3.3 en betydning på at den er i lukket tilstand som egentlig er konstant 5V signal, men er konvertert til 3.3 for lesing for mikrokontrolleren og ved 2.97 er i åpen tilstand altså egentlig et signal på 0-5V med duty cycelen 90% HØY og 10% LAV.

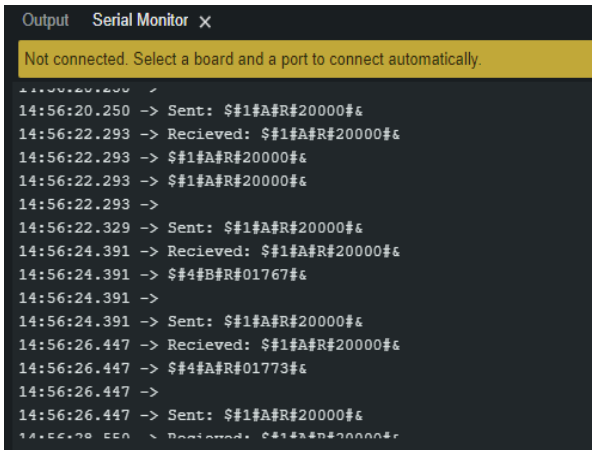
Se vedlegg [20] Track Testprogramkode.

Det ble også brukt en enkel kode for å lese inn signalet fra enkoder til mikrokonrolleren, men her var det også kode som gjorde det mulig for testing av egendefinert signal. Ved testing kom det ingen feilkode på roboten, når det ble generert et eget signal eller ved vanlig passthrough modus. klarte roboten å lese signalet og retningen. Den enkle koden var ikke optimal og printet derfor en ekstra print når den leter etter den aktive enkoderen, men siden det var en enkelt test kode har det ikke mye å si. Det kan bli sett på figur 53 at melding som blir printet er $#4#A#R#01767& og dekodet betyr det at enkoder 4 har retningen A som er framover og modulen er i modus R som betyr reading (passthrough) og at frekvensen er 1767 Hz. Se vedlegg [18] M7 Testprogramkode og vedlegg [19] M4 Programkode

Siden dette var en ganske tidkrevende test arbeid forsvant mye tid i bare å teste enkoder og track. Valget i å måtte stoppe testingen selv om den ikke var fullført ble gjort fordi fokuset måtte bli satt over til rapportskriving for å komme i mål før innleveringsfrist.



Figur 52 Fysisk Test Oppsett



Figur 53 Fysisk Test Enkoder Resultat

6.3.4 Styring fra Arduino Cloud

For å verifisere at styring fra Arduino Cloud er mulig ble det utført tester med en Arduino UNO som signal generator og logikk innkoblet. Test oppsettet besto av en Arduino UNO som genererte et signal inn til logikk kretsen som ble lest av på Arduino GIGA.

GIGA kortet ble koblet mot logikk kretsen med styresignal og generert signal i tillegg til å lese inngangen.

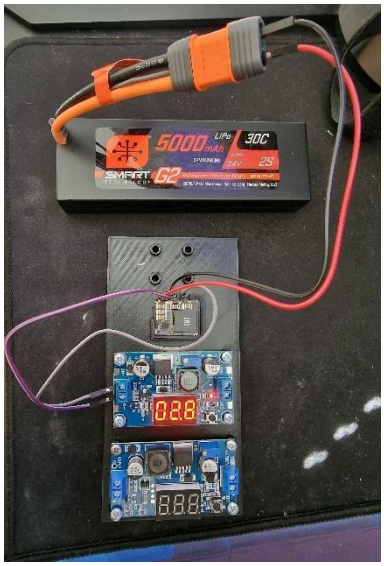
For å teste temperatur kretsen ble det generert et 5V signal fra Arduino UNO som var koblet til et potensiometer for å generere et analogt signal mellom 0 og 5V. Signalet ble også lest av på utgang av logikk kretsen på UNO'en for å kunne verifisere tallverdi lest på GIGA kortet.

Under testingen ble verdien lest av i Cloud Dashboard og sammenlignet mot returnert verdi til UNO. Det ble raskt avdekket en feil lesning hvor GIGA kortet leste 10 over reel verdi, feilen kom fra en tastefeil i koden. Det ble verifisert at avlest verdi på GIGA kortet var innenfor +/- 2 fra verdi avlest på UNO. Generering av signal ble testet og verifisert ved å legge inn en verdi i cloud-variabelen og så aktivere den boolske cloud-variabelen som styrer overstyring. I forbindelse med temperatur målingen vil det være noe treghet på grunn av tidskontrollen for avlesning og tiden det tar per programkjøring. Tidskontrollen vil legge til en forsinkelse mellom 0 og 3 sekunder. Dersom modulen man skal benytte modulen for å generere feil mot temperatursensorene spesifikt vil det være en fordel å øke lesefrekvensen på temperaturene for å unngå forsinkelser.

