



UiT Norges arktiske universitet

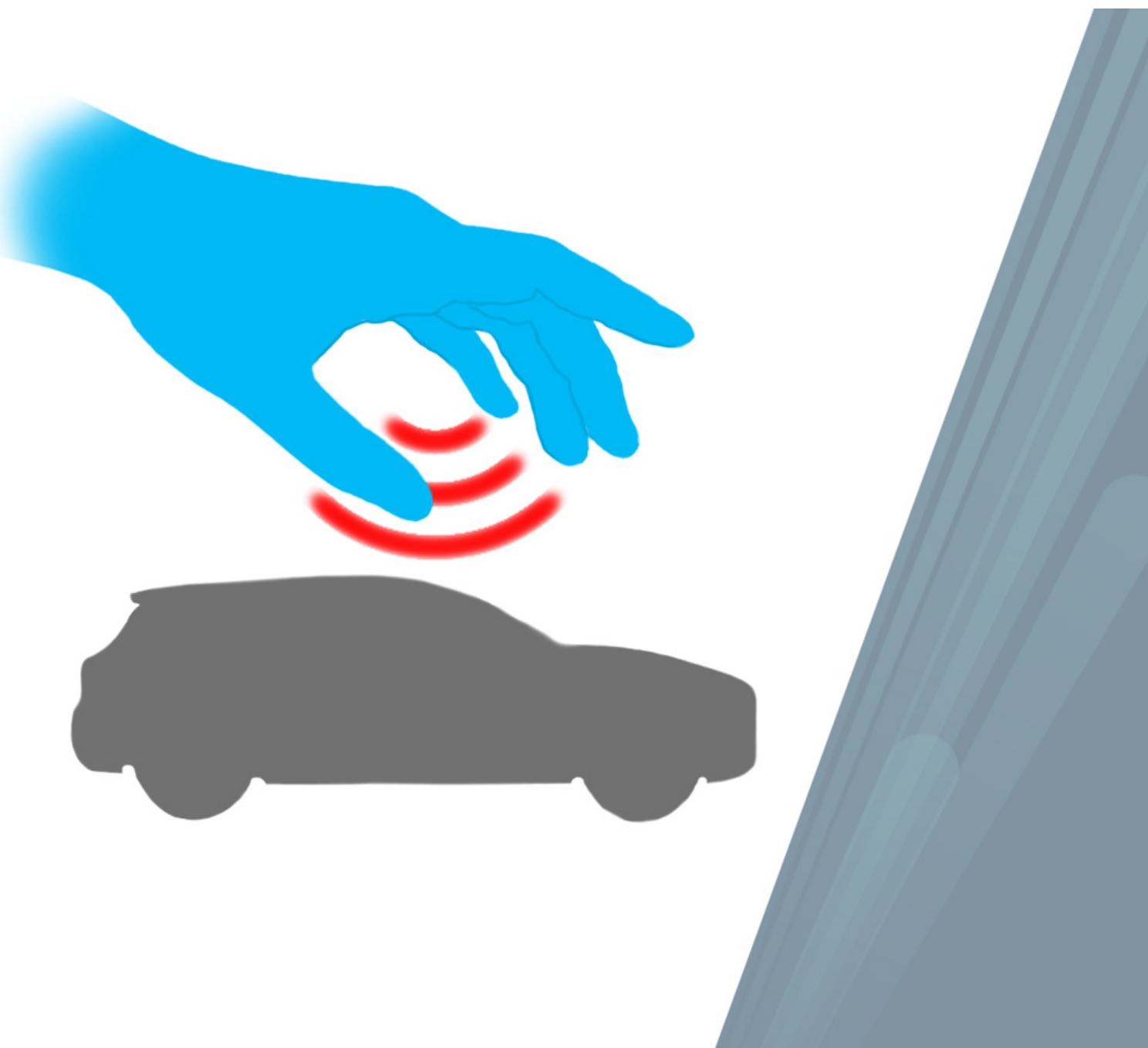
Fakultet for ingeniørvitenskap og teknologi

Bacheloroppgave: Prosjektrapport

Hands-off ROV

Sivert Rabben Aam Øystein Falkeid
saa070@uit.no ofa018@uit.no

Automasjon, kull H2020



Studieprogram:	År:
Bachelor i ingeniørfag - Automasjon	2020
Tittel: Hands-off ROV	Dato: 15.05.2023 Gradering: _____
Forfattere: Sivert R. Aam Øystein Falkeid	Antall sider: 56
Fortrolighet: Åpen	
Veileder:	Puneet Sharma
Oppdragsgiver: UiT	Oppdagsgivers kontaktperson: Puneet Sharma

Innhold

1 Sammendrag	ix
2 Forord	x
3 Innledning	1
3.1 Om oppdragsgiver	1
3.2 Bakgrunn for oppgaven	1
3.3 Problemstilling	1
4 Metode	3
4.1 Utviklingsmetode	3
4.2 Evolusjoner metode	3
4.3 Ansvarsfordeling	3
5 Teori	4
5.1 PID-Regulator	4
5.1.1 Stegrespons	4
5.1.2 Ziegler Nicols	5
5.1.3 SIMC	6
5.1.4 Bode-Nyquist kriteriet	6
5.1.5 Tidsdisket	7
5.2 ROV	7
5.3 H-bru	7
6 Teknologi og utstyr	9
6.1 Lego Mindstorms EV3	9
6.2 Raspberry pi	9
6.3 ESP32	10
6.4 Kretskort	10
6.5 Leap Motion Controller	11
6.5.1 Oppbygning	11
6.5.2 Håndposisjoner	12
6.5.3 Interaksjonsboks	13
6.5.4 UltraLeap Gemini SDK	14
6.6 Datamaskin	14
6.7 AVR128DB48	14
6.8 XBee	15
6.8.1 Programmering av XBee	15
6.9 Logic Level Converter	15
6.10 L298N Motordriver	16
6.11 EV3 L-Servo Motor	16
7 Gjennomføring	18
7.1 Fremdriftsplan	18
7.2 Overblikk av systemet	19
7.2.1 Fartøyet	19
7.2.2 Systemet i sin helhet	19

7.2.3	Fjernstyrt fartøy	20
7.3	Prototype	21
7.3.1	LEGO Mindstorms	21
7.3.2	ESP32	21
7.3.3	Raspberry PI	21
7.4	ROV	22
7.4.1	PCB	23
7.4.2	Retningsberegning	26
7.4.3	Lengdeberegning	27
7.4.4	AVR128DB48	28
7.4.5	Regulator	33
7.4.6	XBee programmering	40
7.4.7	L298N Motor Driver	40
7.5	Styresystem	41
7.5.1	Brukergrensesnitt	41
7.5.2	Koding av HMI	44
7.5.3	Python	47
7.5.4	Leap Motion Controller	48
8	Resultater Og Diskusjon	49
8.1	LeapMotion testing	49
8.2	Testing og utregning	49
8.3	XBee kommunikasjon	51
8.4	Signalproblematikk på Motorer	52
8.5	Brukergrensesnitt	52
9	Miljøkonsekvensanalyse	55
10	Konklusjon	56
10.1	Videre arbeid	56
11	Litteraturliste	i
12	Vedlegg	iv
A	Prosjektfleksjon	v
B	Forenkling av matematiske formler	viii
C	PID Beegninger	xii

Figurliste

5.1	Tilbakekoblet regulator	4
5.2	H-bru [33]	8
6.1	Lego Mindstorms EV3 Intelligent Brick	9
6.2	Raspberry PI	9
6.3	ESP32	10
6.4	Leap Motion Controller	11
6.5	Leap Motion Controller koordinatsystem [31]	11
6.6	Leap Motion Hånd [2]	12
6.7	Interaksjonsboks [31]	13
6.8	AVR128DB48 curiosity nano	14
6.9	XBee	15
6.10	Logic Level Converter	15
6.11	L298N motor driver	16
6.12	LEGO 45502 EV3 Large Servo Motor	16
6.13	Takometerpuls av motor ved rotasjon	17
7.1	Fremdriftsplan med tidsskjema og frister	18
7.2	Systemets oppbygning	19
7.3	Flytskjema for kommunikasjon	19
7.4	Flytskjema over intern kommunikasjon i ROV	20
7.5	Fartøyet under første funksjonstest	23
7.6	PCB 3D model	24
7.7	PCB tegning	24
7.8	PCB-baner	25
7.9	Rotasjon av fartøy	26
7.10	USART3_init	29
7.11	TCA Motor driver	30
7.12	TCB takometermåling	31
7.13	Flytskjema over funksjonalitet av AVR128DB48	32
7.14	Stegresponsen til systemet i åpen sløyfe	33
7.15	Bodediagram over systemet uten regulator h_u og systemet med PI-regulator h_r i åpen sløyfe	34
7.16	Stegrespons for regulert system kontinuerlig og diskontinuerlig	35
7.17	Stegrespons for regulert system med kontinuerlig h_u og diskontinuerlig h_r	36
7.18	Simulink kode	37
7.19	Regnfeil ved divisjon	39
7.20	PWM motor styring	41
7.21	Fartøy i brukergrensesnitt	41
7.22	Innstillingsfane i HMI	42
7.23	HMI controller	43
7.24	Lerretets koordinatsystem	44
7.25	Fartøyets oppbygning i HMI	45
7.26	Polygonene som danner fartspiler	45
7.27	Polygonet som danner en kompasspil	46
7.28	Rotering av kompasspil	46
7.29	Python main	47
7.30	ROV drive	48

9.1 FN's Bærekraftsmål	55
----------------------------------	----

Formelliste

5.1	PID SIMC	6
7.1	Rotasjon lego car beregniong1	27
7.2	Rotasjon lego car beregniong3	27
7.3	Lengde	27
7.4	hastighet	31
7.5	PI-regulator variabler	33
7.6	Systemets transferfunksjon hu	33
7.7	PI-regulator i s plan	34
7.8	PI-regulator i z plan	38
7.9	Regulerings systemet i z plan	38
7.10	PI-regulator i tids plan	38
7.11	hastighetsmåling konvertert	39
7.12	Regnefeil ved divisjon	39

Tabelliste

1	PID Ziegler-Nichols	5
2	Tidskonstant SIMC	6
3	Avstandstest	50
4	Rotasjonstest	51

1 Sammendrag

Denne oppgaven omhandler konseptet å benytte kontaktløs styring til å kontrollere et fjernstyrte fartøy. Vi har fokusert på å utvikle et demonstrasjonsfartøy for å vise hvordan informasjon fra en Leap Motion Controller kan benyttes til styring.

Det er tidligere blitt satt sammen en rekke forskjellige demonstrasjoner som viser dette konseptet. Vi har bygget på disse og forsøkt å forbedre konseptet slik at bruken blir mer intuitiv og brukervennlig [1, 16, 20].

Gruppen har designet et styresystem som er mer generelt og som enkelt kan implementeres i forskjellige fartøy. Dette fordi vi gjennom planleggingsfasen ønsket å styre en drone, men valgte å starte med en bil. Valget av bilen kom fra to hovedpunkter. Det ene er enkelheten av å styring i to dimensjoner. Det andre var tilgjengeligheten av fartøyet. Gruppen har fokusert på brukergrensesnittet og fartøyet for å skape et sømløst og intuitivt styresystem med toveis kommunikasjon.

Stikkord:

- kontaktløs styring
- fjernstyrt fartøy
- brukergrensesnitt

2 Forord

Denne rapporten er skrevet våren 2023 av Sivert Rabben Aam og Øystein Falkeid i forbindelse med bacheloroppgaven i automasjon. Bacheloroppgavens omfang er 20 studiepoeng og representerer avslutningen på tre år med studier innen automasjonsretningen ved UiT - Norges arktiske universitet.

Rapporten henvender seg til leserne med interesse for automasjon og kybernetikk.

Vi ønsker å takke veileder Puneet Sharma for mye god hjelp. Vi ønsker også å takke Per Anton Olsen for veiledning innen programmering av AVR128DB48, og Anne Mai Ersdal for veiledning innen PID-regulering.

Vi ønsker også å takke medhjelpere som har støttet gruppen i en travl hverdag. Ønsker spesielt å takke Mariel Holmen.

En av medlemmene i gruppen har dokumentert dysleksi, og ber dermed leseren om skjønn med tanke på rettskriving.

3 Innledning

3.1 Om oppdragsgiver

UiT - Norges Arktiske Universitet er et av norges største universitet med nesten 18 000 studenter og rundt 3800 ansatte. Universitetet ble grunnlagt i 1958 og er dermed over 50 år. Universitetet er organisert i totalt syv fakulteter, hvor fakultet for ingeniørvitenskap og teknologi er et av dem. Under dette fakultetet ligger institutt for automasjon og prosessteknologi, som er oppdragsgiver for dette prosjektet.

3.2 Bakgrunn for oppgaven

Universitetet i Tromsø har lenge hatt enheter kalt Leap Motion Controller liggedes til opplæringsformål. Enhetene har ikke blitt brukt så mye som først tiltenkt, og det ble derfor lyst ut et forslag om å skrive bacheloroppgave om bruk av Leap Motion Controller til styring av et radiostyrt fartøy. Oppgaven vakte umiddelbart interesse for studentene i denne gruppen. Da universitetet ville at hovedfokuset på oppgaven skulle være Leap Motion Controller og styringssystemet ut ifra denne, var ikke fartøyet det viktigste. Det ble spekulert i å bruke en drone eller en ubåt, men det ble konkludert med at en bil ga nok kompleksitet for denne oppgaven.

UiT vil gjennom denne oppgaven kunne få mer kunnskap om hvordan Leap Motion Controller kan brukes sammen med forskjellige programmeringsspråk, og hvordan informasjonen kan bli benyttet til å lage et styringssystem. Økt kunnskap kan eksempelvis gi fremtidige forelesere en pil i riktig retning når de legger opp nytt stoff i undervisningsemner.

3.3 Problemstilling

Problemstillingen er å demonstrere et styresystem basert på dataen fra en Leap Motion Controller, og benytte dette systemet til styring av et fjernstyrt fartøy.

De grunnleggende kravene som ble satt i starten av prosjektet var som følger:

1. Trådløs kommunikasjon mellom kontrollenhet og fartøy.
2. Nøyaktig avlesning av håndens posisjon og bevegelse.
3. Uventede brå håndbevegelser eller manglende data skal ikke føre til systemsvikt.
4. Grunnleggende brukergrensesnitt som forteller brukeren om systemdata.

Etter første fase ble det bestemt å videreutvikle funksjonaliteten til systemet. Hovedmålet her er å gjøre systemet mer brukervennlig og intuitivt.

Kravene i fase 2 var som følger:

1. Toveiskommunikasjon til innhenting av data fra fartøyet.
2. Visualisering og loggføring av fartøyets bevegelse.
3. Grafisk brukergrensesnitt med behagelige figurer.
4. Hastighetsregulering.

4 Metode

4.1 Utviklingsmetode

Det er benyttet eksisterende komponenter og sett på eksempler av andre som har demonstrert konseptet tidligere. Det blir benyttet en utviklingsmetode som blir beskrevet som Inkrementell innovasjon [13, 30].

Det er derimot benyttet en mer evolusjoner tilnærming hvor et trinn implementeres og raffineres før neste trin startes. Et eksempel på dette er vår bruk av ESP32 og Raspberry PI til å designe et grunnlag for hvordan vi ønsker at det endelige produktet skal fungere for så å skifte til AVR128DB48.

Det er fremdeles rom for videre utvikling, og oppgaven blir derfor ansett som en liten del av en større innovasjon innen bevegelsesdetektering og berøringsfri styring.

4.2 Evolusjoner metode

Etter oppstartsfasen hadde vi et fungerende produkt som tilfredsstilte de grunnleggende kravene. Vi benyttet en evolusjoner metode for å implementere mer funksjonalitet og for å øke kvaliteten på produktet, og dermed tilfredsstille nye krav [29].

4.3 Ansvarsfordeling

- Sivert R. Aam
 - Programmering av styresystem
 - Design av brukergrensesnitt
 - Programmering av XBee
 - Oppkobling av kommunikasjon mellom fartøy og styresystem
- Øystein Falkeid
 - Mekanisk design og montering på fartøyet
 - Kretskortdesign
 - Elektroteknisk plan og oppkobling
 - Programmering av AVR128DB48
 - Design og testing av regulator

Ikke alle arbeidsoppgavene var like krevende og gruppemedlemmene delte dermed på arbeidsmengden. Selv med individuelle ansvarsområder hadde begge gruppemedlemmene gode innspill til hverandre. Dette førte til et tettere samarbeid og bedre forståelse for det totale systemet.

5 Teori

I dette kapittelet presenteres generell teori som er relevant for gjennomføringen av prosjektet.

5.1 PID-Regulator

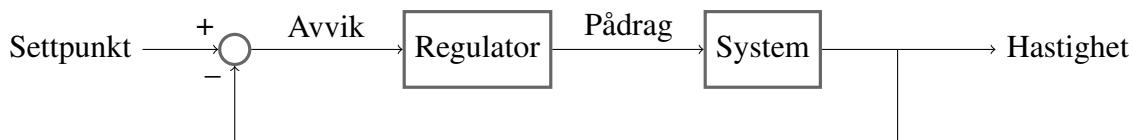
Regulatorer finnes i nesten alle produkter. Noen eksempler på regulatorer som passer inn i denne oppgaven er "cruise control" for hastighetsstyring av biler, og "flight controller" for posisjons- og hastighetsstyring av droner. Disse regulatorene benyttes for å oppnå en mer autonom styring.

Vi kommer til å se grundigere på noen av teoremene beskrevet i boken Reguleringsteknikk[14]. Det finnes 3 hovedkomponenter i en regulator. En tilbakekoblet regulator måler systemet som skal reguleres og bruker denne målingen til å finne avviket fra den ønskede verdien. Dette avviket er essensielt for de fleste formene for regulering.

P-ledd Proposjonal-leddet gir en endring i pådraget proporsjonalt med avviket. Vi ønsker ofte å øke pådraget med en proposjonal verdi fra avviket. Dette kan implementeres med å multiplisere avviket med en konstant.

I-leddet Integral-leddet gir et pådrag som tilsvarer integralet av avviket over hele systemets levetid multiplisert med en konstant. Dette vil føre til at et lite avvik over lang tid skaper stort pådrag, som betyr at det stasjonære avviket blir null. Negative sider med dette er at et stasjonert avvik som ikke kan reguleres bort vil føre til et I-ledd som bygger seg opp i det uendelige.

D-leddet Derivat-leddet gir et pådrag tilsvarende endringen i avvik multiplisert med en konstant. Dette vil føre til at raske og kraftige endringer kan kompanseres for med store og hurtige endringer i pådrag. Store endringer i pådraget kan føre til støy eller så stort pådrag at utstyr kan ødelegges.



Figur 5.1: Tilbakekoblet regulator

Figur 5.1 viser regulering av hastighet. For en bil vil denne være hastigheten frem eller bak, mens en drone vil ha en hastighet for hver dimensjon, totalt tre. I en drone finnes det en tilbakekoblet regulator per akse. Hastigheten måles og sammenlignes med ønsket settpunkt. Systemet er hele fartøyets reaksjon på pådraget. Med reaksjon menes treghetsmomentet til fartøyet, vinkeltregheten til motorer, pådragsorganer og friksjon i akslinger.

5.1.1 Stegrespons

Stegresponsen til et system er en grafisk fremstilling av systemets målte verdi ved et steg på pådraget[14]. Mange systemer tåler ikke store steg, og det kan da benyttes mindre steg. Stegresponsen forteller hvordan det kan forventes at systemet kan reguleres.

Tidskonstanten er den tiden det tar for et system å oppnå 63% av steget i stegresponsen. Denne konstanten forteller hurtigheten til systemet. Det er vanligvis ikke mulig å regulere systemer raskere en tidskonstanten fordi regulatoren er avhengig av systemets respons. Det er allikevel mulig å designe aggressive regulatorer som benytter store pådrag til å senke responstiden til et system.

Stegresponsen er et viktig hjelpemiddel fordi tidskonstanten forteller ca hvor ω_c ligger. Og fordi formen forteller oss omrentlig hvordan den matematiske modellen for systemet ser ut. Hvis systemet har store oversvingninger kan dette komme av kraftig forsterkning eller derivasjonsledd. Et systemet overdempet derimot, kan dette tyde på lav forsterkning med integrator. Det konstante avviket etter steget forteller om det finnes I-ledd og hvis ikke hvor stor forsterkningen er.

Forsterkningen til systemet er det samme som hvor mange prosent av pådraget som representeres i steget. Et eksempel er et pådrag med et steg fra 0 til 1, hvor systemets steg er overdempet og går fra 0 til 0.85 over 3ms. Ut ifra informasjonen vet vi at forsterkningen er $K = 0.85$, at systemet sannsynligvis ikke har derivasjonsledd og at det ikke er integrasjonsledd. Tidskonstanten er antakelig mellom 2ms og 3ms. Fra eksempelet kan vi sette opp standard PI-regulator med $K_p = 1$ og $T_i = 0.001$. Denne regulatoren er ikke optimal og det er ikke mulig å garantere stabiliteten, men det er et enkelt utgangspunkt og gir en grunnleggende forståelse av hvordan systemet oppfører seg.

5.1.2 Ziegler Nicols

Ziegler Nicols metode består av å beregne regulatoren ut ifra observasjoner fra virkeligheten, og benytte en rekke huskeregler for å finne en tilsvarende optimal regulator[14]. Vi kan på denne måten sette opp en regulator gjennom fysiske observasjoner.

Systemet reguleres ved kun P-regulator og K_p settes høyere frem til stående svingninger oppnås. Det er her viktig å påse at systemets mekaniske komponenter tåler stående svingninger. Det er også viktig at K_p økes gradvis slik at svingningene ikke blir ukontrollert store. Et system med stående svingninger (marginalt stabilt system) har spesifikke forhold mellom stabilitetsmarginene. Dermed kan reguleringsverdier beregnes slik at stabilitetsmarginene blir store nok til å oppnå et stabilt system. Store stabilitetsmarginer fører ofte til et langsomt system, og regulatoren beregnes med så små stabilitetsmarginer som mulig.

Noter ned K_p verdien som gir stående svingninger K_{pK} , og periodetiden til svingingene T_K . Beregn PID verdier i henhold til tabell 1 og sett inn i en paralell regulator. Regulatoren vil nå inneholde verdier som er beregnet rundt kryssfrekvensen til systemet i åpen sløyfe.

Regulator	K_p	T_i	T_d
P	$0.5 \cdot K_{pK}$	∞	0
PI	$0.45 \cdot K_{pK}$	$0.85 \cdot T_K$	0
PID	$0.6 \cdot K_{pK}$	$0.5 \cdot T_K$	$0.12 \cdot T_K$

Table 1: PID Ziegler-Nichols

Den store ulempen med denne metoden er at ikke alle systemer tåler stående svingninger. Noen systemer har også stående svingninger utenfor arbeidsområdet, som betyr at det ikke er fysisk mulig å oppnå stående svingninger.

5.1.3 SIMC

SIMC metode krever at systemet kan utsettes for et sprang uten regulator. Vi ser på sprangen og lager en graf som har tilnærmet lik form som den fysiske prosessen[14]. Dette gjøres ved å sette opp et estimat av systemet $h_u(s) = \frac{K e^{-\tau s}}{(1+T_1 s) \cdot (1+T_2 s)}$, $T_1 > T_2$. Velg en spesifikk tidskonstant ut ifra estimert transferfunksjon h_u ved hjelp av huskereglene i tabell 2. Denne tabellen viser hvordan forholde mellom estimert tidsforsinkelse τ og fasemarginen ψ henger sammen når vi benytter formlene i SIMC metode

$T_{cl} > 2\tau$	Langsom	$\psi > 60^\circ$
$T_{cl} = \tau$	Normal	$\psi \approx 60^\circ$
$T_{cl} = 0.3 \cdot \tau$	Aggressiv	$\psi \approx 45^\circ$

Table 2: Tidskonstant SIMC

Etter valgt tidskonstant setter vi opp formler for å beregne K_p , T_i og T_d slik som i (5.1).

$$K_p = \frac{T_1}{K(\tau + T_{cl})} \quad (5.1a)$$

$$T_i = \min(T_1, 4(\tau + T_{cl})) \quad (5.1b)$$

$$T_d = T_2 \quad (5.1c)$$

Det oppstår ofte situasjoner hvor T_2 er så liten at den blir tilsynelatende lik null $T_2 \ll T_1$. Dette medfører at regulatoren blir en PI-regulator.

Ulempene med SIMC metoden er at det kan være vanskelig å finne et estimat som passer godt nok til systemet. Det er kun mulig å estimere overdempede systemer med tidsforsinkelse. Et eksempel på et overdempet system er temperaturstyring med en liten varmeovn i et større rom. Temperaturen vil øke fort, men økningen avtar til temperaturen blir stabil.

5.1.4 Bode-Nyquist kriteriet

Det er mulig å beskrive systemets lukkede sløyfe gjennom å analysere hvordan systemet i åpen sløyfe oppfører seg ved forskjellige frekvenser[14]. Dette kalles et bode-plot og gir en fullstendig oversikt over regulatorenens hastighet og stabilitet.

For å plotte et bodediagram må vi først dele opp $h(s)$ i faseforskyvningen og amplituden. Dette fører til at vi får to transferfunksjoner en $\angle h(j\omega)$ som beskriver fasen og en $|h(j\omega)|$ som beskriver amplituden.

Vi finner to stabilitetsmarginer ut ifra bodeplottet: ψ og ΔK . Disse forteller om systemet er stabilt. Vi kan også finne hastigheten til regulatoren ved å se på ω_c [14]. ω_c er den største frekvensen (laveste periodontiden) hvor $|h(j\omega)| < 0\text{dB}$. ω_{180° er den største frekvensen hvor $\angle h(j\omega) < -180^\circ$.

$$\psi = \angle h(j\omega_c) - (-180^\circ), \quad \Delta K = \frac{1}{|h(j\omega_{180^\circ})|}$$

Frekvensanalyse Hvis vi kan beskrive systemet vi ønsker å regulere matematisk, kan vi analysere hvordan systemet oppfører seg. Det kan utføres en frekvensanalyse basert på systemet i åpen sløyfe. Ved hjelp av frekvensresponsen til forsterkningen og faseforskyvningen kan vi designe en regulator som opprettholder gode stabilitetsmarginer.

Stabilitetsmarginene er, fasemarginen som alltid må være større en -180° , og forsterkningsmarginen som alltid må være større en 0° . Vi må også sørge for at forsterkningen blir mindre en 0° før faseforskyvningen blir mindre en -90° . Dette betyr også at systemer der fasemarginen aldri blir mindre enn -90° , kan ha uendelig forsterkning slik at kryssfrekvensen (frekvensen hvor forsterkningen blir mindre en 0°) kan bli uendelig stor.

5.1.5 Tidsdiskret

Tidsdiskret system er vanlig i dagens datastyrt verden. Dette fordi en digital datamaskin baserer seg på diskontinuerlige målinger og pådrag[14].

Vi kan oppnå tilnærmet kontinuerlig regulering når samplingstiden går mot null. På samme måte som tilnærming til integraler kan foretas numerisk. Det er ikke alltid ønskelig å måle oftere enn nødvendig, og det er derfor vanlig å beregne det minste antallet målinger som kreves for å regulere.

Det finnes en huskeregel som sier at systemet er tilnærmet kontinuerlig når samplingstiden er mye mindre en tidskonstanten til systemet.

Samplingsteorem For å oppnå korrekt måling av et signal må man måle ofte nok[14]. Hvis signalet er et steg, trenger det ikke å måles oftere enn at kravene for hvor treg målingen maksimalt kan være oppnås. Hvis systemet er i endring må det måles ofte nok til å se endringen før ny endring oppstår. Vi kan enkelt illustrere dette ved å se på en sinusbølge. Hvis det måles et stort antall målinger på en periode av en sinusbølge tegnes et fint bilde av selve signalet. Det kan deretter beregnes tredjegradsfunksjoner mellom tre og tre punkter frem til en tilnærmet perfekt kurve kommer frem. Spørsmålet nå er hvor få målepunkter trengs for å kunne gjenskape sinusbølgjen. Samplingsteoremet beskriver at det trengs mer en to målinger per periode for å oppnå en korekt tilnærming. Måles det sjeldnere enn dette vil estimatet ha en drastisk endret form i forhold til det opprinnelige signalet.

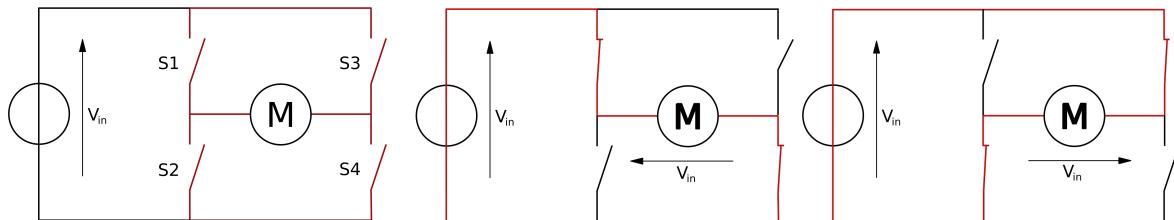
5.2 ROV

Remotely Operated Vehicle er en samlebetegnelse på fartøy som kan kontrolleres fra avstand. Disse må ikke være trådløse, og kan ha kommunikasjonskabler og ekstern energiforsyning. Prinsippet er at personalet som styrer fartøyet må oppholde seg et annet sted enn på fartøyet. Et radiostyrt fartøy følger også disse kravene. Et relevant eksempel på en ROV er en radiostyrt bil, et annet eksempel er en drone.

5.3 H-bru

H-bru er en krets med fire brytere som vist på figur 5.2 [27, 33]. Disse bryterene kan legges inn to og to slik at retningen på strømmen kan endres. Figur 5.2 viser hvordan bryter S1 og

S4 fører strømmen en retning mens bryter S2 og S3 fører strømmen motsatt. Dette er vanlig å oppnå ved hjelp av en transistorkrets. Disse kretsene er designet med tidforsinkelser slik at det ikke er mulig å lage kortslutning. Transistorer har også raskere reaksjonstid og fører til at styresystemet kan styres raskere enn med tradisjonelle kontaktorer eller releer. Det er vanlig å produsere slike halvleder H-bruer med to H-bruer i en pakke slik at det er mulig å styre steppermotorer med en integrert krets. alternativt er det mulig å styre to komponenter med en krets.



Figur 5.2: H-bru [33]

6 Teknologi og utstyr

6.1 Lego Mindstorms EV3

Lego Mindstorms EV3 er en avansert robotsettserie utviklet av Lego Group [6, 17, 23]. EV3 står for "Evolution 3", og det er den tredje generasjonen av Lego Mindstorms-serien. Denne pakken inneholder flere givere og pådragsorganer som kan settes sammen med LEGO til opplæringsformål. Giverene og pådragsorganene kobles opp mot et hovedsentral.

EV3 Intelligent Brick Hovedsentralen i Lego Mindstorms EV3 heter Intelligent Brick, og er avbildet i figur 6.1. Denne sentralen har 8 tilkoblingsporter med en spesialversjon av RJ12, og er preprogrammert til å automatisk gjenkjenne enhetene som kobles til og sette opp I/O liste deretter. Sentralen har bluetooth, usb og internt minne, og kan konfigureres gjennom en applikasjon, styres ved å sende predefinerte kommandoer eller programmeres med en forenklet versjon av Python. Det er mulig å programmere mer generelle koder av en høyere orden enn andre liknende systemer med I/O. Eksempler på andre systemer er PLS-systemer eller mikrokontrollere. Disse systemene krever blant annet at alle tilkoblede enheter settes opp manuelt.



Figur 6.1: Lego Mindstorms EV3 Intelligent Brick

6.2 Raspberry pi

Raspberry pi er en liten ettkortsdatamaskin, originalt designet for å gi utviklere og hobbyister en billig og enkel plattform. Produktet er derfor en meget begrenset datamaskin med I/O utganger. Det er vanlig å kjøre en spesialdesignet versjon av linux som bruker mindre datakraft og gir muligheten til fjernstyring via lokalt nettverk. Denne datamaskinen er mye brukt innen produktutvikling og det finnes derfor mange gratisprogrammer som gir muligheten til å benytte datamaskinen på mange ulike måter. Figur 6.2 viser enheten som ble benyttet. Dette er en Raspberry Pi 4 modell B.



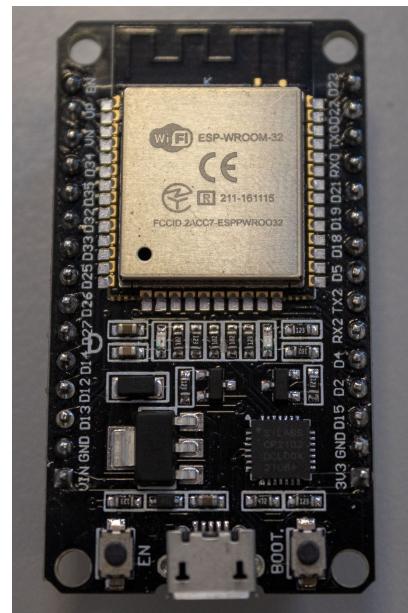
Figur 6.2: Raspberry PI

6.3 ESP32

ESP32 er en mikrokontroller med innebygget wifi-modul [22, 28]. Denne wifi modulen er basert på å benytte ADC og DCA periferiene til å sende og motta wifi-signaler over innebygget antennen som er etset ut på kretskortet kontrolleren kommer på. Vi har benyttet en utviklingsmodul som kommer med lineær spenningsforsyning, knapper til reboot og programmering og dioder som viser CPU status figur 6.3.

6.4 Kretskort

En PCB er en et eller flerlags kobberplate med glassfiberisolasjon mellom lagene. Vi benytter en tolags PCB. Kobberplatene er tynne og tverrsnittet på kobbersporene blir derfor lavt [18, 21]. Produksjonsprosessen består av å etse vekk uønsket kobber før det printes isolerende matereale som hindrer loddetinn fra å feste seg. Til slutt så printes tekst og bilder i hvit farge. Et kretskort er en ryddig og effektiv måte å koble sammen mange små komponenter.



Figur 6.3: ESP32

6.5 Leap Motion Controller

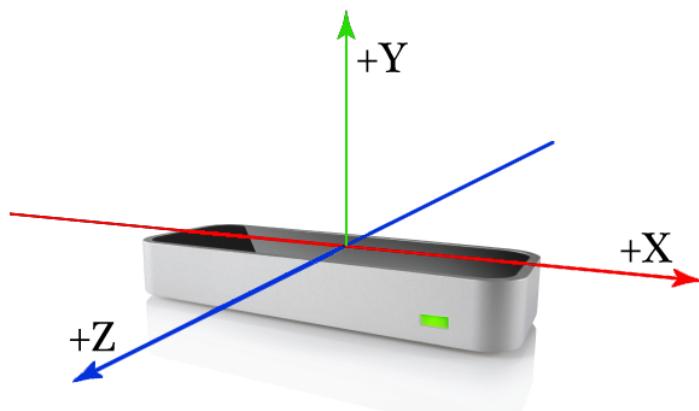
Leap Motion Controller er en enhet for håndbevegelsesdeteksjon, utviklet av Ultraleap [2, 31, 32]. Enheten brukes i mange produkter som ønsker å benytte data om håndens posisjon. Figur 6.4 viser et bilde av enheten som er benyttet i dette prosjektet.



Figur 6.4: Leap Motion Controller

6.5.1 Oppbygning

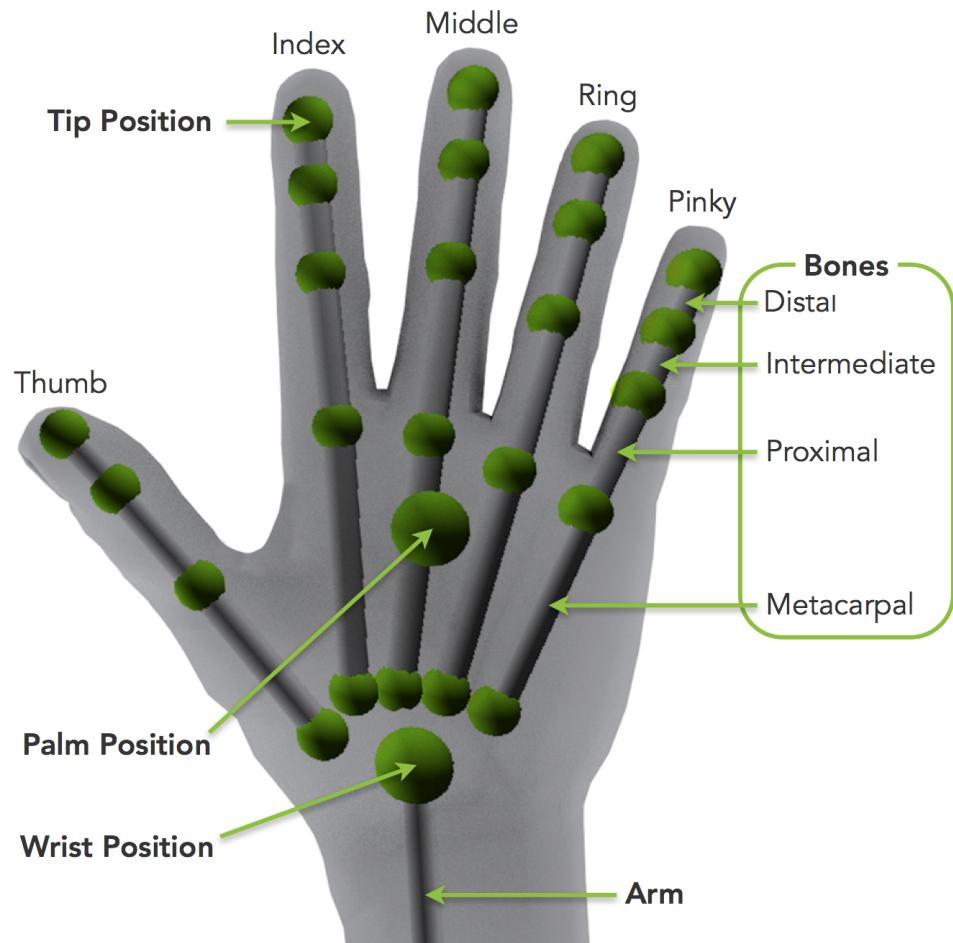
Leap Motion Controller er en enhet som kobles til en PC ved hjelp av USB [31]. Enheten består av en optisk sensor og en nær infrarød LED-lysstripe. Den optiske sensoren fungerer ved å bruke to kameraer til å ta bilder av opp til to hender fra forskjellige vinkler. Deretter brukes en avansert algoritme for å analysere bildene og konstruere en 3D-modell av hendene. Den nær infrarøde LED-lysstripen består av 3 dioder og hjelper til med å belyse hendene for å forbedre nøyaktigheten til sensoren. I utgangspunktet kreves det ingen spesiell programvare eller driverinstallasjon når enheten kobles til en datamaskin, fordi disse ofte implementeres i programvare som benytter enheten. Når enheten er tilkoblet er det ment at den automatisk skal gjenkjenne hendene og begynne å spore bevegelsene deres. Dette gjør at en bruker kan bruke hendene som styreenheter i spill og applikasjoner. I tillegg kan det lastes ned en programvare for ekstra innstillinger og visualisering av hånddeteksjon. Enheten kan ta mellom 20 og 200 bilder i sekundet, avhengig av maskinkraften til den tilkoblede datamaskinen og brukerinnstillingen.



Figur 6.5: Leap Motion Controller koordinatsystem [31]

6.5.2 Håndposisjoner

Hvert bilde blir analysert individuelt og gir nøyaktige 3-dimensjonale posisjonsvektorer for håndens posisjon[2]. Posisjonen er ut ifra origo som vist på figur 6.5 . Posisjonsvektorene blir gitt til hvert ledd i hver finger, i tillegg til håndflaten og håndleddets posisjon. Dette gjør at det til sammen blir 26 forskjellige posisjonsvektorer per hånd fra ett enkelt bilde. Figur 6.6 illustrerer alle leddene som detekteres.