For å få testet track sensoren, ble UNO koblet inn som generator mot logikk kretsen og genererte et signal ved hjelp av PWM, der 90% PWM ved lesing av åpen tilstand og 100% PWM ved lesing av lukket tilstand. Her ble det oppdaget at det er en feil i logging, hvor alle verdier logges som et heltall. Dette ble raskt omgått ved å legge inn en multiplikasjon med 1000 slik at alle loggede verdier er i millivolt.

6.3.5 Testing av batteri

Det ble gjort litt testing av batteriløsning for å se om den klarte å generere konstant spenning på de forskjellige spenningsområdene. Et batteri ble koblet opp til DFR0753[13] slik at det kom 5V konstant signal ut til DFR0123[12] og DFR0379[11]. Disse klarte da fint å genere 2.8V, 5V og 24V spenninger som skulle brukes.



Figur 54 Batteriløsning

6.4 Kjente feil

Det finnes en rekke feil som ikke har blitt utbedret på grunn av tidsmangel. Her beskrives feil som er kjente og «workarounds» brukt for å unngå problemet.

- **Klokka settes tilbake til UTC tid etter en programkjøring.**
  Ved første gjennomkjøring av programmet vil klokken settes til rett tid hvor det er tatt høyde for tidssone. Det mistenkes at når ArduinoCloud.update() kjøres så setter denne RTC tilbake til UTC uten tidssone. Det må undersøkes å sette lokal tidssone via ArduinoCloud biblioteket.

- **Loggfil opprettes før svar fra NTP server.**
  Loggfil får navnet 01011970.CSV og vil utføre logging i denne filen frem til NTP melding er mottatt. Når NTP melding er mottatt opprettes den rette logge filen og logging foregår normalt. Problemet ser ut å være knyttet mot opplasting av programmet. Når programmet starter etter vellykket opplasting oppstår feilen. Dersom mikrokontrolleren gjøres strømløs, logge filer slettes og startes en gang til vil feilen ikke oppstå.

- **Arduino Cloud støtter ikke array.**
  Dersom en cloud variabel legges til i en array vil ikke denne oppdateres når den endres i cloud. «Workaround» for dette så oppdateres hver cloud variabel inn i hvert relevant array i starten av hver program loop.

- **Float lagres som Integer.**
  I forbindelse med logging av data blir enkelte float verdier lagret direkte som int. Dette ble oppdaget sent i prosessen og ble forsøkt utbedret. I utbedringen ble det gjort så store endringer med feil slik at hele koden måtte rulles tilbake til siste back up som var over 12 timer gammel.

7 Diskusjon

7.1 Fysiske utfordringer

7.1.1 Port

I starten var planen å lage en modul som skulle kunne brukes til flere ulike komponenter i AutoStore sitt system. Her var det planlagt å benytte modulen til å logges signal fra sensorene i portene, B1 og R5. Etter å ha undersøkt nærmere på portene ble det funnet at disse vil være svært tidkrevende å koble til siden de sitter på en roterende karusell som rent fysisk ville vært krevende å koble til. Alle sensorene i hver arm på karusellen sender så til en egen kontroll enhet som sender pakker med informasjon videre. Skal man komme etter kontrollenheten er det nødvendig å pakke opp sendingen og ved sending må disse også pakkes på rett format før videresending. Dette ble vurdert til å være utenfor omfanget av prosjektet og ble derfor utelatt.