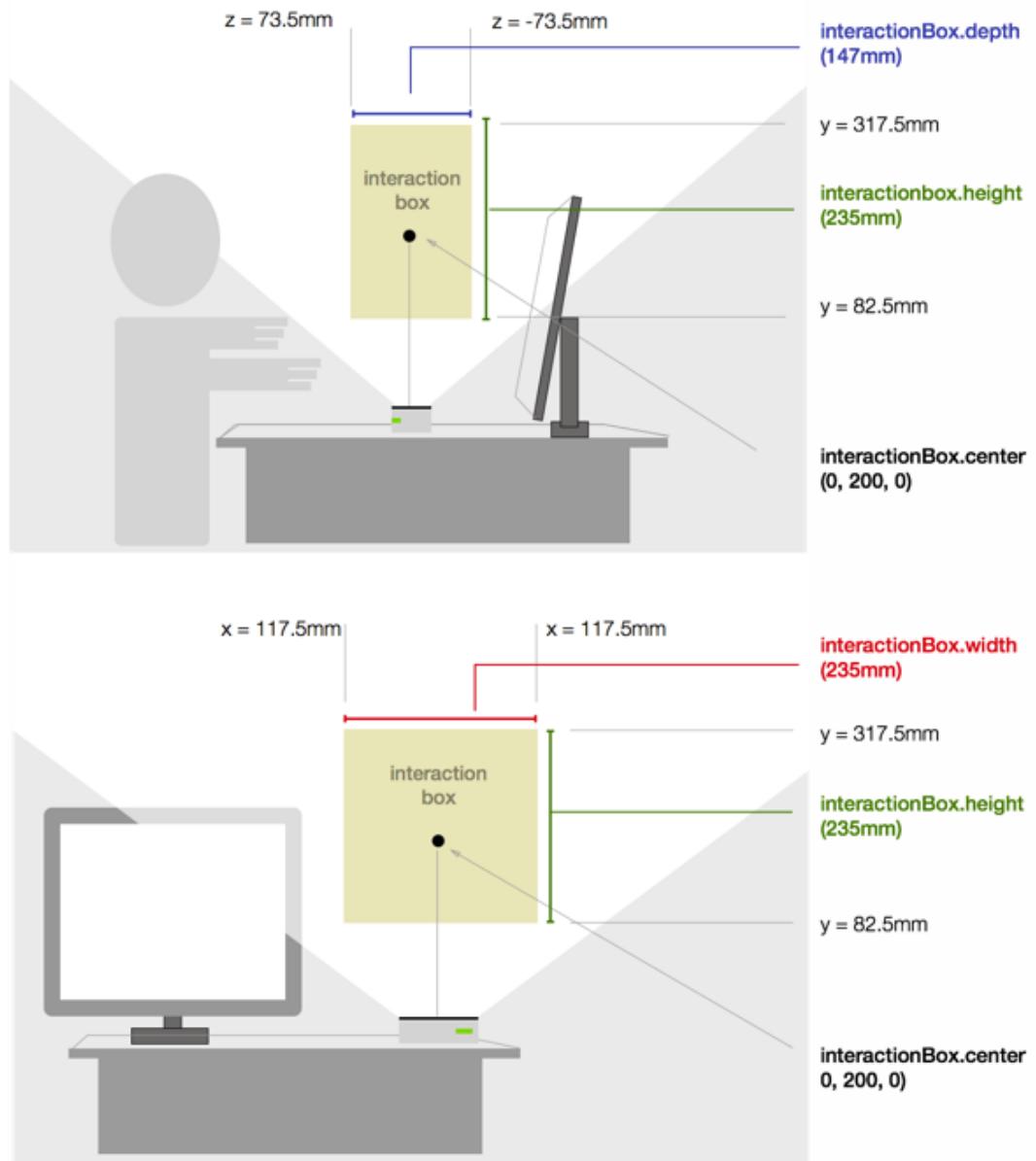


Figur 6.6: Leap Motion Hånd [2]

6.5.3 Interaksjonsboks

Interaksjonsboksen er en usynlig boks plassert over fremfor Leap Motion enheten. Figur 6.7 illustrerer hvor denne kan være plassert i forhold til enheten. Leap Motion enheten kan detektere håndbevegelser i et kjegleformet område opp ifra enheten, men algoritmen for analysering er ikke like nøyaktig ved områdets kanter. Det er derfor avgrenset et prisme, som illustrert i nevnt figur kalt interaksjonsboks. Boksens grenser kommer som standard fra utvikler, men kan ved hjelp at programvareverktøyet settes til å være helt til kantene av deteksjonsområdet. Det er først når det kommer en hånd på innsiden av interaksjonsboksen at Leap Motion enheten vil analysere håndbevegelsene. I dette prosjektet er det benyttet standardgrensene som kommer fra utvikler.

**The Leap Motion interactionBox for a sample frame. Actual numbers will be different from frame to frame.*



Figur 6.7: Interaksjonsboks [31]

6.5.4 UltraLeap Gemini SDK

Programvaren som kan lastes ned for ekstra brukerinnstillinger og visualiseringer heter UltraLeap Gemini SDK [31]. I denne oppgaven benyttes SDK V4. Denne versjonen er utdatert og nyere versjoner finnes på UltraLeap sine nettsider. Den eldre versjonen benyttes fordi denne er kompatibel med Python 3. Det eksisterer en SDK V3 og V2 for eldre enheter som er kompatibel med Python 2.

Gemini 5 driveren og tilhørende SDK V5 inneholder ikke programvare som oversetter windows-objektene, som kommer fra driveren gjennom .dll filen, til Python objekt. Dette fordi driveren er designet til bruk i grafiske programmeringsspråk til design og programmering av VR applikasjoner, eller programmering i C++.

SDK V4 kan kun anskaffes gjennom direkte kontakt med Ultraleap teamet. SDK V2 og V3 kan lastes ned fra den eldre utvikler siden [32] og den nyere SDV V5 kan lastes ned fra den nyere utviklersiden [31].

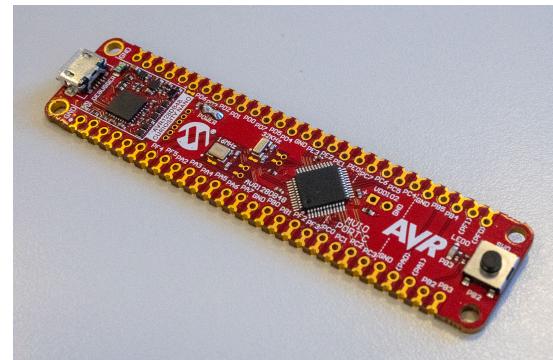
6.6 Datamaskin

Det kreves en datamaskin med windows 7 eller nyere, samt programvaren ULTRALEAP GEMINI V4 [31]. Dette fordi driveren til Leap Motion Controller benytter Windows-struktur og objektekjetet som inneholder informasjonen om håndens bevegelse genereres av driveren ULTRALEAP GEMINI V4, og hentes inn ved hjelp av SDK V4. Hvis datamaskinen ikke er kraftig nok til å oppnå en oppløsning på 30 FPS i driveren, vil dette føre til at programmet henger seg opp og funksjonalitet vil forsvinne.

6.7 AVR128DB48

AVR128DB48 er en mikrokontrollerenhet fra Microchip Technology [10, 11]. Mikrokontrolleren er designet for større varierte prosjekter. Den inneholder periferier spesialdesignet til kommunikasjon og tidsnøying. Kommunikasjonsperiferiene dekker seriell digital kommunikasjon som USART/UART, I2C og SPI. Det er også mulig å kommunisere med analoge signaler som 4-20mA 0-10V eller lyd data inn eller ut. Tidsnøying foregår på flere hastigheter slik at det både kan måles frekvens eller pulsbredd med 16 bit oppløsning og timer eller døgn med høy nøyaktighet. RTC (Real Time Counter) periferiet har også feilkorrigering slik at eksterne klokker kan stille periferiet. Periferiene kan generere events og bruke disse uten å benytte CPU. Dette gir muligheten til å utføre kompliserte jobber uten bruk av CPU, også kalt automatiske bakgrunnsprosesser. Systemklokken drives av ekstern oscillator og kan kjøres opp til 24MHz. De fleste periferiene kan kjøre på separat klokke og det kan implementeres lavfrekvent klokke til strømsparing.

Figur 6.8 viser utviklingskortet som blir benyttet. Dette inneholder debugger, strømforsyning og oscillatorer.



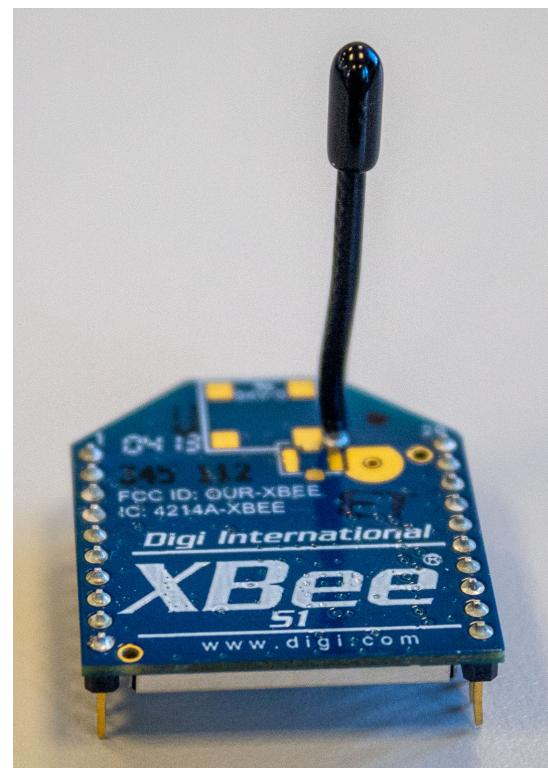
Figur 6.8: AVR128DB48 curiosity nano

6.8 XBee

XBee er en serie av radiomoduler som brukes for trådløs kommunikasjon over korte avstander[7, 8, 9, 12]. Modulene er produsert av Digi international og er basert på Zigbee-protokollen [25]. I denne oppgaven benyttes XBee S1, som er en utgave av XBee serien utgitt i 2007. Modellen har en maksimal overføringshastighet på 250 kbps og opererer i frekvensbåndet rundt 2,4GHz. I et åpent landskap og optimale forhold har Xbee S1 en rekkevidde opp til 100 meter.

6.8.1 Programmering av XBee

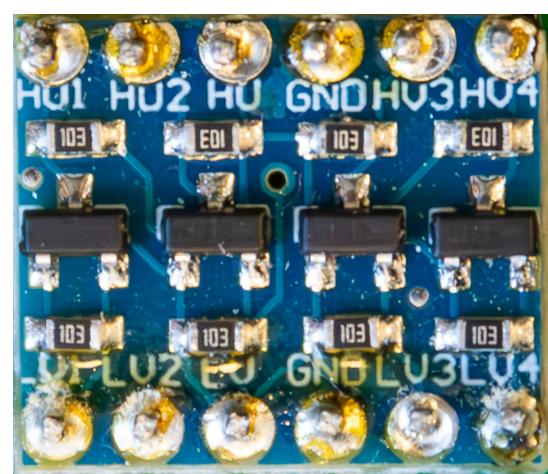
Programmering av XBee-moduler skjer ved hjelp av programvareverktøyet XCTU (XBee Configuration Test Utility). For å programmere XBee-modulene må det benyttes et utviklingskort. Utviklingskortet består av et kretskort med kontaktflate for XBee og en serielltilkobling. Når alt er tilkoblet på denne måten vil det være mulig å programmere inn en rekke diverse innstillinger[8]. Figur 6.9 viser et bilde av XBee-modulen som er benyttet i dette prosjektet.



Figur 6.9: XBee

6.9 Logic Level Converter

LLC (Logic Level Converter) er en transistorkrets som gjør om spenningen på et digitalt signal [3, 4, 5]. I dette prosjektet benyttes denne til å omgjøre 3V3 til 5V. LLC kretskortet har en jordingstilkobling som ikke benyttes av kretsen, en tilkobling til LV (Low Voltage) og en til HV (High Voltage). Det er viktig at den laveste spenningen kobles på LV fordi transistoren benytter spenningsfallet mellom HV og LV. Hvis en av sidene dras lavt med stor nok strøm sendes den andre siden lavt gjennom transistoren. Det motsatte oppstår hvis en av sidene dras høyt med stor nok strøm. HV og LV sidene har $10K\Omega$ pull-up resistans som drar signalene høyt hvis ingen av sidene dras lavt av andre komponenter. Figur 6.10 viser et bilde av den ene av to LLC benyttet i dette prosjektet.

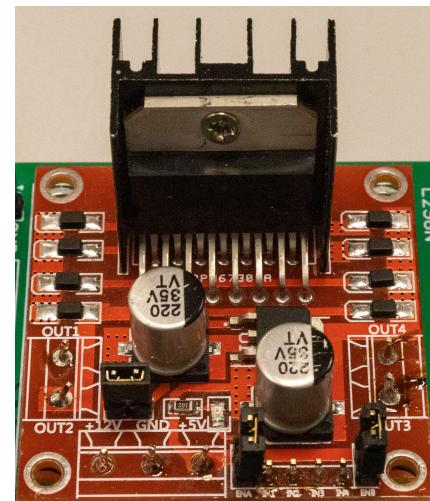


Figur 6.10: Logic Level Converter

6.10 L298N Motordriver

L298N er en ferdig motordriver som benytter L298 H-bru til å styre utgangssignalene [26, 27]. Denne har innebygget lineær transformator fra opp til 12V inn og 5V ut. Spenningsfallet over driveren er nesten 2V som betyr at ved en 11 volts spenninng inn blir det ca 9V ut. Driveren tar inn 4 styresignal. Digitalt 5V spenningsignal inn på disse fører til høyt eller lavt nivå på tilsvarende motorutganger. Dette betyr at det ved å sette begge utgangene til en motor like, enten høye eller lave, vil det ikke gå strøm gjennom motoren. Retningen kan styres med å bytte hvilke utgang som settes høy. L298 har en kapasitet til å kunne drive motorer på opp til 46 volt og dra 2 ampere per motor.

Figur 6.11 viser motordriveren loddet på hoved kretskortet.

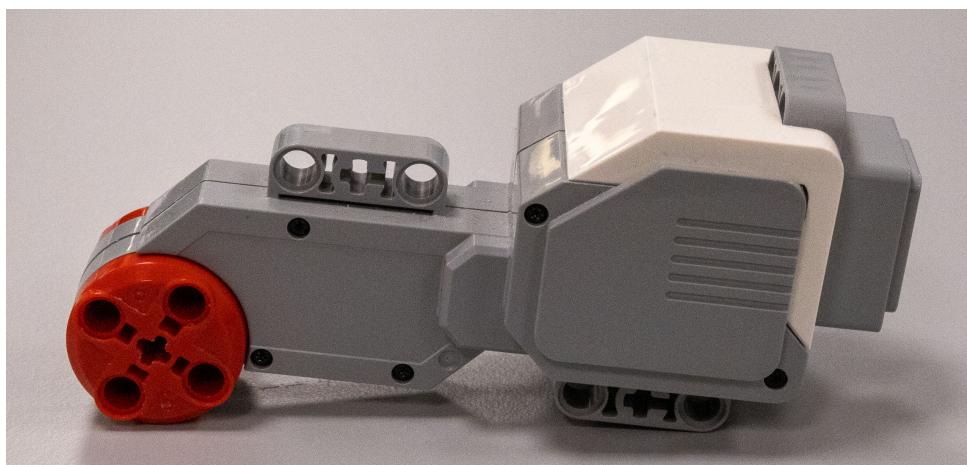


Figur 6.11: L298N motor driver

6.11 EV3 L-Servo Motor

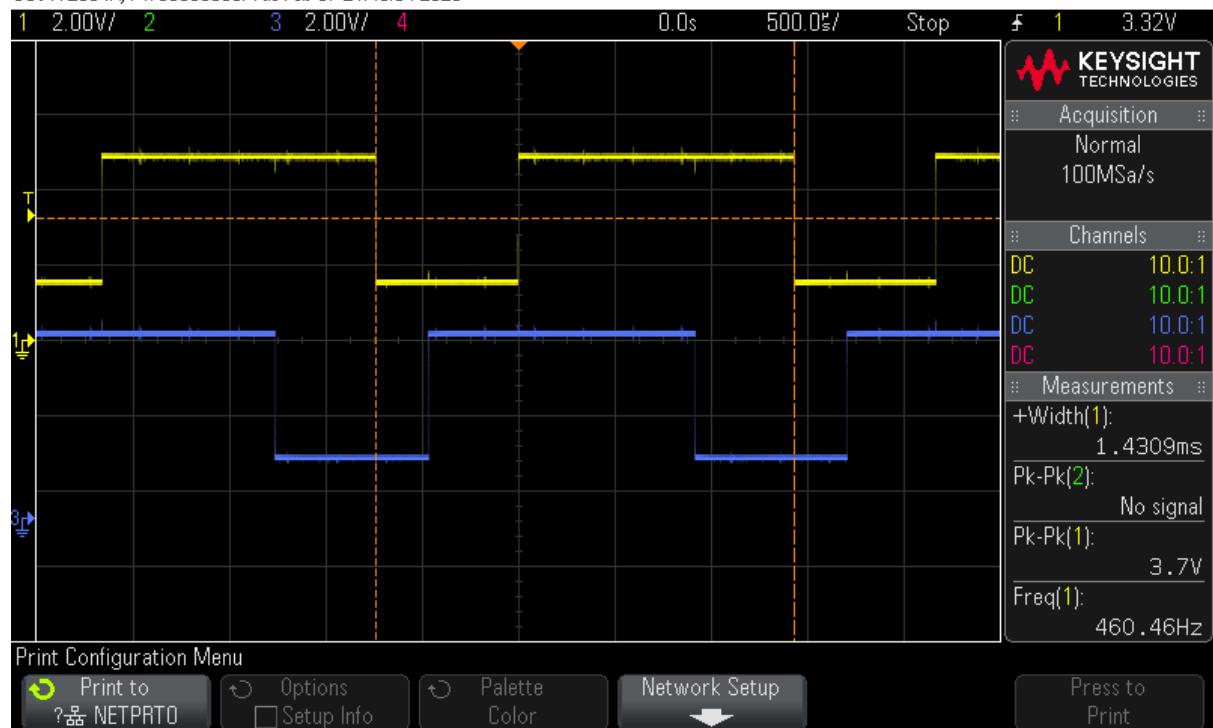
"LEGO 45502 EV3 Large Servo Motor" er en relativt kraftig og robust DC-motor [17]. Det skal bemerkes at navnet er noe misvisende. På grunn av teknologien i LEGO Mindstorm EV3, vil motoren oppføre seg som en servomotor i Mindstorm-prosjekter. Dette er drøftet i delkapittel 6.1. I dette prosjektet brukes motoren uten hovedsentralen, og motoren kan ansees som en DC-motor med innebygde takometer.

Motoren inneholder en kraftig likestrømsmotor parallelkkoblet med en kondensator. Motoren kommer også med to innebygde takometer som individuelt gir 180 pulser per rotasjon. Disse kan benyttes til hastighetsmåling og retningsmåling, og fører til en opplosning på 720 punkter per 360 grader dersom både stigende og synkende flanke måles. Figur 6.13 viser målt takometerpulser for roterende motor i begge retninger. Motoren er spesialdesignet til å bli styrt av hovedsentralen til Lego Mindstorms (NXT). Normalt strømtrekk på motorene er ca. 60mA, men dersom motoren stanses fysisk ved maksimumsspenningen på 9V, trekkes det 1,8A.