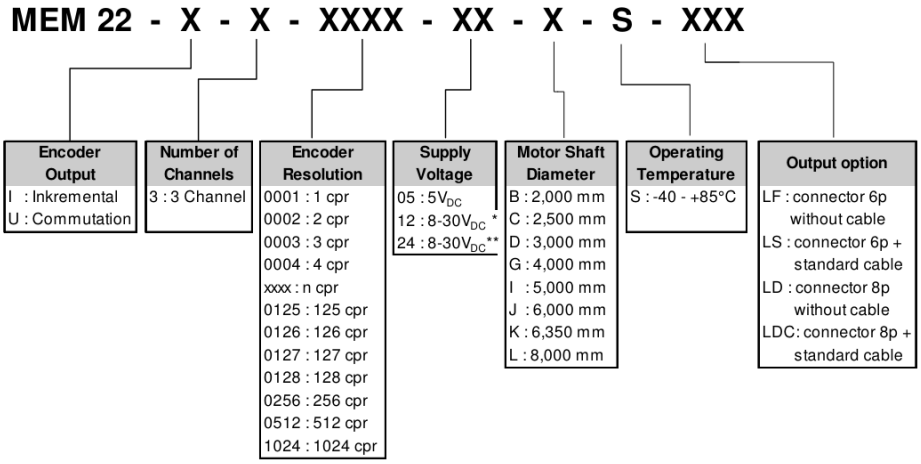
7.1.2 B1 robot

I forbindelse med oppgang av kontakter som sensorene benytter for tilkobling til hovedkortet ble det også oppdaget at på B1 så er tracksensor og enkoder koblet på et eget kontrollkort som videresender til hovedkortet. Denne kontrolleren sitter festet i en ramme som holder hjulene på roboten og det er meget trangt. Etter å ha sjekket opp muligheten for å rute en kabel opp og tilbake fra hovedkortet ble det opplyst om signalets sensitivitet for forstyrrelser. Modulen vil kun være mulig å plassere på toppen av roboten på grunn av plassbehovet. Det ble vurdert om det ville være mulig å benytte en mindre mikrokontroller som for eksempel Arduino NANO montert i hjul rammen, men selv den ville hatt plassproblemer. Det ble derfor bestemt at fokuset skulle holdes spesifikt på R5 serien.

7.2 Utfordring med signal håndtering

7.2.1 Enkoder

Av signalene som skulle bearbeides var enkoderen det mest utfordrende signalet, det er forståelse og behandling av dette signalet som har krevet mest ressurser. I en tidlig fase ble det oversett at enkoderen hadde to kanaler og at disse er fase forskjøvet med 90 grader og den kanalen som kommer først forteller hvilken retning enkoderen roterer. Årsaken til at dette ikke ble oppdaget i starten kommer av at enkoderen ble oppgitt til å være en «MEM22 Y09» ikke har vært mulig å identifisere. MEM22 enkoderen har en nomenklatur i databladet som ble funnet som ikke har mulighet for «Y09»



Figur 55 MEM22

Se vedlegg [22] MEM22 side 9.

Dette førte til at det ble lagt lite vekt på datablad funnet på nettet da disse ikke harmonerte med nummeret gitt.
Senere i prosjektet ble det gjort en motor med enkoder montert til gjort tilgjengelig for testing av enkoder signalet. Når denne ble gjort tilgjengelig ble nummeret på enkoderen sjekket opp mot datablad som ga håndfast informasjon om enkoderen. Mye av dette kunne vært unngått ved å gå grundigere gjennom tilgjengelig informasjon til å begynne med, sammen med å forsøke å innhente mer informasjon fra AutoStore når det ble oppdaget at databladet ikke kunne finnes med den tilgjengelige informasjonen.

Tidlig testing på laboratoriet viste oss at Arduino GIGA kunne produsere et signal som var tilstrekkelig hurtig og var helt avgjørende for å kunne velge GIGA kortet som mikrokontrolleren løsning.

7.2.2 Tracksensor

Ved arbeid for behandling av tracksensor signal ble det misforstått om hvordan denne komponenten fungerer. Det ble trodd at den genererte en PWM på 90% ved lesing av gaps i rullebanen og et konstant 5V signal når det ikke var gaps i rullebanen. Signalet ble konvertert fra området 0-5V til 0-3.3V og da ble verdier lik 2.97V var indikasjon på at det var gaps og ved 3.3V var det lest ingen gaps.

Dette gjorde at det ble tatt litt lett på tracksensoren og fokuset ble rettet mot enkoderen i stedet. Det som skulle blitt gjort er å grundig sjekket opp signal funksjonsmåte slik at den viktige informasjon som var at den hadde en PWM på 90% med en 3 kHz frekvens ikke hadde blitt oversett. På grunn av dette ble ikke signal lesing og generering for tracksensor gjort på korrekt måte.

7.2.3 Temperatursensor

Sent inn i testingen av kode for lesing og generering av temperatursensor signal, ble det funnet at det hadde blitt misforstått om korleis signalet som ble generert oppstod som. Det som skulle være et analogt signal generert mellom spenningen 0-2.8V på sensorside, ble det antatt at en PWM kunne bli brukt for å generere en ønsket spenning mellom 0-3.3V som da ville bli konvertert til 0-2.8V. Siden PWM generer en gjennomsnittspenning ble tenkt at dette kunne brukes slik. Ved 90% PWM blir det generert et gjennomsnitt spenning på 2.97V og det tilsvarer konvertert til 0-2.8V, en gjennomsnittspenning på 2,52V.

Dette virket logisk og det ble ikke grundigere sett på igjen før testing av simulert signal gjennom en Arduino UNO og da ble det sett at det bare ble registrert 0V eller 3.3V. Da ble det funnet ut at dette ikke var en tilstrekkelig måte å generere signalet på og det måtte bli gjort endringer som det ikke var tid til å utføre. Disse endringene var å introdusere logikk for behandling av DAC for å kunne generere et reint analogt signal.

7.3 Arduino Cloud utfordringer

For å styre GIGA kortet trådløst ble det besluttet å forsøke Arduino Cloud siden det fremsto som en god og enkel mulighet for å få testet muligheten. Avgjørelsen å bruke Arduino Cloud ble tatt mot slutten av prosjektet basert på at det ikke var tid igjen til å utvikle en egen applikasjon til formålet.

Det ble allikevel en del utfordringer som følge av den «enkle» løsningen siden en del av de funksjonene som var blitt programmert ble håndtert av Arduino Cloud. Dokumentasjonen og informasjonen tilgjengelig på hvordan Arduino IoT Cloud virker og hvilke innebygde muligheter var ikke tydelig nok. Hadde det vært tydeligere kunne avgjørelsen om å benytte den blitt fattet på et tidligere stadium og noen redundante funksjoner hadde ikke blitt brukt.

De redundante funksjonene har likevel gitt mulighet for læring og økt forståelse på bekostning av fremdrift. Hvordan Arduino GIGA håndterer RTC og WiFi er nå bedre kjent siden funksjonene ble bygd opp først for å så inkludere Arduino Cloud som gjorde disse redundant.

7.4 Proof of Concept

Som proof of concept er kanskje løsningen bra hvor man har en lavkost hoved enhet som kan levere det nødvendige resultatet. Mens ved problemer med enkoder så er det programmert på en sånn måte at man kan ta en annen enhet som utfører denne oppgaven og kan kommunisere med hovedenheten?

Som et utviklings prosjekt ble det lagt en åpen plan fra starten. Målet i utgangspunktet var å utvikle et «minimum viable product» med flere funksjoner enn det som var satt i krav. Etter hvert som tiden gikk og en rekke utfordringer tok tid å løse ble det klart at produktet ikke kom til å bli like velutviklet som først planlagt. Som proof of consept er modulen kun noen få steg i fra å kunne bli en MVP. Disse stegene er mest knyttet til å få modulen i en innpakning som tar mindre plass. Programvaren må også finpusses, men det vil være nyttig å teste om Arduino Cloud er egnet for bruken eller om det er mer hensiktsmessig å kommunisere med en egen applikasjon på en pc. Det må også nevnes at skal man benytte Arduino IoT Cloud så vil det kreve en kommersiell lisens.

8 Konklusjon

8.1 Vår opplevelse av gjennomføringen

Dette prosjektet har vert lærerikt på både godt og vondt. Vi har snublet en del ganger og vi har reist oss opp igjen. Produktet ble ikke 100% slik som vi hadde ønsket og det har vært mange endringer underveis. Heldigvis har gruppen jobbet godt i lag og dratt hverandre opp. Problemene ble løst så godt så vi kunne. Det gikk ganske tregt i starten på grunn av venting på informasjon fra AutoStore. Dokumentasjonen til utstyr var ikke på plass når vi startet etter nyttår på denne oppgaven. Vi kunne ha tenkt oss at dette var på plass før start, men sånn er det ikke alltid.

Dette gjorde at vi kanskje tok litt for lett på problemstillingen i starten og følte oss trygge på at dette skulle gå helt fint. Dette fant vi lenger ut i semesteret at ikke var tilfelle. For det å produsere et produkt som kan lese, logge og generere signal mellom utstyr og hovedkort på produkt hos AutoStore var mye mer komplisert og tidkrevende enn antatt. Vi måtte finne løsninger på signalbehandling for de ulike signalene, men det var tidkrevende på grunn av lodding av komponentene, oppkobling og testing av funksjon. Skriving av koden som skulle utføre arbeidet har vært en merkelig berg- og dalbane. Det har oppstått bugs, utallige timer med feilsøking, frustrasjon over kode som ikke oppfører seg som forventet, misforståelse av hvordan signal og utsyr fungerer. Ikke minst rant tiden vi hadde til utføring fort ut.

Vi føler at det har blitt gjort mye bra og krevende arbeid som vi er stolt over. Det er lagt inn mange timer for å nå så langt som vi har kommet med tanke på at vi har hatt andre fag og jobb utenom. Vi syns gruppen har jobbet godt i lag og er godt fornøyd med hverandres innsats.