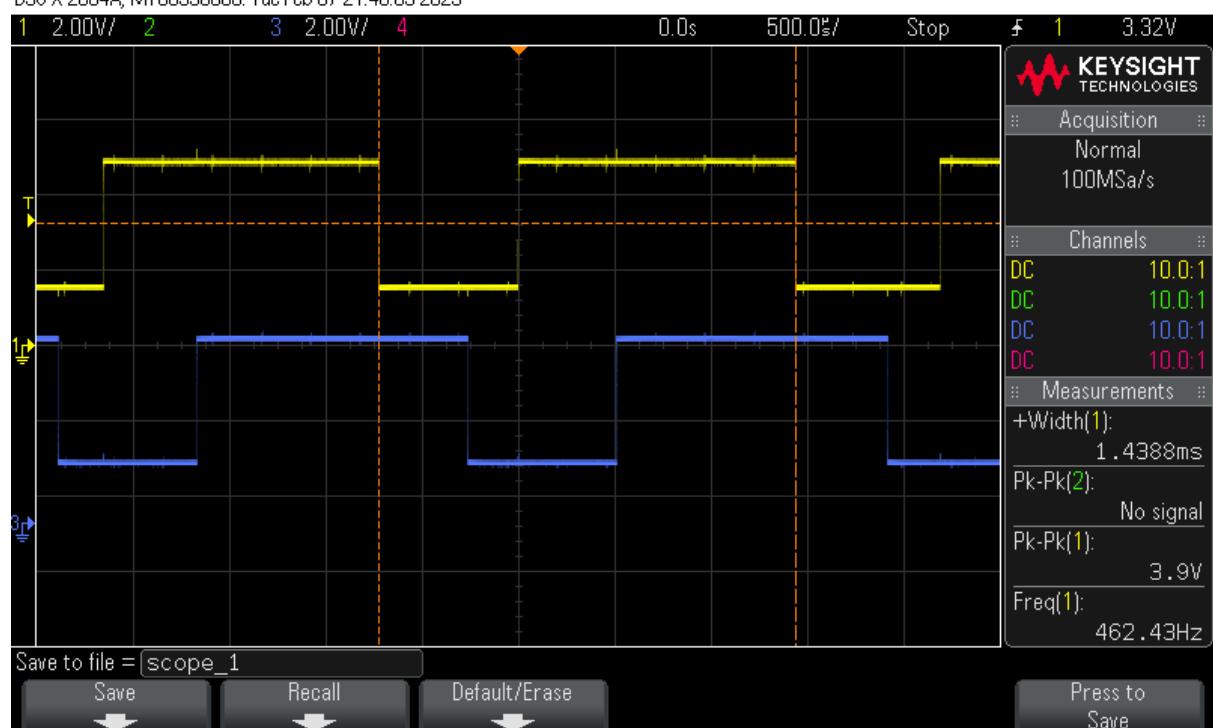
Figur 6.12: LEGO 45502 EV3 Large Servo Motor

DSO-X 2004A, MY55390805: Tue Feb 07 21:46:34 2023



(a) Rotasjon i positiv retning

DSO-X 2004A, MY55390805: Tue Feb 07 21:46:59 2023



(b) Rotasjon i negativ retning

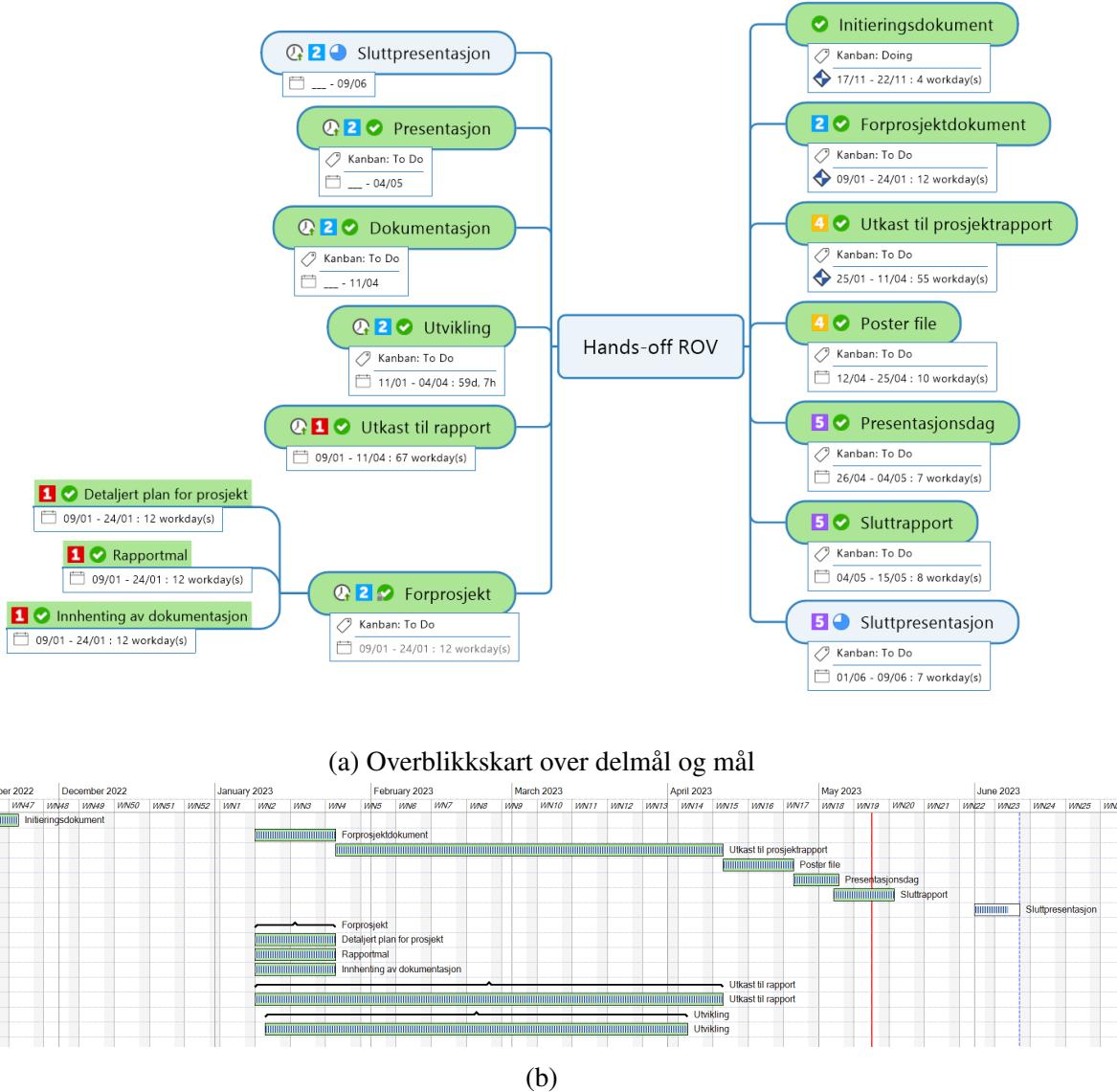
Figur 6.13: Takometerpuls av motor ved rotasjon

7 Gjennomføring

Dette delkapittelet inneholder utviklingsprosessen, valgene gruppen gjorde og hvordan demonstrasjonen ble konstruert. For å få en innblikk i ferdig produkt kan denne videoen benyttes <https://youtu.be/i7mk75wOXS0>, videoen ble produsert parallelt med utviklingen av demonstrasjonen. Den siste versjonen av kodene som er benyttet ligger på Github og oppdateres fortløpende <https://github.com/Oystein-Falkeid/Hands-off-ROV>.

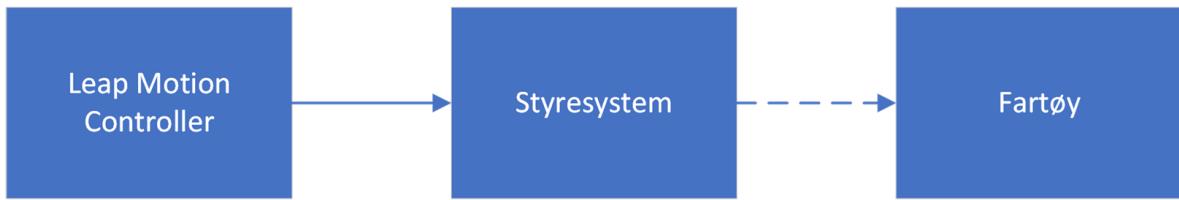
7.1 Fremdriftsplan

I forprosjektet ble det utviklet en fremdriftsplan. Denne planen er visualisert i figur 7.1, og basert på fristene knyttet til bachelorprosjektet.



Figur 7.1: Fremdriftsplan med tidsskjema og frister

7.2 Overblikk av systemet



Figur 7.2: Systemets oppbygning

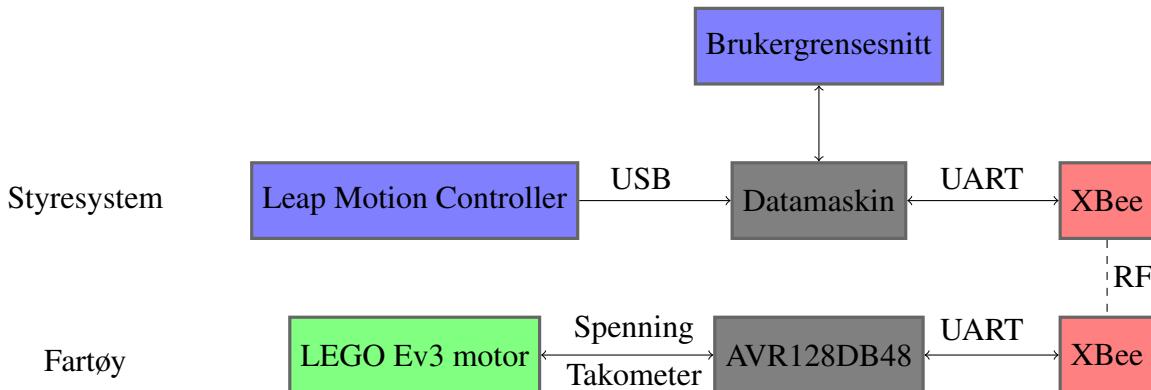
Figur 7.2 viser i korte trekk hvordan styresystemet er bygget opp og hvordan kommunikasjonen fungerer. Heltruket linje er kommunikasjon via kabel og stiplet linje er trådløs kommunikasjon. Denne modellen gir grunnlag for utviklingen av demonstrasjonen. Vi vil senere se en utvidet versjon av av kommunikasjonen i figur 7.3, og en enda mer detaljert for fartøyet i figur 7.4.

7.2.1 Fartøyet

Bakgrunnen for fartøyet som benyttes i dette prosjektet er sammensatt av deler og komponenter som gruppen har vært innom i løpet av studietløpet Automasjon, ingeniør. Fartøyet er inspirert av hvordan andre har taklet utfordringene ved kontaktløs styring [1, 16, 20].

Fartøyets base er EV3 legorobot som de nye automasjonsstudentene hvert år blir introdusert til i faget "TEK-1513, Innføring til ingeniørfag". I faget benyttes en legorobot som skal programmeres til å bli styrt av en PID-regulator, og skal bli kjørt gjennom en bane automatisk. Motorene og basen fra oppgaven er tatt videre til denne oppgaven, men i steden for EV3-hovedsentralen, som var basen i innføringsfaget, benyttes her en AVR128DB48. Mikrokontrolleren er hentet fra faget "AUT-2602, Programmering med Mikrokontroller" og programmeres ved hjelp av programvareverktøyet Microchip Studio (C-programmering).

7.2.2 Systemet i sin helhet

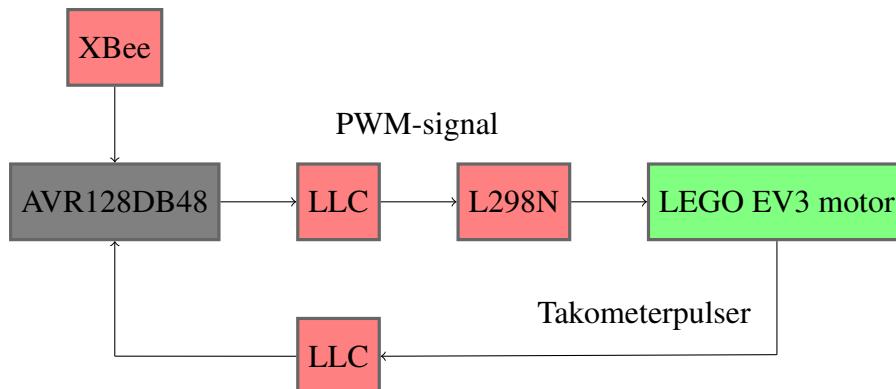


Figur 7.3: Flytskjema for kommunikasjon

- Leap Motion Controller. Tar bilder av hånden mellom 20 og 200 ganger i sekunder og analyserer dette ved hjelp av en innebygget AI (kunstig intelligens). Data basert på analysen av håndens form og posisjon sendes via USB til en datamaskin.

- Datamaskin. Bildene og dataen fra Leap Motion Controller mottas og behandles ved hjelp av en ferdig driver og en SDK (System Development Kit). Denne dataen er basert på håndens fysiske begrensninger og settes sammen til et objekt i C. Objektet blir hentet inn i Python 3.7.3. Objektet gir tilgang til en rekke ferdig prosesserte data som vektorer, normaler og håndtegn.
- Brukergrensesnitter har blitt en stor del av oppgaven og står for all kommunikasjon med brukeren.
- XBee. En simpel metode for trådløs overføring av data. Vi benytter UART 8n1 med baudrate 115200.
- AVR128DB48 er en kraftig mikrokontroller som programmeres i C. Fartskommando kommer inn fra XBee som $\pm 100\%$ på høyre og venstre. Disse brukes til å regulere hastigheten til fartøyet. Avstanden og retningen som kjøres sendes tilbake til datamaskinen via XBee.
- LEGO EV3 motor kjører på 5-9V og kan hastighetsreguleres ved å senke tilført effekt. Motoren har innebygget takometer som brukes til retning- og hastighetsberegnning.

7.2.3 Fjernstyrt fartøy



Figur 7.4: Flytskjema over intern kommunikasjon i ROV

- LLC (Logic Level Converter). Et kretskort som inneholder blant annet en BSS138 transistor. Denne brukes til konvertering av spenningen til det logiske signalet fra 3V3 til 5V og motsatt.
- L298N Motor Driver. Motordriveren består av et kretskort med blant annet intern spenningsomformer. Det må anvendes ca 11V inn for å få 9V ut til motorene. Spenningsomformeren leverer 5V til styring så lenge inn spenningen er mindre en 12V. 5V benyttes til å kjøre takometer og debugger på AVR.
- AVR128DB48 curiosity nano. Kortet har en 5V til 3V3 lineær spenningskonverter som forsyner AVR og XBee. AVR har 4 utganger som kjører motordriveren ved hjelp av PWM (pulsbreddemodulasjon), og 4 innganger som leser pulsene fra takometer.

7.3 Prototype

For å finne løsninger som kan være lengre inn i utviklingsprosessen er det viktig å finne ut hva som ikke fungerer. Dette ble en stor del av oppstartfasen.

Som mange kunstnere sier er det viktig å finne ut alle metodene det er mulig å ødelegge et kunstverk på, slik at det på neste forsøk kan unngås. Det blir derfor i dette delkapittelet presisert hvilke utstyr og metoder som ikke fungerte. Slik at et eventuelt videre arbeid kan foregå uten å gå i de samme fellene som denne gruppen har vært innom.

7.3.1 LEGO Mindstorms

Det er i dette prosjektet benyttet to motorer med navn "EV3 L-Servo Motor" [17]. Disse er hentet fra EV3-pakken og inneholder en kraftig saktegående motor med hastighets- og rotasjonsmåling. Motorene trekker 60mA ved drift uten last, men kan trekke opp til 1.8A ved maksimal motstand. Motorene er robuste og bygget for å tåle hard bruk. De tåler dermed den røffe behandlingen som kommer gjennom prototyping og testing. Dette hindrer utsettelse i fremdriften grunnet ødelagte motorer.

Ved prosjektets start ble det hovedsentralen EV3 Intellegent Brick benyttet som styreenhet for motorene. Det ble konkludert med at denne var for treg og utdatert til å kunne benyttes som en ROV, da responstiden var for stor. Kontrollenheten er designet for å utføre preprogrammerte kommander i steden for å bli benyttet som en slave. Dette betyr at systemet bruker relativt lang tid fra det mottar en kommando til kommandoan utføres. For å oppnå raskere responstid må det benyttes en annen kontrollenhet til å styre motorene.

7.3.2 ESP32

ESP32 er en ypperlig mikrokontroller til testing [28]. Den kommer med et brukervennlig programspråk og mange nettressurser som er verdifulle i oppstartfasen. Det er enkelt og raskt å sette opp programmer som kommuniserer over Wifi eller Bluetooth. Det finnes predefinerte funksjoner som hjelper programmeren med å sette sammen programmer som benytter de periferimodulene på mikrokontrolleren. ESP32 ble derfor benyttet til testing av kommunikasjon med XBee.

Oppgaven kunne også ha blitt utført ved å benytte ESP32 som kontrollenhet på fartøyet. Dette velges imidlertid bort fordi oppdragsgiver har AVR128DB48 tilgjengelig [10]. Denne mikrokontrolleren er ikke nødvendigvis bedre en ESP32 men det er mulig å programmere den spesifikt til våre formål, og kompilatoren gir tilgang til alle de perifere modulene. Disse modulene kan settes opp som beskrevet i et detaljert datablad, og ESP32 blir derfor utkonkurrert av AVR grunnet sin allsidighet og fleksibilitet.

7.3.3 Raspberry PI

I prosjektet har det vært en ide å bruke en Raspberry PI for databehandling og for styring av motorer. Komponentene på Raspberry PI er i lav prisklasse, og hastigheten er også deretter. Planen var å sende håndkoordinatene til Raspberry PI og bruke denne til å behandle dataen, for så å videresende styrekommandoer til motordriveren.

Programmet som inngår i kontrolleren blir først implementert på en datamaskin. Det kommer tydelig frem at Raspberry PI ikke er egnet til å behandle mengden data som kreves for å kjøre driveren til Leap Motion Controlleren. Det er derfor ikke anbefalt å sette opp kontrolleren på små eller eldre datamaskiner.

Raspberry PI ble imidlertid benyttet til test av kommunikasjon med EV3. Dette fordi linux-operativsystemet gir mulighet til å enkelt sette opp UART kommunikasjon på TX og RX pinnene på I/O modulen. Over UART er det mulig å sende json objekter med kommandoer til EV3, men som nevnt tidligere er EV3 for treg til at styringen fungerer som ønsket.

7.4 ROV

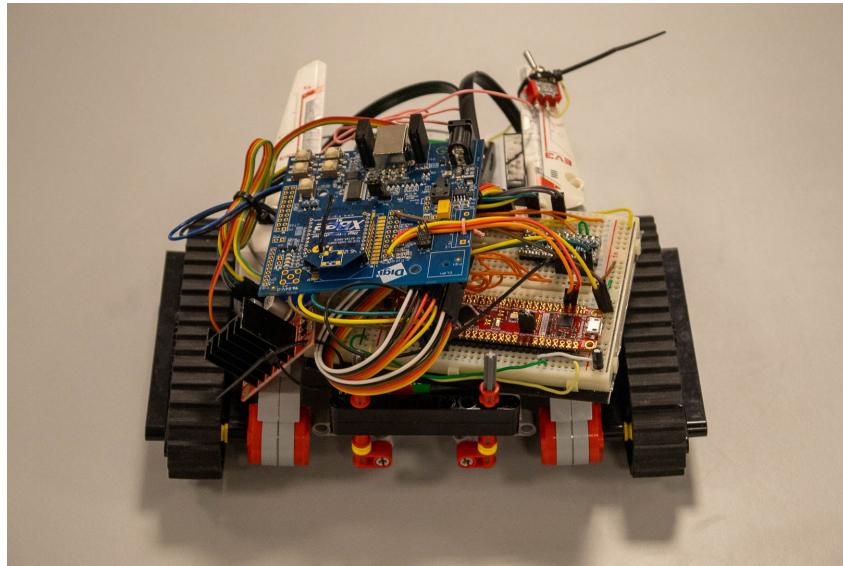
Remotely Operated Vehicle er samlebetegnelsen vi sikter mot. Vi har laget et eksempel på denne typen fartøy for å vise hvordan Leap Motion Controller kan benyttes til interaksjon med fjernstyrte fartøy. Vi har benyttet et et landbasert fartøy med belter heretter kalt fartøyet. Dette begrenser mulighetene for styring, men forenkler styresystemet betraktelig.

Fartøyet består av to motorer med belter som drivverk. Motorene består av en børsteløs likestrømsmotor med to innebygde takometer. Takometrene er laget av en lysgivende diode og en lys, følsom transistor som er adskilt med en sirkuler plate med hull. Platen roterer rundt aksen til motoren og gir ut 180 pulser per rotasjon. Dette gir fartøyet muligheter for hastighetsmåling, posisjonsmåling og retningsmåling.

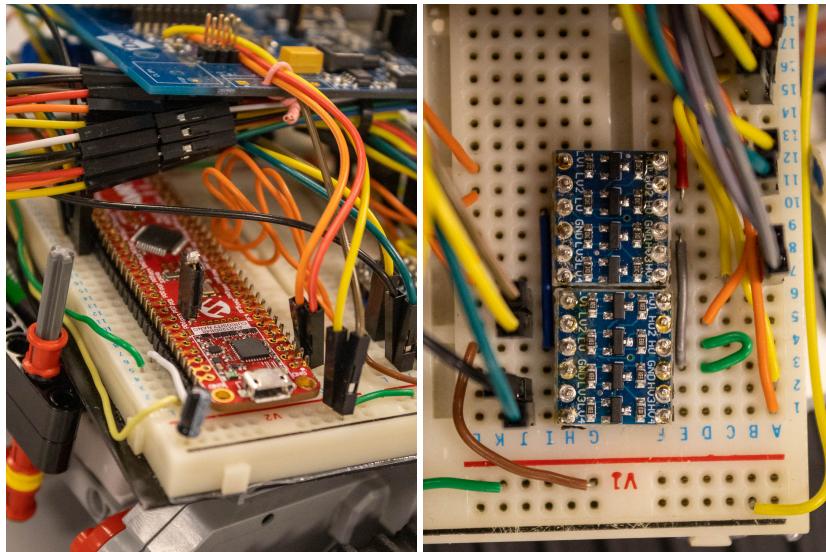
Diameteren på drivverket er ca 37mm, som fører til at fartøyet forflytter seg ca 116mm per rotasjon på motorene. Ut ifra denne avstanden og takometrene skal vi senere beregne hvor langt fartøyet kjører og hvordan fartøyet svinger.

7.4.1 PCB

Det ble raskt åpenbart at fartøyet trengte en bedre måte å koble komponenter sammen på. Det oppsto mange hendelser hvor gruppen trodde koden ikke fungerte, mens det i etterkant viste seg å være løse ledninger eller dårlig kontakt. En ide var da å utvikle et kretskort hvor komponentene kunne loddes på. Figur 7.5 viser hvordan fartøyet så ut tidlig i prosjektet.



(a) Første prototype av fartøyet



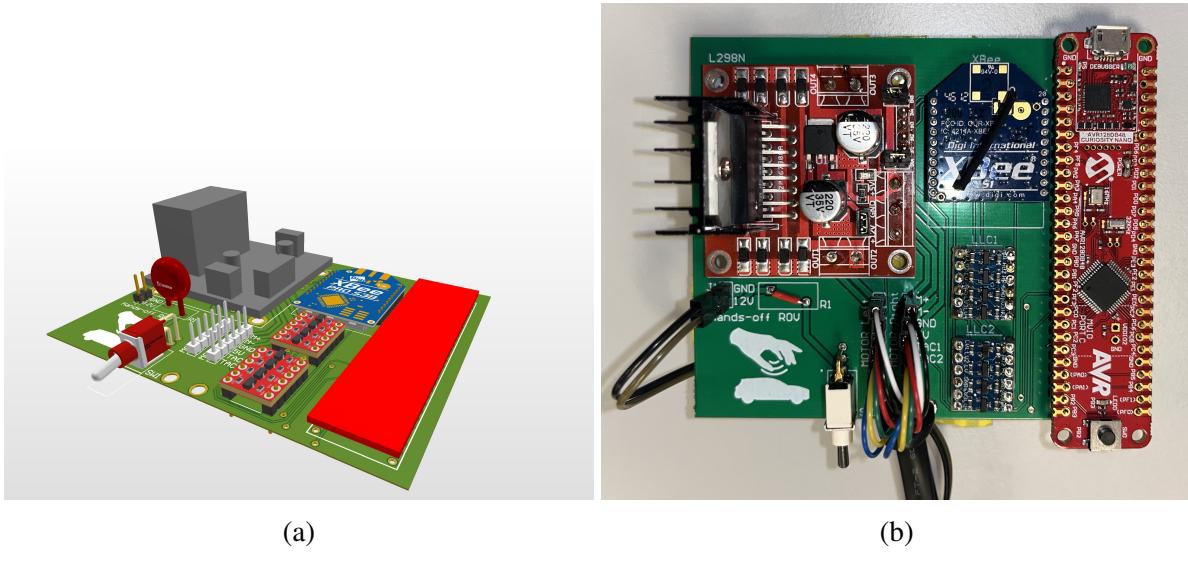
(b) AVR128DB48 curiosity nano

(c) LLC

Figur 7.5: Fartøyet under første funksjonstest

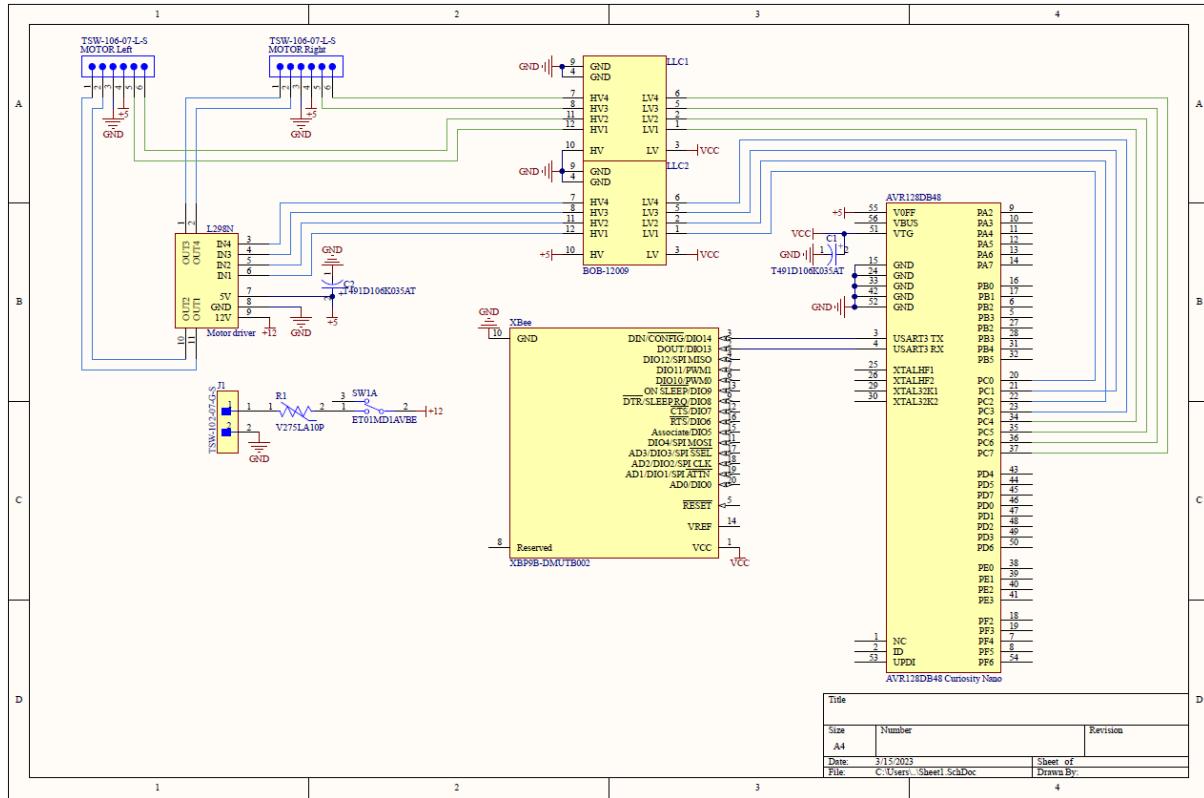
Kretskortet gjør fartøyet mer kompakt. Vi koblet originalt opp hele kretsen på koblingsbrett med lange ledninger, og stripset dette fast til fartøyet som vist i figur 7.21a. Kretskortet består av en tosidig PCB med jordingsplan på begge sider. For å utvikle og designe kretskortet ble programvareverktøyet Altium Designer brukt. Altium har muligheten til å hente detaljer om forskjellige komponenter fra produsenten, som for eksempel fotavtrykk. Fotavtrykkene til XBee S2 Pro (identisk til XBee S1 som blir benyttet) og BOB-12009 (Logic Level Converter) ble hentet ut på denne måten, slik at disse komponentene kunne plasseres rett på kretskortet.

Fotavtrykkene til AVR128DB48 Curiosity Nano og L298N Motor Driver kunne derimot ikke oppdrives, og disse måtte derfor måles og ble designet på egen hånd.



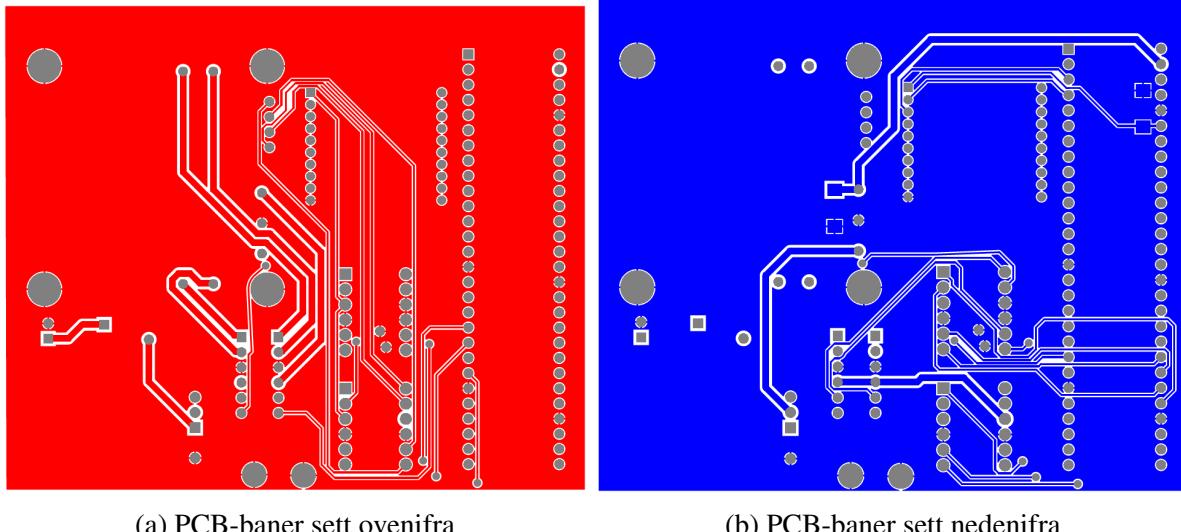
Figur 7.6: PCB 3D model

Motorene kobles til kretskortet ved hjelp av en TSW-106 PCB header og en avklipt RJ12 kabel. Batterispenninng forsynes til L298N via en TSW-102 PCB header. Tilførselen på 11V er designet med ekstra kraftige spor og litt større avstander til andre komponenter, for å unngå varmgang og kortslutning[18, 21]. Figur 7.8 viser kobber banene etset ut på toppen figur 7.8a og på bunn figur 7.8b av kortet, og at 12V og motor tilførselen er større en signal banene.



Figur 7.7: PCB tegning

Fra skjematikken i figur 7.7 kan vi se hvordan komponenten er koblet sammen. Vi har valgt å sette sammen ferdige kretser til en større krets. Dette betyr at kortet benyttes hovedsakelig til overføring av informasjon og effekt. Kortet består av en batteritilkobling som deretter sender 11V spennin gjennom en PTC (Positive Temperature Coefficient). PTC-elementet benyttes som sikring for å hindre varmgan på selve koblingsbrettet, og beskytte batteriene (PTC ble aldri levert fra produsent). Det er også koblet opp en bryter som fungerer som av/på bryter eller resett.



Figur 7.8: PCB-baner

11V blir transformert ned til 5V i en lineær spenningsregulator på L298N og forsyner HV-siden på LLC, samt debugger på AVR128DB48. 5V blir transformert ned til 3.3V i en lineær spenningsregulator i debuggeren og sendes til AVR128DB48, XBee og LV på LLC.

PWM-signal til motorene sendes fra AVR128DB48 PORTC PIN 0 og 1. Signalet forsterkes til 5V gjennom LLC og 9V gjennom L298N. Signalene som styrer retning sendes fra AVR128DB48 PORTC PIN 3 og 4. Takometerpulsene fra motorene mottas på AVR128DB48 PORTC PIN 5, 6, 7 og 8. Dette signalet blir transformert fra 5V til 3V3 i LLC1.

Merk at HV-siden på LLC1 er koblet til GND. Dette fordi forsterkeren i takometerkretsen ikke trekker nok strøm til å trekke spenningen lav hvis HV settes til 5V. LLC inneholder en 10Kohm pull-up motstand til HV og denne fungerer nå som en pull-down. HV kan settes flytende, men da vil støy fra takometrene påvirke hverandre gjennom pull-up resistansene.

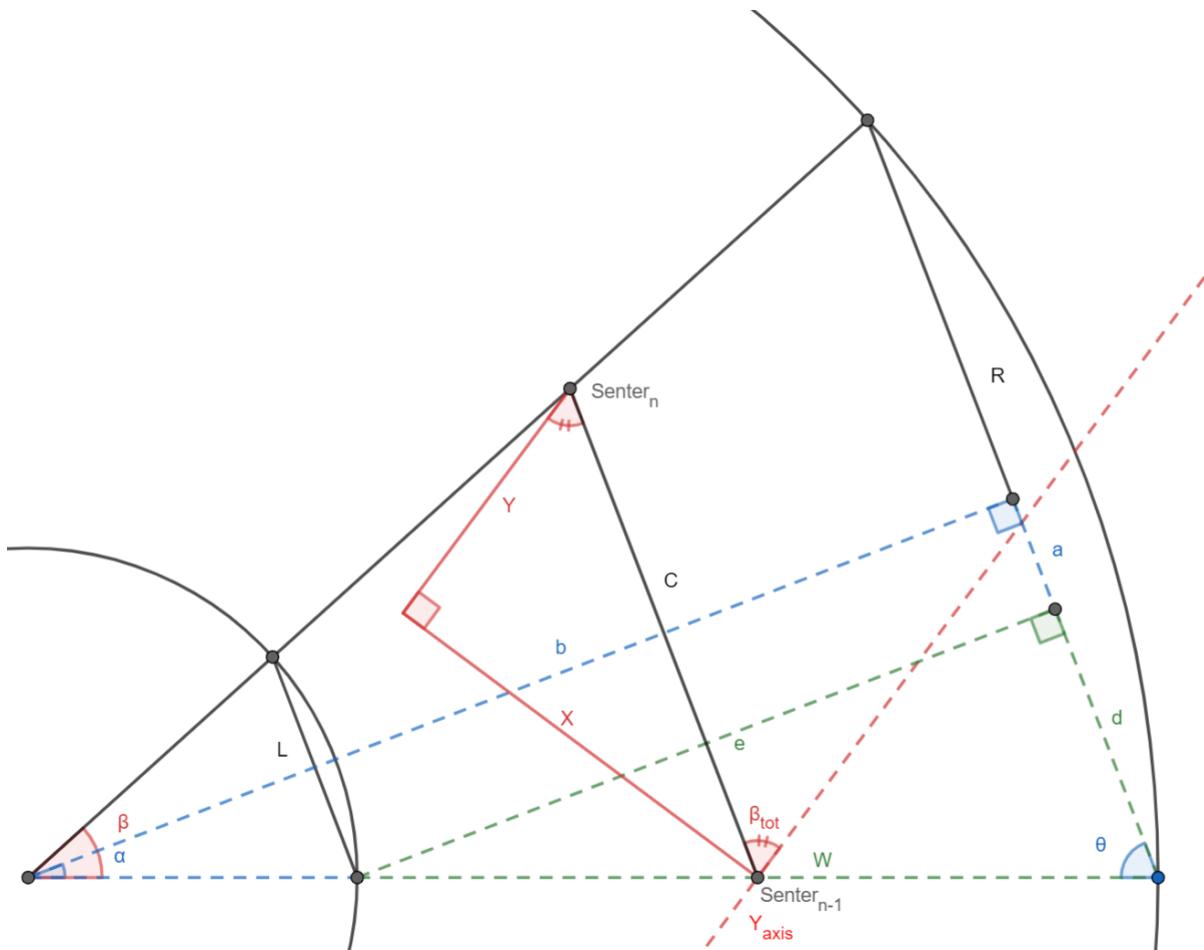
7.4.2 Retningsberegning

Vi kan se fra figur 7.9 at det er mulig å beregne rotasjonen til bilen fartøyet et punkt ved hjelp av trigonometriske funksjoner. Disse er listet opp i beregning (7.1) og settes sammen til et forkortet uttrykk for vinkelen β .

$$\beta = 2 \arctan \left(\frac{R - L}{\sqrt{4W^2 - (R - L)^2}} \right)$$

Vi ser tydelig fra figur 7.9 at avstanden L og R ikke tilsvarer den faktiske distansen fartøyet kjører. Ved å dele opp målingene i mindre biter, kan vi komme nærmere den faktiske distansen som fartøyet kjører. Vi har 180 målinger per rotasjon og kan dermed beregne den minste rotasjonen som vist i beregning (7.2).

Vi tar utgangspunkt i at beltene svinger rundt midtpunktet uten glidning. Dette er antatt fra observasjoner, og vi ser at den økte diametern på støttehjulet i midten skaper et vippepunkt som bilen vrirs rundt.



Figur 7.9: Rotasjon av fartøy

$$\begin{aligned} d &= \frac{R-L}{2} & e &= \sqrt{W^2 - d^2} & \tan(\theta) &= \frac{e}{d} \\ a &= \frac{R}{2} & b &= \tan(\theta) \cdot a & \tan(\alpha) &= \frac{a}{b} \end{aligned} \quad (7.1)$$

$$L = 0 \quad R = 0.000563 \quad W = 0.175$$

$$\beta = \arctan \left(\frac{0.000563}{\sqrt{0.175^2 - \left(\frac{0.000563}{2} \right)^2} \cdot 2} \right) \cdot 2 = 0.00322 \text{ rad} \quad (7.2)$$

7.4.3 Lengdeberegning

Avstanden som hvert hjul forflyttes kan enkelt beregnes med diameteren på hjulet og antall omdreninger. Et problem med denne forenklingen er friksjon. Dersom beltene forflytter seg med ulik hastighet vil fartøyet svinge teoretisk perfekt, eller sklir begge beltene på underlaget? Dette skal vi se mer på i resultatene.

For å finne et korekt mål på lengden må vi ta høyde for rotasjon. Dette fører til at lengden fartøyet har forflyttet seg blir lik lengden på grafen fartøysts posisjon tegner under kjøring. Dette kalles buelengde og beregnes med integrasjon. Fartøyet måler avstand numerisk og dermed vil en numerisk forenkling av buelengden gi et godt estimat på avstanden. Avstanden C kan beregnes med trigonometri som vist i figur 7.9 og forenkles ned til en enklere formel.

$$C = \frac{R+L}{2}$$

Forenklingen ligger i vedlegg B.

$$\text{Diameter} = 0.037m \quad (7.3a)$$

$$\text{Omkrets} = 2 \cdot \pi \cdot \frac{0.037m}{2} = 0.116m \quad (7.3b)$$

$$\text{Strekning per puls} = \frac{\text{omkrets}}{180} = 0.000646 \frac{m}{puls} \quad (7.3c)$$

7.4.4 AVR128DB48

AVR128DB48 programmeres i C, som er et programmeringsspråk basert på det engelske språket [10, 11]. Kode og kodestruktur beskrives derfor på engelsk for å oppnå en mer sømløs og lettleselig kode.

For å øke kapasiteten til mikrochipens CPU anvendes noen av de perifere modulene. Disse benyttes til å generere nøyaktige PWM-signal til motorene, til å lese motorhastigheten med høy nøyaktighet og til kommunikasjon over UART med XBee. Ved å gjøre dette i periferiene frigjøres plass på CPU til PID-regulering og behandling av inn og ut data over UART til kontrolleren.

Programmet følger en enkel struktur og kjører for det meste i en while-loop i main som vi har kalt "Idle" i figur 7.13. Ved oppstart kjøres en rekke initialiseringfunksjoner som definerer hvilke periferier som skal aktiveres og hvilke funksjon disse skal ha, dette vises i boot i figur 7.13). Etter Boot er fullført, aktiveres interrupts og main programmet går i idle.

Det er utviklet et flytskjema som beskriver funksjonaliteten til AVR programmet figur 7.13. Fra figuren kan vi se at programmet hovedsakelig står i Idle og venter på kommandoer. Kommandoene kommer inn i form av interrupts. Interrupts kommer fra periferiene og behandles av CPU.

En interrupt er et hopp i adressefeltet som koden leses fra. Dette hoppet oppstår når en periferimodul sender signal til Event-handeler. For å visualisere dette på en mer menneskelig måte må vi se på maskinkoden. Main-koden er en løkke som kjører gjennom kodelinjene til den når et hopp tilbake til toppen før den kjører på ny. Her står CPU-en og kjører til et interrupt inntreffer. Når dette oppstår hopper koden til adressen med den første kodelinen definert i SEI funksjonen (koden tilhørende det spesifikke interruptet) og CPU-en leser inkrementelt gjennom kodenelinjene i SEI funksjonen. Når SEI funksjonen er ferdig sender event-handeler CPU-en tilbake til den neste kodelinen i main koden hvor CPU-en igjen kjører kodelinjer til nytt interrupt. På denne måten kan det settes opp store koder og programmer som kun kjøres ved spesielle anledninger eller på spesifikke tidspunkt uten å bruke CPU til å påse om disse skal kjøre eller ikke.

USART Det er tre dedikerte periferimoduler til USART kommunikasjon. Det benyttes USART3 som kommunikasjonsmodul fordi denne er koblet opp mot debuggermodulen, slik at vi kan motta signalene over USB til en innebygget Data Visualiser i Microchip Studio. Denne periferi modulen har egen interrupt slik at forskjellige USART3 funksjoner kan aktivere en koderekke i CPU-en. Det settes opp en "message complete interrupt" som kjører kode hver gang RX bufferen på preiferiet har mottatt en datapakke på 8 bit. Denne datapakken lagres i et array slik at verdien kan benyttes andre plasser i programmet og slik at det er plass til neste datapakke i bufferen.

UART init funksjonen i Figure 7.10 starter periferi USART3 for kommunikasjon til debugging gjennom serieladapteren i debuggeren og til kommunikasjon med XBee.