8.2 Refleksjon over gjennomføring

For å kunne gjort denne oppgaven bedre skulle vi ha tatt hånd om noen av problemene tidlig i start fasen. Da hadde vi ikke valgt utstyr før alt av informasjon hadde vært til stedet og vi hadde gått enda grundigere inn på hvordan hvert utstyr sin funksjon fungerte og lagd en mer strukturert gjennomføringsplan før noe ble påbegynt. Vi snakket mye om en modul basert struktur og vi prøvde å holde oss til det, men det var ikke lett når det var høyt fokus på å få et ferdig produkt. Dette gjorde at vi ikke fikk fullført testing av produktet. En mer modulær tilnærming til funksjoner som skulle benyttes kunne ha vært bedre, da funksjoner kunne ha blitt fullført i en prioritert rekkefølge. Dette ville også ha spredd en del av problemstillingen med feilsøking som kunne ha gjort den enklere å isolere feil.

For programmeringsdelen av oppgaven har det vært en uvurderlig erfaring å ta med seg videre i forhold til god programmerings praksis og metode. Systematisk feilsøking og gjennomgang av dokumentasjon er kanskje det som bør trekkes frem som noe av de viktigste punktene. Forståelse for kode ved grundig gjennomgang av dokumentasjonen er starten for å bygge programmer som oppfører seg som forventet. Med bedre forståelse av koden vil også det bli enklere å feilsøke da fundamentale feil kan unngås. Systematisk feilsøking vil også gjøre prosessen smidigere og er en viktig egenskap for alle som skal programmere.

8.3 Krav oppnåelse

- **Logge og observere signaler mottatt**
    - Ble fullført, men bare enkoder og track ble testet på fysisk robot.
    - Logging til SD-kort ble gjennomført.

- **Produsere støy på analoge signaler**
    - Ble ikke fullført på grunn av mangel på tid.

- **Forsinke signaler**
    - Ble ikke fullført på grunn av mangel på tid.

- **Overstyre og blokkere signaler**
    - Overstyring ble fullført selv om track og temperatur signal var på feil format.
    - Blokkering av signal er ikke implementert som en egen funksjon, men kan utføres ved å sette overstyringsverdien til minimum eller maksimum.

- **Selvforsynt med strøm**
    - Selvforsynt strømløsning ble fullført med oppladbare batteri og sikring mot spenningsfall ved utlading.

- **Konfigurerbar for ulik funksjonalitet**
    - Ikke fullført fordi de forskjellige produktene som port og B1 robot var for komplekst satt sammen til at det var nok tilstrekkelig tid for gjennomføring. Bare fokus for funksjonalitet med R5 PRO robot.

- **Brukervennlighet; produktet skal kunne konfigureres av testavdelingens personell**
    - Brukervennlighet må videreutvikles, men de funksjonene som ble implementert i Arduino Cloud er operative.

- **Modulen skal kobles på en slik måte at feilkobling ikke medfører kortslutning eller skade på utstyr og personell**
    - Dette ble ikke fullført fordi det var altfor tidkrevende og andre deler be prioritert.

8.4 Tilleggs Funksjoner

- **Skjerm for avlesning av system status og innstillinger**
  Ikke fullført på grunn av mangel på tid.

- **GUI programvare for konfigurering**
  Arduino Cloud ble benyttet som alternativ løsning, men GUI ble ikke fullført på grunn av mangel på tid.

- **Modulen skal kunne konfigureres og sende data trådløst via WiFi gjennom en datamaskin**
  Ikke fullført nok til skikkelig bruk, men kunne brukes til testing.

- **Det skal være mulig å ha flere moduler i bruk samtidig med WiFi tilkobling**
  Ikke fullført på grunn av mangel på tid.

- **Kunne sette opp nødvendige hendelser for at mikrokontrolleren skal utføre endringer på signalet(ene)**
  Ikke fullført på grunn av mangel på tid.

8.5 Videre utvikling av modulen

På grunn av tidsbegrensning vil det være nødvendig å gjøre ytterligere utvikling. Dersom premissene som er lag til grunn og valgene som er gjort i oppgaven blir akseptert vil deler av oppgaven kunne benyttes som den er utført. Strømforsyningen kan tas i bruk slik den er og vil levere de ønskede spenningsnivåene og den eneste ukjente er batterilevetiden i full operasjon. Kabel settene som er produsert kan kobles som planlagt fra sensor inn i logikken og ut av logikken til robotens hovedkort. Logikk håndteringen er implementert og utfører funksjonen sin som tiltenkt. Dette gjør at det er kun mindre endringer i koden som skal være nødvendig for å etablere et «minimum viable product». Derifra er det et fokus på den fysiske oppbyggingen av et kabinett med mulighet for montering som gjenstår for et sluttprodukt. For å oppnå dette må signalføringen som er utviklet produseres som en PCB for å redusere størrelsen.