Vi har valgt baudrate på 115200bps for å redusere tiden det tar å sende meldinger. Baudraten begrenser hvor ofte vi kan sende og motta datapakker og hvor lange disse kan være. For å gi muligheten til senere utvikling har vi valgt en stor baudrate, uten å gå ut over RS232 standarden. Baudraten på AVR128DB48 styres av et register som teller klokkepulser. Baudraten er derfor begrenset av CPU klokvens hastighet. Databladet [10] beskriver hvordan baudraten kan

```

1 //init to use PortB0 and PortB1 for USART
2 void USART3_init ()
3 {
4     USART3.BAUD = 137;
5     PORTB.DIR |= USART_TX_PIN_bm;
6     USART3.CTRLA |= USART_RXCIE_bm;
7     USART3.CTRLB |= USART_TXEN_bm | USART_RXEN_bm;
8 }
```

Figur 7.10: USART3_init

beregnes i forhold til periferiklokken. Det er et eget register som fungerer som en "prescaler" for å oppnå korrekt baudrate. Formelen for baudrate tar hensyn til kommunikasjonsmetode og andre innstillingar i periferiet. Det benyttes dermed $BAUD = \frac{64 \cdot f_{CLK_PER}}{S \cdot f_{BAUD}}$ hvor $S = 16$ for asynkron normal mode .

$$BAUD = \frac{64 \cdot 4000000}{16 * 115200} = 136.71875 \approx 137$$

Formelen gir en BAUD verdi på 137 som betyr at vi har et avvik på $\frac{0.71875}{136.71875} \approx 0.0052 = 0.52\%$. Avviket kan ha konsekvenser for kommunikasjonen hvis det blir stort nok.

Periferiet og valgte pinner må konfigureres slik at det er mulig å sende data og å lese mottatt data. For å kunne sende data må TX settes PortB pin1 som utgang. Dette fører til at periferier kan sette utgangen høy eller lav gjennom output modulen på pinnen, se linje 5 i figur 7.10. Vi ønsker også å kjøre en egen separat kode når data mottas. Vi kan oppnå dette med å skru på interrupt hver gang en datapakke er lest inn på RX, se linje 6 i figur 7.10. Til slutt skrur vi på USART RX- og TX-driveren i linje 7 i figur 7.10. Vi kan nå lagre en "uint8_t" eller en "char" hver gang interrupt fra USART3 kjører. Vi kan også sende data med å skrive til USART3.TXDATA.

TCA For å oppnå mer nøyaktig PWM-signal benyttes TCA (Timer counter A). Dette periferiet har muligheten til å generere signal på pinne 0, 1 og 2 på valgt port, i single mode. Det settes opp en dedikert funksjon vist i figur 7.11 som setter opp TCA0 til å generere PWM signal på pinne 0 og 1 på port C. Dette periferiet kjører helt uavhengig av CPU, men benytter samme klokkepuls som CPU. Dette fører til at den eneste kommandoen som kreves for å endre hastigheten til motorene er å lagre en 16 bits verdi til CMP0 og CMP1, noe som kun krever en linje med kode. I realiteten kompileres dette til et par linjer maskinkode men dette er en ubetydelig mengde prosessering sammenlignet med alternativene til PWM generering, som bitbang PWM.

Dette gir muligheten til å styre motorene hastighet med PWM signal på to av motordrivernes pinner og logisk høyt eller lavt på to av motordrivernes pinner. Hvordan dette kobles vises og forklares i kapittel 7.4.7. I grove trekk fører dette til at vi kan styre retning med å sette en pinne høy eller lav, og styre hastigheten med PWM.

TCB Det trengs en måte å måle hastigheten på bilen. I motoren er det innebygde takometer som sender pulser 180 ganger per rotasjon. Hastighet kan måles i RPM multiplisert med omkretsen. Vi kan derfor konkludere med at frekvensen til takometerpulsene kan omgjøres til hastigheten:

$$Hastighet[m/s] = \frac{f_{takometer} \cdot \text{omkrets}}{180} = \frac{\text{omkrets}}{T_{takometer} \cdot 180}$$

```

1 void TCA_Motor_Driver_init()
2 {
3     TCA0.SINGLE.PER = TCA0_CMP_MAX;
4     TCA0.SINGLE.CMP0 = TCA0_CMP_MIN;
5     RightMotor.OUTCLR = LeftMotorReverse_bm;
6     TCA0.SINGLE.CMP1 = TCA0_CMP_MIN;
7     LeftMotor.OUTCLR = RightMotorReverse_bm;
8
9     TCA0.SINGLE.CTRLA = TCA_SINGLE_ENABLE_bm | TCA_SINGLE_CLKSEL_DIV1_gc;
10    TCA0.SINGLE.CTRLB = TCA_SINGLE_CMP0EN_bm | TCA_SINGLE_CMP1EN_bm |
11        ↪ TCA_SINGLE_WGMODE_SINGLESLOPE_gc;
12
13    PORTMUX.TCAROUTEA |= PORTMUX_TCA0_PORTC_gc;
}

```

Figur 7.11: TCA Motor driver

Fra databladet [10] kommer det frem at det finnes 3 relevante periferier spesiallaget til frekvensbehandling. TCA kan generere frekvenssignaler, men ikke måle og TCD er designet for kompleks frekvensgenerering. TCB kan derimot telle tid mellom inngangssignaler.

TCB periferiet har mange funksjoner og kan settes opp på vidt forskjellige måter. Det er derfor en mer kompleks prosess å sette opp TCB til å utføre den oppgaven som ønskes. Fra databladet [10] er det definert en tidsnålingsmetode som heter "Input Capture Frequency Measurement" med "Input Capture on Event mode". Disse innstillingene fører til at tiden mellom hvert event lagres som en 16 bits verdi i CCMP registeret.

Et event kan vere forskjellige endringer som oppstår på mikrokontrolleren. Her benyttes endringen på en I/O pinne som Event-generator slik at TCB kan abonnere på dette eventet og lagre tiden mellom hver gang inngangen har stigende flanke.

En utfordring med TCB periferiet er at den mangler "prescaler" slik som det blir benyttet i TCA for å redusere klokkepulsen til ønsket verdi. Det må derfor settes opp at TCB benytter samme klokkepuls som TCA slik at "prescaler" på TCA kan styre frekvensen på klokkepulsen til TCB

TCB benyttes til tidsnåling mellom takometerpulsene. Tiden telles og lagres i TCB.CCMP hver gang en interrupt mottas fra stigende flanke på takometeret. Compare interrupt kjøres og data leses av fra TCB.CCMP registeret slik at registeret tømmes og interrupt-flagget settes lavt. Tiden omgjøres til meter per sekund ved å dele lengden bilen kjører ved 2 grader rotasjon på tiden den bruker. For å oppnå god oppløsning på både lave og høye frekvenser benytter vi prescaler fra TCA1 til å definere klokkefrekvensen inn på TCB.

Ved TCB interrupt testes hvilke type interrupt som er aktivert. Dette kan vere enten "CAPT interrupt" som oppstår når det er stigende flanke på takometeret eller "OVF interrupt" som oppstår når telleren TCB teller til mer en 65535 og begynner på ny. Ved overflow kan vi anta at bilen står stille og vi må sette interrupt-flagget lavt før programmet går tilbake til main. Ved CCMP interrupt må verdien i CCMP leses av før programmet går tilbake til main.

```

1 void TCA1_init_TCB_CLOC()
2 {
3     TCA1.SINGLE.CTRLA = TCA_SINGLE_ENABLE_bm | TCA_SINGLE_CLKSEL_DIV4_gc;
4 }
5 void TCB2_init()//right motor
6 {
7     TCB2.CTRLA = TCB_CLKSEL_TCA1_gc | TCB_ENABLE_bm;
8     TCB2.CTRLB = TCB_CNTMODE_FRQ_gc;
9     TCB2.EVCTRL = TCB_EDGE_bm | TCB_CAPTEI_bm;
10    TCB2.INTCTRL = TCB_CAPT_bm | TCB_OVF_bm;
11
12    PORTC.PIN4CTRL = PORT_ISC_RISING_gc;
13    EVSYS.CHANNEL2 = EVSYS_CHANNEL2_PORTC_PIN4_gc;
14    EVSYS.USERTCB2CAPT = EVSYS_USER_CHANNEL2_gc;
15 }
```

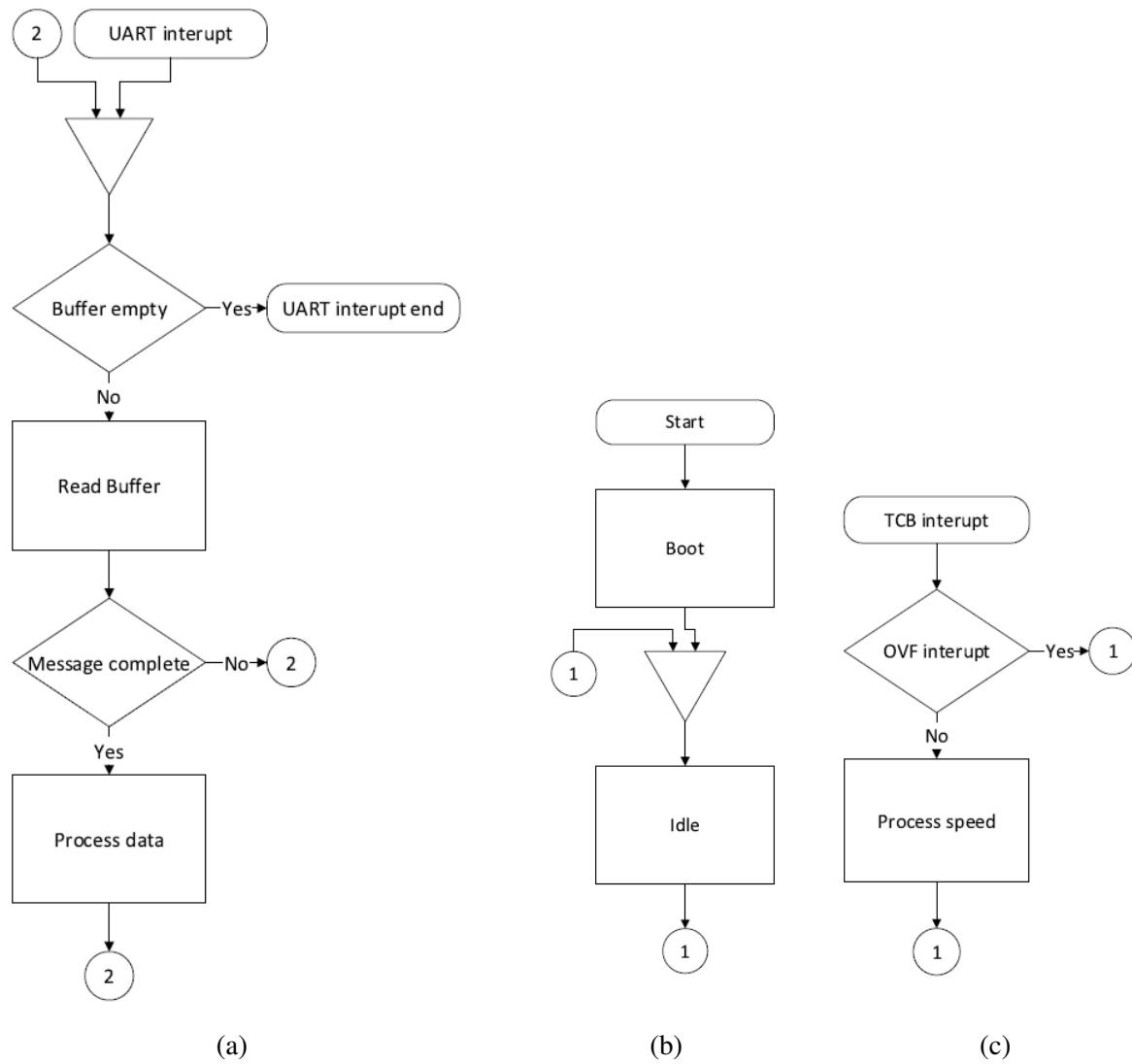
Figur 7.12: TCB takometermåling

Figur 7.12 viser koden som setter opp frekvensmåling. I denne koden settes følgende funksjonalliteter:

- Linje 3 aktiverer TCA1 og setter divisjonsfaktoren på preskaleren til 4. Denne timeren har ingen andre funksjoner en å telle hver fjerde klokkepuls.
- I Linje 7 er det ingen preskaler i TCB med divisjonsfaktor større enn 2, og det benyttes derfor preskaleren fra TCA1 som klokkepuls. TCB2 aktiveres også ved å sette enable registeret høyt.
- Linje 8 definerer grunnoperasjonen til periferiet til frekvensmåling.
- Linje 9 definerer at frekvensen skal måles på flankene og ser ut ifra databladet at dette kun gjelder stigende flanke i "Input Capture Frequency Measurement mode". Det defineres også at tiden skal lagres på "Input Capture".
- I Linje 10 settes det opp interrupts på både "capture" og "overflow" slik at det er mulig å lese ut hastigheten under kjøring eller sette hastigheten til null ved stans.

Merk at linje 12-14 er kun for bruk av PORTC.PIN4 som inngang. Vi ser også at en positiv flanke på PORTC.PIN4 gir positiv flanke på event channel 2, som gir positiv flanke på TCB2 capture. Ligning 7.4 viser utregningen for hastighet.

$$\begin{aligned}
 \text{Hastighet}[m/s] &= \frac{2^\circ \cdot (m/1^\circ)}{\text{TCB.CCMP} \cdot \frac{1/4\text{MHz}}{\text{CLKSEL}}} \\
 &= \frac{2 \cdot 0.0562m}{\text{TCB.CCMP} \cdot \frac{0.25 \cdot 10^{-6}s}{2}}
 \end{aligned} \tag{7.4}$$



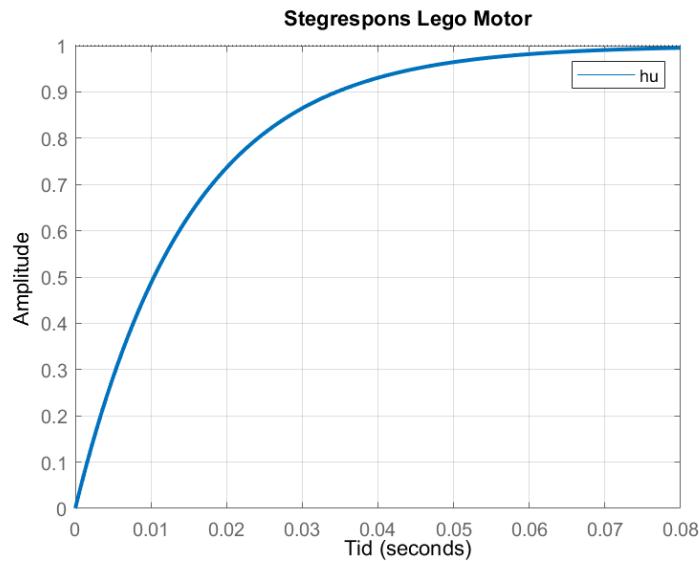
Figur 7.13: Flytskjema over funksjonalitet av AVR128DB48

7.4.5 Regulator

For å designe en regulator trenger vi en metode for å analysere systemet [14]. Vi har to veldig kjente metoder: Ziegler-Nichols og SIMC metode. Det er ikke implementert en metode for grafisk fremstilling av hastigheten som gjør Ziegler-Nichols metode vanskelig å benytte. Hastigheten kan loggføres ved hjelp av datavisualiseringsverktøyet i Microchip Studio, og stegresponsen tegnes manuelt. Stegresponsen i figur 7.14 kan brukes til å finne en tilnærming av $h_u(s)$ ved hjelp av SIMC metode (7.6).

$$\begin{aligned}
 h_u &= \text{Systemets transferfunksjon} & K &= \text{Systemets forsterkning} \\
 \tau &= \text{Tidsforsinkelse} & T_1 &= \text{Førsteordens tidskonstant} \\
 T_2 &= \text{Andreordens tidskonstant} & h_r &= \text{Regulator} \\
 K_p &= \text{Forsterkning} & T_i &= \text{Integrator tidskonstant} \\
 T &= \text{Samplingstid} & u &= \text{Pådrag} \\
 y &= \text{Målt verdi} & e &= \text{Avvik} \\
 s &= \text{Kontinuerlig variabel} & z &= \text{Diskontinuerlig variabel} \\
 k &= \text{Indeks for tidsdiskret system}
 \end{aligned} \tag{7.5}$$

$$h_u(s) = \frac{K}{(1 + T_1 s)} \tag{7.6}$$



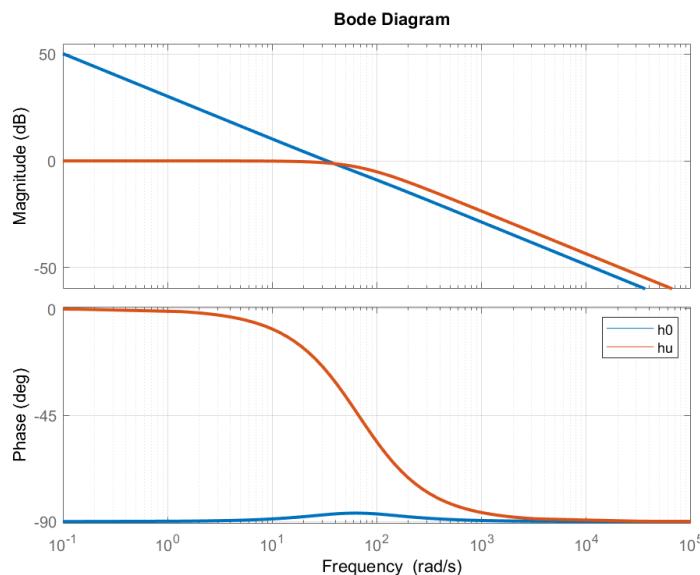
Figur 7.14: Stegresponsen til systemet i åpen sløyfe

SIMC metode for beregning av regulatorverdier baserer seg på tidsforsinkelsen til systemet. Det er mulig å sette inn tidsforsinkelse i den matematisk estimerte stegresponsen til systemet men dette fører til feil fordi den matematiske modellen ikke blir lik den fysiske.

Det kan settes opp en P-regulator som gjør at systemet blir marginalt stabilt. Dette vil gi oss muligheten til å benytte Ziegler-Nichols på den teoretiske modellen. Vi vet derimot at Ziegler-Nichols kun er en forenklet metode for å finne stabelitetsmarginer. Det settes derfor opp egne krav til stabilitetsmarginene og velge PID verdier som passer våre krav.

Krav til stabilitetsmarginer:

1. $\Delta K > 0$
2. $\psi > 0$



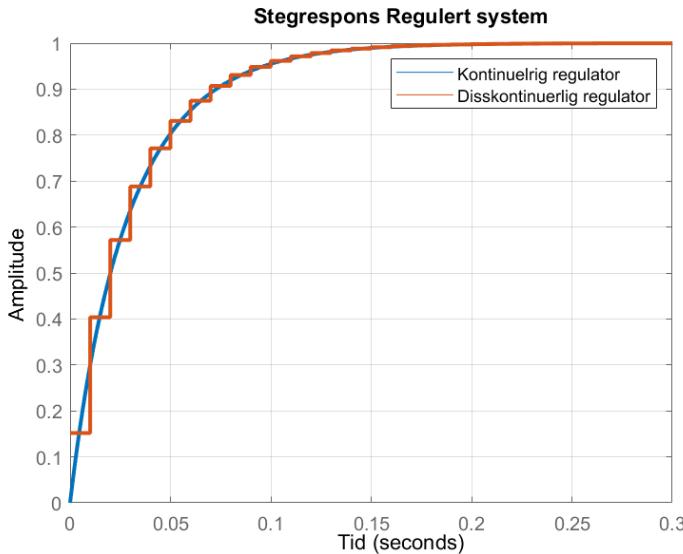
Figur 7.15: Bodediagram over systemet uten regulator h_u og systemet med PI-regulator h_r i åpen sløyfe

Ved å se på bodeplottet figur 7.15 kan vi merke oss et par viktige elementer. Det ene er at fasen aldri går under -90° som medfører at systemet ikke kan bli ustabilt ($\Delta K = \infty$). Vi har også en fase på 0° frem til knekkfrekvensen på ca 56 rad/s.

Vi velger å designe en PI-regulator for å fjerne stasjonert avvik, og transferfunksjonen for en slik regulator er vist i 7.7. Det er ikke ønskelig å implementere D-ledd fordi det kan føre til forsterket støy og fordi fasemarginen aldri blir mindre en 90° . Det er mulig å øke T_i hvis det implementeres D-ledd fordi vi kan dra fasemarginen opp rundt ω_c med et begrenset D-ledd. Dette vil føre til at vi motvirker faseforskyvningen forårsaket av $\frac{1}{1+T_1 s}$ i systemets transferfunksjon. Men som nevnt tidligere er det ikke ønskelig å benytte D-ledd på grunn av faren ved forsterket støy.

$$h_r(s) = K_p \frac{T_i s + 1}{T_i s} \quad (7.7)$$

Denne regulatoren settes opp som en ideell regulator. Vi vet ikke enda hva K_p bør være, derfor settes $K_p = 1$. Fra den estimerte transferfunksjonen til systemet kan det konkluderes at vi kan sette $T_i = \frac{1}{56}$.



Figur 7.16: Stegrespons for regulert system kontinuerlig og diskontinuerlig

Vi har frem til nå estimert systemet som en kontinuerlig transferfunksjon. Dette er korrekt i henhold til h_u , men h_r skal implementeres digitalt og vil dermed være diskontinuerlig. Et diskontinuerlig system kan oppføre seg tilsvynelatende kontinuerlig hvis samplingstiden er mye mindre en stegresponsen til h_u . Samplingstiden er avhengig av rotasjonshastigheten på motorene fordi farten måles med tiden det tar å rotere 2° . Vi vet at den laveste ønskelige hastigheten fører til en samplingsrate på $T = \frac{18757\text{puls/s}}{1000000\text{puls/s}} = 0.018757\text{s}$ og at den høyeste mulige hastigheten fører til en samplingsrate på $T = \frac{1792\text{puls/s}}{1000000\text{puls/s}} = 0.001792\text{s}$. Den laveste samplingsraten er dermed mindre en stegresponsen på ca. 0.014s, men den er ikke mye mindre slik som det kreves for å estimere systemet som kontinuerlig [14].

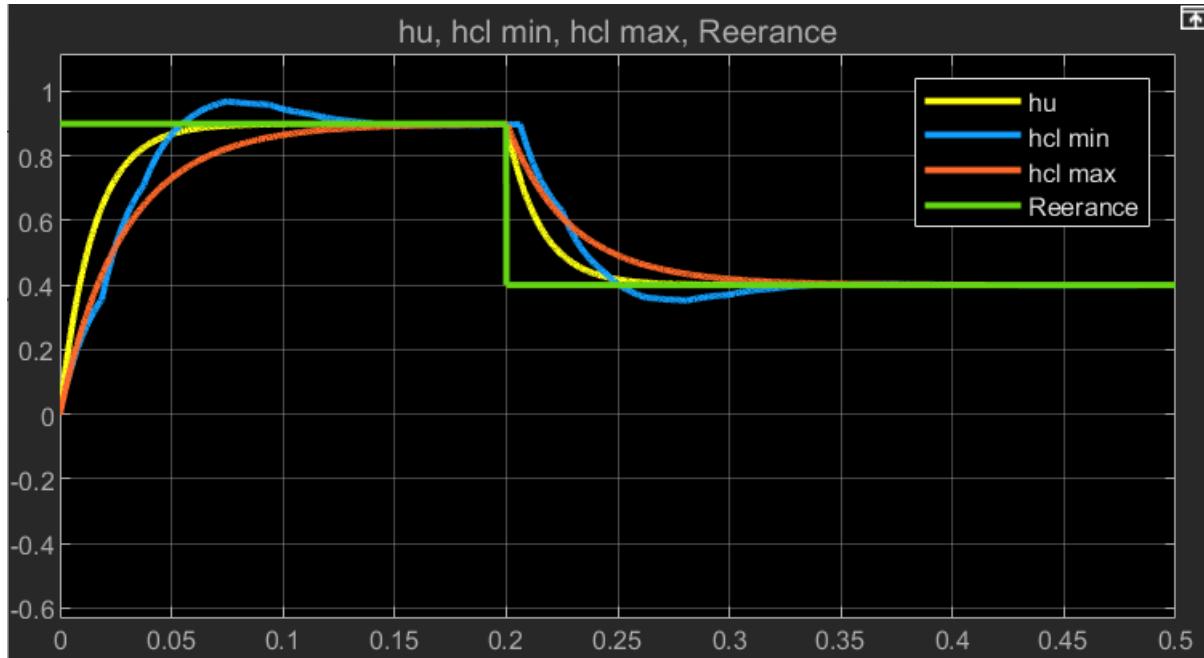
Gjennom prøving og feiling med matematiske tilnæringer kan det konstanteres at kryssfrekvensen må være mindre enn stegresponsen til systemet. Ved å sette kryssfrekvensen lav blir regulatoren roligere og oversvingninger reduseres. Gruppen har kommet frem til nye stabilitetsmarginer for å oppnå en rask men stabil regulator

Krav til stabilitet med diskret regulator:

1. $\Delta K > 0$
2. $\psi > 90$
3. $\omega_c < 66$

Stegresponsen for regulert lukket sløyfe i figur 7.16 viser en overdempet langsom regulator. Vi ser også at den diskontinuerlige regulatoren følger den kontinuerlige. Dette estimatet er imidlertid ikke perfekt fordi samplingsraten er for lav og fordi h_u blir simulert diskontinuerlig. For å simulere en mer nøyaktig stegrespons benyttes Simulink til å dele opp h_r og h_u slik at h_u blir en kontinuerlig prosess og h_r blir en diskontinuerlig prosess. Simuleringen er vist i figur 7.17.

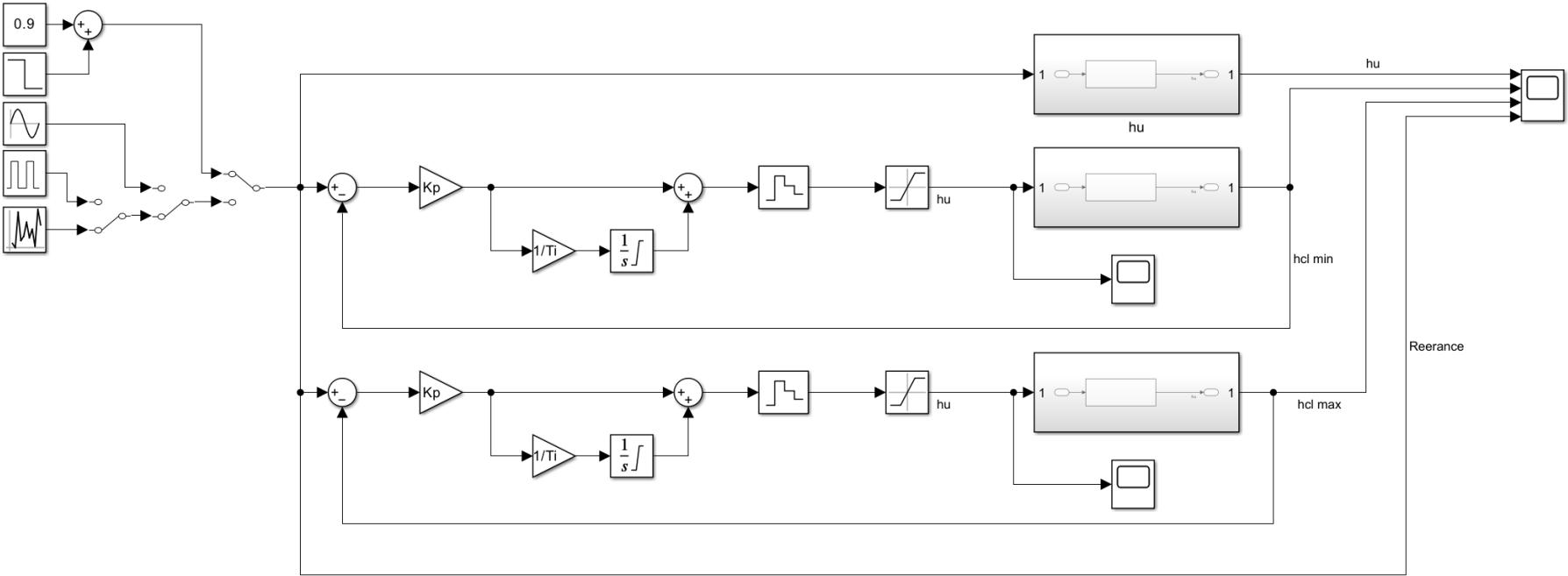
Det er stor forskjell på stegresponsen ved høyest hastighet og stegresponsen ved lavest hastighet (" $h_{cl} \text{ max}$ " og " $h_{cl} \text{ min}$ " i figur 7.17). Dette betyr at vi har begrensninger for hvor aggressivt vi kan regulere ved lave hastigheter. Det kommer også tydelig frem at stegresponsen ved høyest



Figur 7.17: Stegrespons for regulert system med kontinuerlig h_u og diskontinuerlig h_r

hastighet har en bedre tilnærming til det kontinuerlige systemet i figur:7.16. Læreboken Reguleringsteknikk beskriver også hvordan system kan bli ustabil hvis samplingstiden blir større enn den minste tidskonstanten[14]. h_{cl} min har samplingstid større enn tidskonstanten til h_u , $T > T_1 \Rightarrow 0.0187 > 0.015$ som fører til et mer ustabilt system.

systemet blir ustabilt ved lavere samplingsrate fordi dette gir motorene tid til å akkselerere til en høyere hastighet før det settes nytt ådrag. Dette er også årsaken til at større forsterkning K_p fører til at systemet blir ustabilt. En oppoverbakke vil føre til tregere stegrespons og stasjonert avvik på h_u . Det motsatte oppstår i en nedoverbakke hvor stegresponsen blir raskere og h_u får et positivt stasjonert avvik. Vi kan unngå at dette systemet går mot en marginal stabil stegrespons med å sette forsterkningen så liten at avviket ikke fører til en endring i pådraget prosentvis større enn avviket.



Figur 7.18: Simulink kode

Vi har sett på systemet som en kontinuerlig prosess og benyttet stabilitetsmarginene til å designe en regulator. Vi har oversatt denne regulatoren til et diskret system og sett på stabiliteten til den diskrete systemet ved forskjellige samplingsrater og funnet at den laveste samplingsraten fører til at vi ikke kan estimere systemet kontinuerlig når vi kjører sakte. For å sikre at regulatoren ikke blir veldig underdempet ved lave hastigheter kan vi sammenligne ω_c og T_1 . Vi vet at ω_c er frekvensen hvor systemet har 0dB forsterkning. Vi kan dermed anta at systemet ikke får kraftige oversvingninger hvis $\frac{1}{\omega_c} > T_1$ fordi dette vil føre til at systemet heller er tregere en stegresponsen til hu;

Vi har nå sett hvordan vi teoretisk kan finne en matematisk modell for regulatoren hvor vi kun benytter kjente måleverdier og tidligere pådrag til å sette nåverende pådrag. Vi skal nå se hvordan vi kan uttrykke denne matematiske funksjonen i c uten for store avvik eller avrundingsfeil. Her er det viktig å huske at variabelstørrelser stor betydning og at delestykker mister alle dessimaler og kan føre til store regnfeil. Vi må derfor sørge for at ligningen settes opp på en som hindrer overflow og hvor divisjoner ikke fører til store regnfeil.

$$s = \frac{2(z-1)}{T(z+1)} \quad (7.8a)$$

$$h_r(z) = \frac{K_p(2T_i + T)z + (T - 2T_i)}{2T_i z - 2T_i} \quad (7.8b)$$

Z-transformasjon omgjør det kontinuerlige systemet til et ekvivalent system i Z planet (7.8). Z planet er tidsdiskrete og kan benyttes til å finne en matematisk modell for diskontinuerlig regulator regulatoren (7.10).

$$h_r(z) = \frac{u(z)}{e(z)} \Rightarrow u(z) = h_r(z) \cdot e(z) \quad (7.9a)$$

$$2T_i u(z)z - 2T_i u(z) = K_p(2T_i + T)e(z)z + K_p(T - 2T_i)e(z) \quad (7.9b)$$

Regulatoren er forholdet mellom regulatorens pådrag $u(z)$ og systemets avvik $e(z)$. Det er dermed mulig å finne et uttrykk for regulatorens pådrag basert på transferfunksjonen $h_r(z)$ og avviket (7.9).

$$u(z) \cdot z^n = u(k+n) \quad (7.10a)$$

$$u(k) = u(k-1) + \frac{2T_i + T}{2T_i}e(k) + \frac{T - 2T_i}{2T_i}e(k-1) \quad (7.10b)$$

Samplingstiden T er basert på hastigheten til fartøyet og måles i antallet klokkepulser per takometerpuls. Ved å dele måleverdien på antallet klokkepulser per sekund får vi samplingstiden. Regulatoren er et regnestykke basert på forrige pådrag, forrige avvik, nåverende avvik og måleverdi fra TCB (7.11).

$$T = \frac{y(k)}{1000000} \quad (7.11a)$$

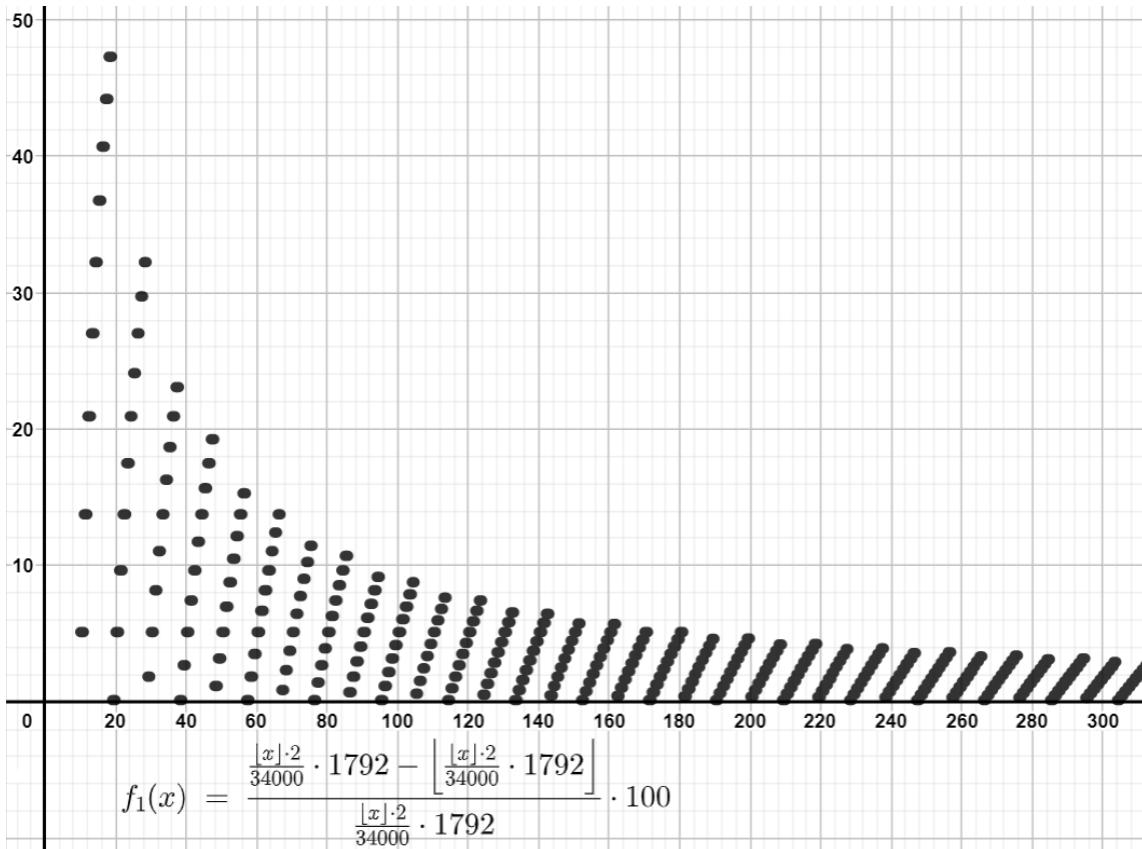
$$2T_i = 2 \cdot 0.017 = 0.034 \quad (7.11b)$$

$$u(k) = u(k-1) + e(k) - e(k-1) + \frac{y(k)(e(k) + e(k-1))}{34000} \quad (7.11c)$$

Alle verdiene i regnestykke har maksimal verdi på $\pm(2^{16} - 1)$ som blir lagret i en 32 bit integer noe som gir 31 tiljengelige dessimaler. Vi kan også se fra figur 7.19 at regnfeilen i divisjonsstykket (7.11c) ikke blir større enn 5% før avviket blir lavere en ca 190. Figur som tilsvarer 0.3% avvik og at vi ved 1% avvik ikke har mer en 1.45% regnfeil.

$$\frac{\frac{2e(k)}{34000} \cdot 1792 - \left\lfloor \frac{2e(k)}{34000} \cdot 1792 \right\rfloor}{\frac{2e(k)}{34000} \cdot 1792} \cdot 100 \quad (7.12)$$

AVR runder alltid divisjoner ned fordi desimalene ikke kan lagres. Hvor stor prosent av det eksakte svaret som representeres med desimaler. Regnestykket er forenklet til å benytte den minste måleverdien 1792 (maksimal hastighet), og $e(k)$ er avviket som representeres fra 0 til 65535 med heltall. Funksjonsuttrykket (7.12) er grafisk fremstilt i figur 7.19.



Figur 7.19: Regnfeil ved divisjon

7.4.6 XBee programmering

XBee-modulene er laget for å kunne kommunisere flere moduler sammen, alle veier imellom. I dette prosjektet behøves det bare kommunikasjon mellom to moduler med halv dupleks. Koden til styresystemet er satt opp slik at hver gang fartøyet mottar en styrekommando, sender mikrokontrolleren data tilbake over UART med informasjon som brukes til kartet i HMIen. På denne måten vil det ikke være praktisk mulig at informasjon blir sendt to veier samtidig. XBee har også en innebygd kø-funksjon som automatisk venter dersom den mottar signal, selv om modulen får beskjed om å sende. Dette gir dobbel sikring mot tap av data.

For at modulene kan kommunisere kreves samme kanal og samme PAN-ID [15]. Kanalene som kan velges er 11-26 [19]. Kanal 15-19 overlapper den mest brukte wifi-kanalen, kanal 6 [12]. Det ble derfor bestemt at dette prosjektet kjører kanal 20 på XBee-modulen. Denne ble vurdert til å ha minst sannsynlighet for å interferere med wifi.

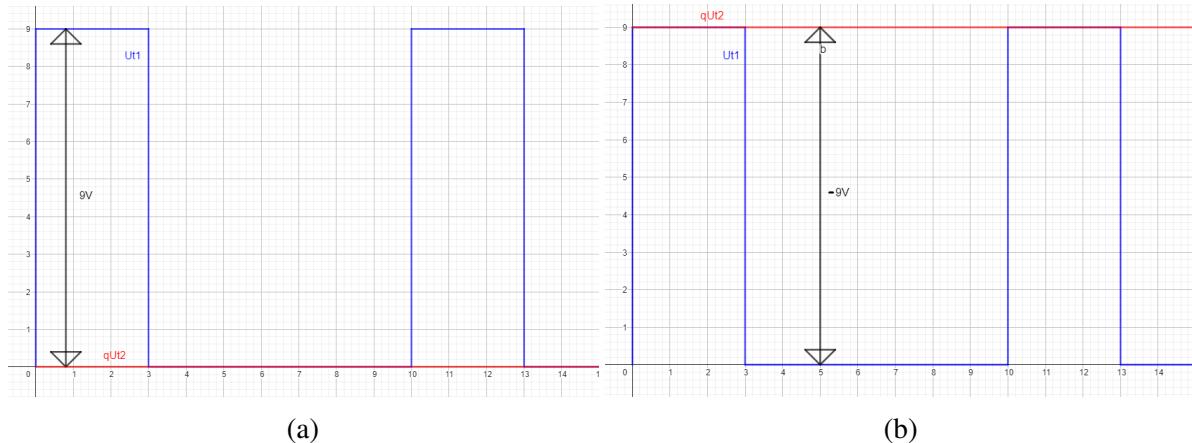
Standard PAN-ID for XBee er 0x3332, og det kan velges en tilfeldig ID mellom 0x0000 og 0xFFFF. Tanken var å ikke bruke standarden, det ble derfor valgt å bruke 0xB923. En helt tilfeldig verdi langt unna standarden, noe som gir en liten sannsynlighet for at noen i nærområdet har valgt samme ID.

Det ansees heller ikke som nødvendig å bruke hele det 32-bits serienummeret som adressering, men kringkasting vil unngås. Det blir derfor benyttet en forkorte 16-bits destinasjonsadresse som programmeres inn i modulen. Adressene som er valgt er 0xF57A og 0xF57B, og hver enkelt XBee vil adressere sin informasjon til den andre, og bare den.

Oppsummert så vil det altså ved hjelp av kanalvalg, PAN-ID og adressering være lite sannsynlig med forstyrrelse fra andre Zigbee- og Wifi-enheter, samt at andre lytter på overført data. Baudraten blir satt til 115200 da dette er det største mulige for XBee.

7.4.7 L298N Motor Driver

Effekten tillført motorene styres av et PWM signal per motor. Dette PWM signalet sendes inn på in1 til venstre motor og in3 til høyre motor. Disse forsterkes og sendes ut på motorene på ut1 og ut3. Retningen på motorene bestemmes ved å bytte polaritet på in2 og in4 slik at ut2 og ut4 bytter polaritet. På figur 7.20a er det tydelig at det oppstår 9V spenningsfall over motorene. Figur 7.20b viser at det ikke bare oppstår -9V spenningsfall over motoren men også at PWM signalet har motsat virkning.