8.6 Gjenstående arbeid i prosjektet

Dette er en oversikt over de funksjonene som er ufullstendige og de funksjonene som kun er planlagt, men ikke påbegynt

8.6.1 Enkel Interface

Dette var en ekstrafunksjon som inngikk i tidlig plan, men som har utgått på grunn av manglende tid til gjennomføring.

Det var tenkt å bruke et GIGA Display Shield for enkel interface og skjermen var tiltenkt å disse funksjonene i prioritert rekkefølge:

1. Tilkoblingsstatus for WiFi, vise nettverksnavn og signalstyrke
2. Enkel interface for å koble til et trådløst nettverk.
3. Skjermvisning av måleverdier.
4. Innstilling av overstyringsverdier og velger for overstyring
5. Grafisk fremstilling av loggede verdier

Det ble kun gjennomført helt enkel testing av tilkobling og bruk av skjermen og deretter ble det satt til sides til fordel for andre oppgaver.

Det ble også brukt «dashboard» i Arduino Cloud med samme formål som GIGA Display Shield. Det kan vise leste verdier og har en bryter for overstyring av signalet sammen med et felt for å sette overstyringsverdien. Det ble brukt for oss ved testing av generering og lesing av verdier. Her er det muligheter til å sette opp grafisk fremstilling av data.

8.6.2 Kabinett

Det var i tidlig fase planlagt å designe et kabinett som gjør monteringen av modulen enkel og som samtidig beskytter elektronikken. For å kunne gjøre dette må først funksjonaliteten i modulen verifiseres slik at et PCB kan bestilles.

8.6.3 PCB

For å sikre at modulen tar så lite plass som mulig må det designes et PCB som kan holde elektronikken. Ved å montere de integrerte kretsene på et PCB vil det driftssikkerheten øke da kabler ikke lenger kan løsne og kretsene kan komprimeres til en størrelse som er enkel å håndtere. På grunn av leveringstid og potensialet for å måtte gå gjennom flere revisjoner ble gruppen frarådet å bruke tid på dette.

8.6.4 Strømforsyning

Testing av at batteri kapasitet skulle vare i 8 timer i ulike temperaturmiljøer ble ikke gjennomført fordi fokuset var på å få et fullstendig produkt. Det var også en plan å bruke en strømmåler for å måle forbruket til modulen, men dette ble også satt til side på grunn av tid.

Testing av batterikapasitet er naturlig å gjøre i forbindelse med andre tester slik at man får kjennskap til hvor lenge et batteri varer. Den første og enkleste testen som kan foretas er å koble opp modulen for logging og la den kjøre kontinuerlig til den er tom for strøm. Ved å se på loggfilene vil man enkelt finne start og stop.

8.6.5 Mangler i programkode

1. Track sensor generering og lesing av 90% HØY og 10% LAV duty cycle med en 3 kHz signal frekvens, nå er det bare lesing og generering av PWM.
2. Kode for RPC kommunikasjon mellom M7 kjernen og M4 kjernen er ikke ferdig integrert i hovedprogrammet. RPC kode er testet mellom kjernene og det gjenstår kun å integrere denne koden med hovedprogrammet og et kall for å logge returnert verdi til hovedprogrammet.

8.6.6 Temperatur Sensor generering

For generering av temperatur sensor verdier trenger det å bli designet en logikk for generering av analogt signal med hjelp av DAC eller lignende. Grunnen til at det ikke er blitt gjort er fordi det ble sent funnet ut at det ikke var gjort tilstrekkelig og mikrokontrolleren hadde for lite DAC utganger for å kunne generere til 4 sensorer samtidig. Det kan også lages en logikkstyring som gjør at 1 DAC utgang switches mellom 2 ulike sensorer, dette kan enkelt gjøres med de samme komponentene som er brukt på sensor signalene.

9 Referanser

[1] AutoStore, «System» AutoStoresystem.com. Hentet fra: https://www.autostoresystem.com/system (Hentet: 02.02.2024).

[2] AutoStore, The Cube Storage Pioneers™, AutoStoresystem.com. Hentet fra: https://www.autostoresystem.com/company (Hentet: 02.02.2024).

[3] Raspberrypi, Raspberry PI. Hentet fra: https://www.raspberrypi.com/documentation/ (Hentet: 21.05.2024).

[4] RS, ARDUINO GIGA R1 WIFI. Hentet fra: https://no.rs-online.com/web/p/arduino/2623452?searchId=e770ca1b-cc06-4076-b274-67b04eb077f2&gb=s (Hentet: 19.05.2024).

[5] Nucleo, Nucleo. Hentet fra: https://no.mouser.com/new/stmicroelectronics/stm-nucleo-development-boards/ (Hentet: 21.05.2024).

[6] Adafruit, INA169 Analog DC Current Sensor Breakout. Hentet fra: https://www.adafruit.com/product/1164 (Hentet: 20.05.2024).

[7] RS, ARDUINO GIGA Display Shield. Hentet fra: https://no.rs-online.com/web/p/shields-for-arduino/2686967?searchId=907385bf-928f-462e-aa47-d92f8dd64794&gb=s (Hentet: 19.05.2024).

[8] Arduino, ARDUINO UNO R3. Hentet fra: https://store.arduino.cc/products/arduino-uno-rev3 (Hentet: 20.05.2024).

[9] Elefun, Spektrum 7.4V 5000mAh LiPo HC G2 IC3. Hentet fra: https://www.elefun.no/p/prod.aspx?v=51503 (Hentet: 19.05.2024).

[10] Elefun, Spektrum 9.9V 3200mAh LiFe IC3. Hentet fra: https://www.elefun.no/p/prod.aspx?v=56268 (Hentet: 19.05.2024).

[11] Elefun, Spektrum S100 USB-C Smart Charger 100W. Hentet fra: https://www.elefun.no/p/prod.aspx?v=58938 (Hentet: 19.05.2024).

[12] Elefun, Spektrum Kontakt – IC3 Male 2 stk. Hentet fra: https://www.elefun.no/p/prod.aspx?v=56175 (Hentet: 19.05.2024).

[13] Digikey, DC-DC Buck Converter DRF0753. Hentet fra: https://www.digikey.no/no/products/detail/dfrobot/DFR0753/13590872 (Hentet: 19.05.2024).

[14] Digikey, DC-DC Adjustable Buck Converter DRF0379. Hentet fra: https://www.digikey.no/no/products/detail/dfrobot/DFR0379/7087190?s=N4IgTCBcDaICYDMBOAGAzAdgJwgLoF8g (Hentet: 19.05.2024).

[15] Digikey, DC-DC Adjustable Buck Converter DRF0123. Hentet fra: https://www.digikey.no/no/products/detail/dfrobot/DFR0123/6588566?s=N4IgTCBcDaICYDMBOAGAjGAzCAugXyA (Hentet: 19.05.2024).

[16] Draw.io, Gratis diagram program. Hentet fra: https://www.drawio.com/ (Hentet: 19.05.2024).

[17] TinyCAD, Gratis Online diagram. Hentet fra: https://online.tinycad.net/fs/ (Hentet: 19.05.2024).

[18] Github, Arduino IoTCloud. Hentet fra: https://github.com/arduino-libraries/ArduinoIoTCloud

(Hentet: 21.05.2024).

[19] Github, Arduino IoTCloud Pull. Hentet fra: https://github.com/arduino-libraries/ArduinoIoTCloud/pull/235 (Hentet: 21.05.2024).

10 Vedlegg

- [1] INA169 Analog DC Current Sensor Breakout
- [2] Deleliste
- [3] ARDUINO GIGA R1 WIFI
- [4] ARDUINO GIGA R1 WIFI FULL PINOUT
- [5] ARDUINO UNO R3
- [6] SN74LVC2G157
- [7] TMUX405X
- [8] Bi-Directional-Logic-Level-Converter
- [9] Adafruit-data-logger-shield-1141
- [10] ARDUINO GIGA Display Shield
- [11] DFR0379
- [12] DFR0123
- [13] AutoStore Test Module Connection Diagram
- [14] CodeStandardASTestModule
- [15] Test prosedyre for signaler til modulen
- [16] Oscilloskoptest Video
- [17] Sammenstilt Programkode
- [18] M7 Testprogram
- [19] M4 Programkode
- [20] Track Testprogramkode
- [21] NTCLE214E3
- [22] MEM22
- [23] Timeføring
- [24] Forprosjekt Rapport
- [25] Forprosjekt Gant
- [26] Rettleiingsavtale
- [27] Midtveispresentasjon
- [28] AutoStore Testmodul Brukerveiledning

Appendiks A Forkortelser og ordforklaringer

Forkortelse:	Forklaring:
MVP	Minimum Viable Product
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
IC	Integrated circuit
RTC	Real Time Clock
RPC	Remote Procedure Calls
FAT16	File Allocation Table (et filsystem basert på 16bits arkitektur)
FAT32	File Allocation Table (et filsystem basert på 32bits arkitektur)
I2C eller I^2C	(Inter-Integrated Circuit) er en halv duplex synkron seriell kommunikasjons bus.
SPI	Serial Peripheral Interface er en full duplex synkron seriell kommunikasjons bus
SD	Sand Disk som er en standard for flash minnekort
IDE	Integrated Development Environment
NTP	Network Time Protocol
UDP	User Datagram Protocol
UTC	Coordinated Universal Time (UTC) som er arvtageren etter Greenwich Mean Time (GMT)

Ord:	Ordforklaring:
Workaround	En metode for å overkomme et problem eller en begrensning i et program eller system.
Firmware	Firmware er en programvare som gir kontroll og styring over spesifikk hardware.
Mbed	Mbed er en utviklingsplattform og operativsystem for utstyr med 32 bits ARM Cortex-M mikrokontrollere.
Op-amp	Operational amplifier, bruke til å steppe opp spenninger.

Appendiks B Prosjektledelse og styring

B.1 Prosjektorganisasjon

Planen i starten var at prosjektleder rollen skulle rulleres på gjennom hele prosjektet. Prosjektleder jobben ble først å fremst planlagt å dreie seg om å være møteleder. Siden alle tre har jobbet sammen mye tidligere ble en flat leder stil vurdert som en akseptabel løsning. Dette skjedde fordi alle i gruppen tør utale seg og er god til å motivere hverandre. Rent praktisk har mye av arbeidet dreiet seg rundt felles arbeids økter i første halvdel av prosjektet, hvor gruppen har vært samlet for å jobbe med våre ulike oppgaver. Gruppen har også logget timeføring av utført arbeid gjennom bacheloroppgaven. Se vedlegg [23] Timeføring.

Oppgavefordeling har skjedd ved at noen oppgaver har blitt delt ut mens andre oppgaver har blitt tatt av den enkelte. Hardware har vært Martins hovedansvar. William har gjort fysisk oppkobling av logikk og flere av programfunksjonene med hovedfokus på enkoder. Kim har stått for sammenstilling av koden med feilsøking og testing av den. Utenom har alle vert med på fysisk testing av komponentene på AutoStore.

B.2 Prosjektform

Det har blitt brukt en form for hybrid scrum der kanban metoden har blitt benyttet. Her har en møteleder blitt rullert, men en utpreget leder har ikke blitt valgt fordi det ikke har vært nødvendig fordi samarbeidet i gruppa har gått godt uten. Det at arbeidsflyt og samarbeid har vert bra har gjort at det ikke har vert bruk for hyppige møter og dette er årsak til at en ren scrum struktur ikke har blitt valgt.

B.3 Fremdriftsplan

Det har ikke blitt prioritert å bruke tid på å oppdatere fremdriftsplanen for prosjektet siden utfordringene har endret seg underveis. Det måtte ha blitt laget en helt ny fremdriftsplan som ikke vil vært sann siden nye utfordringer har dukket opp og nye problemstillinger måtte løses. Derfor refereres det bare til forprosjekt fremdriftsplan som er planen som var ønsket.

Forprosjektet ble levert 5. februar, midtveispresentasjonen ble gjennomført 21. mars, og presentasjonen av det endelige produktet for AutoStore fant sted 16. mai. Det blir muntligeksamen av oppgaven som blir siste presentasjonen vil bli holdt 13. juni på Høgskulen på Vestlandet. Se vedlegg [25] Gantt forprosjekt, vedlegg [24] forprosjektrapport, vedlegg [26] Rettleiingsavtale og vedlegg [27] midtveispresentasjon.

Appendiks C Brukerdokumentasjon

Enkel forklaring på hvor ting ligger i brukerdokumentasjon vedleggene.

C.1 Brukerveiledning

Side 1 er Innholdsliste.

Side 2 er Førbruks kontroll.

Side 3 er Montering.

Side 4 er Oppstart.

Side 5 og 6 er Funksjoner.

Se vedlegg [28] AutoStore Testmodul Brukerveiledning.

C.2 Koblingsskjema

Side 1 er oversikt og koblingskart over strømforsyning.

Side 2 er koblingskart over Heis, Brems og Nødstopp signalbehandling og styring.

Side 3 er koblingskart over Track sensor signalbehandling og styring.

Side 4 er koblingskart over Enkoder signalbehandling og styring.

Side 5 er koblingskart over Temperatur sensor signalbehandling og styring.

Side 6 er koblingskart over Loggekort.

Se vedlegg [13] AutoStore Test Module Connection Diagram.