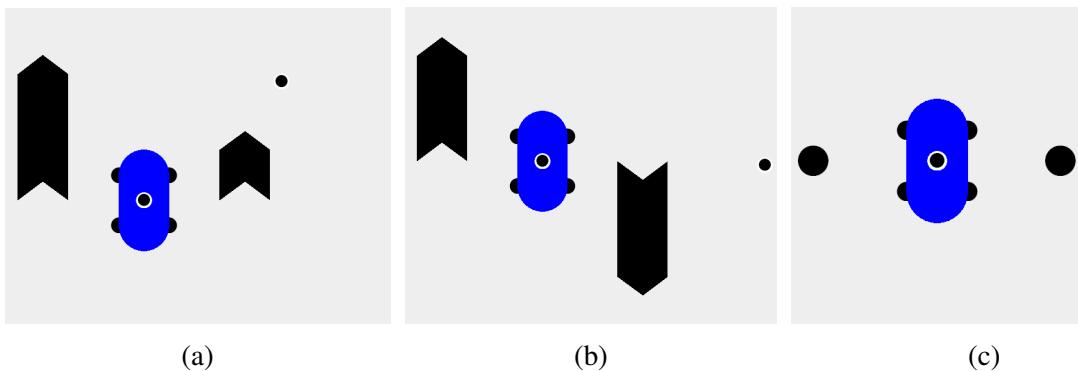


Figur 7.20: PWM motor styring

7.5 Styresystem

7.5.1 Brukergrensesnitt

HMI (Human Machine Interface) er en samlebetegnelse for grensesnittet mellom mennesker og maskiner og blir ofte benyttet til å beskrive knappepaneler i industriell sammenheng. Det utvikles i denne oppgaven et mer sofistikert brukergrensesnitt som er en underkategorie av HMI.



Figur 7.21: Fartøy i brukergrensesnitt

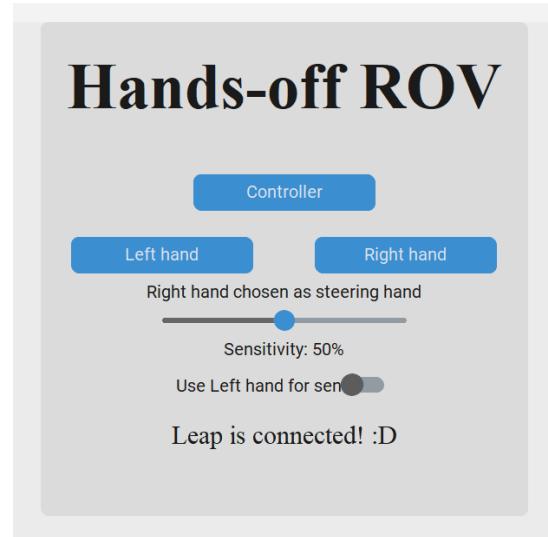
Brukergrensesnittet er satt opp med intuitive figurer og beskrivende tekst [24]. Ved design av gode brukergrensesnitt er det viktig å være konsist og tydelig. Menyer skal være selvforklarende og brukeren må forstå hva som skal gjøres uten eksterne hjelpeemidler. Figurer skal være enkle, og tekst bør unngås fordi tegn og symboler gir raskere oversikt og er enklere å forstå.

Grensesnittet er fordelt i tre deler, hvor to er grafiske med minimalt av tekst. Selve styringen av fartøyet representeres med former og figurer. Her settes det opp bilde av en bil og det tegnes piler rundt bilen figur 7.21. Pilene representerer hastigheten til beltene. Når fartøyet står stille blir pilene sirkler for å representere stopp figur 7.21c.

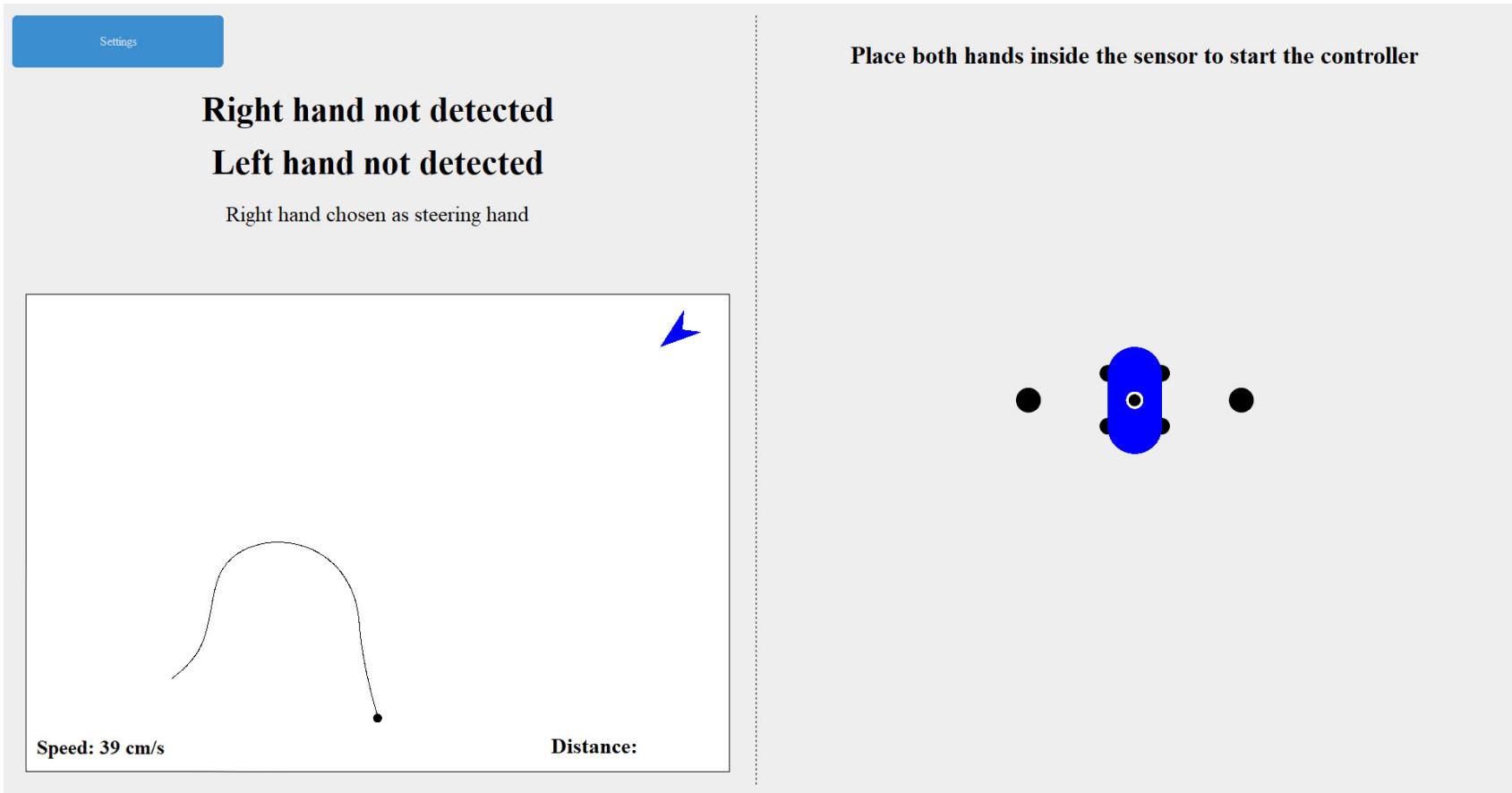
Over figuren av fartøyet står det beskrevet med store bokstaver hva brukeren må gjøre for å starte kjøring figur 7.23. Dette inkluderer å fortelle brukeren at hånden må plasseres over kameraet for å starte styringen, eller dersom styresystemet ikke finner en hånd eller begge hendene i Leap Motion interaksjonsboksen. Ved siden av feilmeldings- og informasjons-teksten står det beskrevet hvilke innstillinger brukeren har valgt. Dette er hvilken hånd som benyttes til styring og om denne er detektert, åpen eller lukket.

Ved siden av figurene for hastighetsstyring er det satt opp et firkantet kart. Dette kartet inneholder en pil som representerer fartøyets retning relativt til startposisjonen. Kartet inneholder også en linje som beskriver fartøyets kjørte strekning. Hvis fartøyet rygger, forblir pilen pekende i samme retning som fartøyets fremover-retning. Dermed kan kartet og pilen kombinert fortelle brukeren hvordan fartøyet beveger seg, slik at det er mulig å manøvrere uten å fysisk se fartøyet.

Dersom Leap Motion enheten ikke er tilkoblet er det ikke mulig å åpne det grafiske brukergrensesnittet, og programmet åpnes dermed direkte i innstillingsmenyen. I denne menyen er det en tilbakemelding om enheten er tilkoblet eller ikke. Figur 7.22 viser et eksempel på innstillingsmenyen når enheten er tilkoblet. I innstillingsmenyen er det også mulig for brukeren å tilpasse styresystemet etter behov. Det er mulig å bytte mellom høyre- eller venstrehandsstyring, endre sensitiviteten, og sette opp sensitivitet basert på avstanden mellom tommel og pekefinger på den passive hånden.



Figur 7.22: Innstillingsfane i HMI



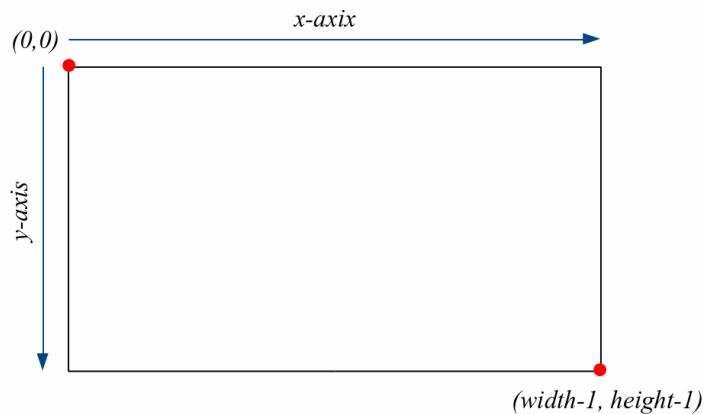
Figur 7.23: HMI controller

7.5.2 Koding av HMI

Brukergrensesnittet er oppbygd av python-modulen CustomTkinter. CustomTkinter er en tilpasset versjon av Python-biblioteket Tkinter som gjør det mulig å lage grafisk brukergrensesnitt. Custom-versjonen gjør det mulig å lage mer tilpasset grensesnitt med et penere utseende. Som nevnt før består grensesnittet av to faner. Innstillingsfanen og Kontrollerfanen.

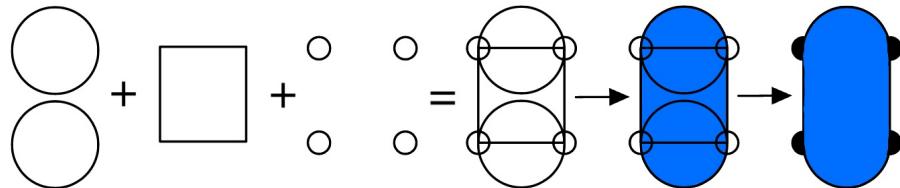
Innstillingsfanen Innstillingsfanen består av en base som blir plassert øverst midt på skjermen uansett hvor stor skjermen er. Her vil det alltid forekomme en overskrift, knapper og informasjon som vist og forkart rundt 7.22. Kodingen bak det er et eksempel på hvor likt CustomTkinter fungerer sammenlignet med html- og css-koding. Det blir plassert knapper, slidere og header, og det kan settes på forskjellige stiler. Knappene har også tilhørende funksjoner, og det skjer en event hver gang en knapp blir trykket.

Kontrollerfanen Kontrollerfanen består også av en base på samme måte som innstillingsfanen, men til forskjell så er hele basen her dekt med et lerret (canvas). Lerretet består av et koordinatsystem som gjør det mulig å tegne hva som helst, hvor man måtte ønske. 7.24 viser at origo befinner seg øverst i venstre hjørne, med positiv x-akse horisontalt mot høyre og positiv y-akse vertikalt nedover. Størrelsen på skjermen til bruker må også betraktes her. Derfor er det laget en variabel for bredden og høyden som endrer seg hver gang lerretet åpner seg. Disse variablene blir også endret av en event som kjører dersom brukeren skulle ha endret på størrelsen på sitt vindu mens programmet kjører. Ut ifra bredden og høyden på vinduet har det også blitt definert en ny origo som er midt på skjermen. Ny origo befinner seg alltid ved koordinater som tilhører halvparten av bredden og halvparten av høyden i forhold til lerretets origo. Disse har i koden henholdsvis fått navnene "senter_x" og "senter_y". På denne måten vil disse variablene alltid være i midten av skjermen selv om bredden og høyden vil variere ut ifra både skjermens størrelse og brukerens preferanse. I tillegg er det laget en variabel med navn "coords_unit". Dette er en variabel som alltid er med som faktor når noe tegnes i lerretet. Variabelen har alltid en verdi lik $\frac{1}{30}$ av høyden. Med variabelen med som faktor vil alt som tegnes i lerretet holde seg til samme størrelse i forhold til høyden. Dette innebærer, men er ikke begrenset til knapper, tekster og tegninger.



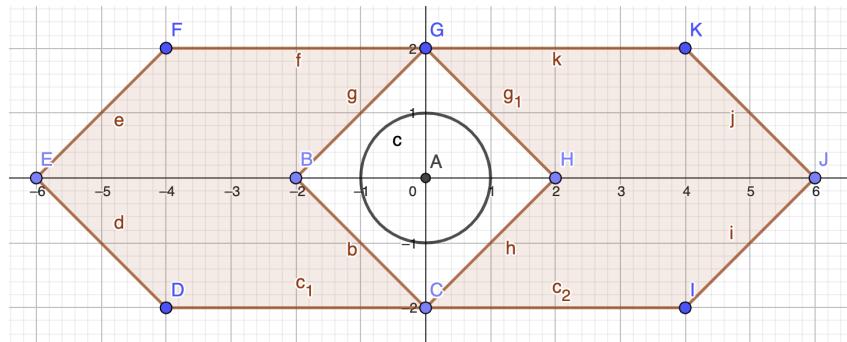
Figur 7.24: Lerretets koordinatsystem

Fartøyet Siden lerretet er basert på et koordinatsystem, betyr det at det kan benyttes geometriske verktøy for å tegne. Figur 7.25 viser hvordan fartøyets utseende i brukergrensesnittet er oppbygd. Kroppen til fartøyet består av to store sirkler og et kvadrat. Hjulene består av fire små sirkler med senter i hvert av kvadratets hjørner. For å få fargene som ønsket tegnes hjulene med svart fyll først i koden, deretter kroppen med blått fyll. Størrelsen av fartøyet bestemmes i forhold til en "coords_unit", noe som gir litt utregning på koordinatene hvor de geometriske figurene skal plasseres.



Figur 7.25: Fartøyets oppbygning i HMI

Fartspiler Pilene ved siden av fartøyet i HMilen kalles fartspiler. Pilene defineres ved å tegne et polygon i koordinatsystemet til lerretet. Figur 7.26 illustrerer de tre forskjellige stadiene kan være for fartspilen. Er farten lik 0, vil fartspilen være en sirkel med radius 1. Er farten positiv eller negativ vil pilen tegnes vertikalt henholdsvis oppover eller nedover. Når farten varierer flyttes punktene D, E, F, eller I, J, K avhengig av positiv eller negativ fart. Punktenes avstand og hvor mye de flyttes er også bestemt av en "coords_unit". Det er til enhver tid kun en av de tre figurene som er synlig i HMilen.



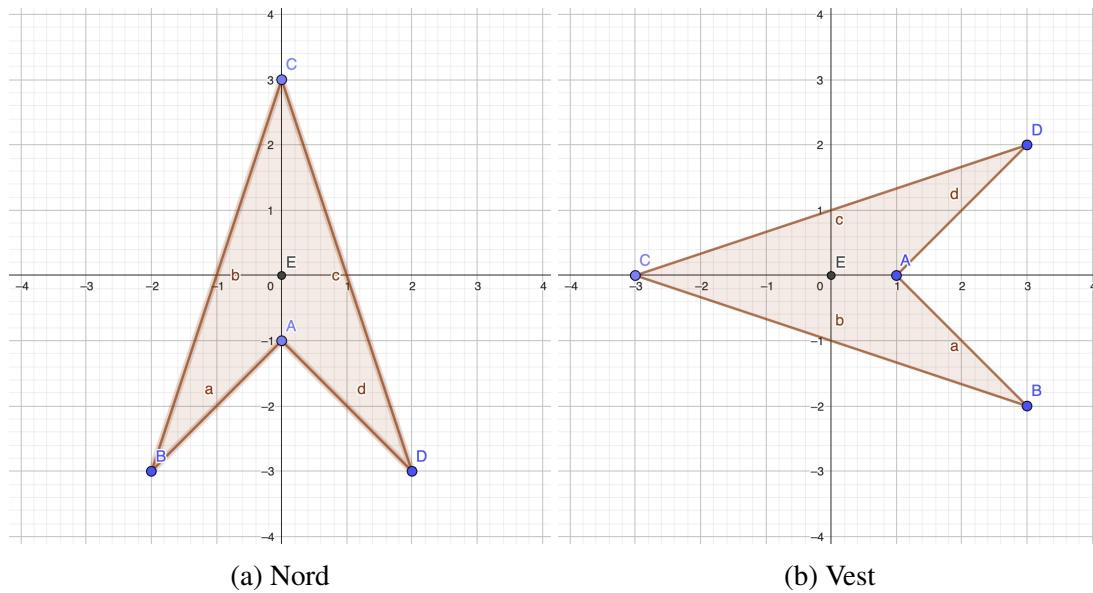
Figur 7.26: Polygonene som danner fartspiler

Håndbevegelsesindikator I figur 7.23 vises en svart sirkel med hvit ytterlinje i midten av fartøyet. Dette er en indikator som flyttes tilsvarende til posisjonsvektoren som oppfattes fra Leap Motion enheten. I koden er det lagt inn to slike, en til hver hånd. Hver indikator består av en hvit sirkel med radius lik $\frac{1}{3}$ av en "coords_unit", etterfulgt av en svart sirkel med radius lik $\frac{1}{4}$ av en "coords_unit". Dette gir indikatoren høy kontrast og er lett synlig uansett hvilken bakgrunn den havner over.

Kart Kartet i grensesnittet er avgrenset til et rektangel med hvit bakgrunn. Det er plassert en sirkel som markerer startposisjon for fartøyet. Det tegnes deretter en strek fra startposisjonen

til neste posisjon for fartøyet. Posisjonen mottas hver kjøring av koden og blir lagret i en vektor i python. Siden kjøringene går rundt 100 ganger per sekund så gjør dette strekene så korte at resultatet blir en jevn bane av kjørt løype.

Kompass Kompasset ser tilsynelatende ut som det er en del av kartet, men det er like mye et selvstendig polygon i lerretet som er plassert på innsiden av kartet. Polygonet er tegnet av fire koordinater og formet som en pil. Siden komasspilen blir tegnet som et polygon med definerte koordinater i hvert hjørne, kan det benyttes matriserotasjon for å rotere denne. Det er akkurat det som er blitt gjort her. Figur 7.27 inneholder to skjermbilder fra et rutenett som illustrerer hvordan komasspilen er blitt implementert i lerretet. Kompasset roteres rundt punkt E, et punkt som er i midten av pilen. Det skal bemerkes at for å kunne rotere kompasset på denne måten må koordinatene til komasspilen først flyttes til lerretets origo, ganges med en rotasjonsmatrise som roterer rundt origo, og deretter flytte kompasset tilbake til sin posisjon. Det blir derfor litt matematikk i koden, dette er vist i figur 7.28. Vinkelen som er benyttet i rotasjonsmatrisen er vinkel β som er beskrevet i kapittel 7.4.2. Ved oppstart peker pilen nordover på kartet som vist i figur 7.27a, og vil deretter rotere etter fartøyets rotasjon. Hvis fartøyet roterer $\frac{\pi}{2}$ radianer peker pilen vestover slik som i figur 7.27b.



Figur 7.27: Polygonet som danner en komasspil

```

1 def Rotate_CompassArrow(self):
2     rotate = [[np.cos(rov.beta_tot), -np.sin(rov.beta_tot)], [np.sin(rov.beta_tot), np.cos(rov.
3         ↪ beta_tot)]]
4     self.map_compass_vect_rotate = np.dot(rotate, self.map_compass_vect)
5     self.canvas.coords(self.map_compass,
6         self.frame_senter_x - self.coords_unit*3 + self.map_compass_vect_rotate[0][0] , self.
7             ↪ frame_senter_y - self.coords_unit*2.5 + self.map_compass_vect_rotate[1][0],
8         self.frame_senter_x - self.coords_unit*3 + self.map_compass_vect_rotate[0][1] , self.
    ↪ frame_senter_y - self.coords_unit*2.5 + self.map_compass_vect_rotate[1][1],
9         self.frame_senter_x - self.coords_unit*3 + self.map_compass_vect_rotate[0][2] , self.
    ↪ frame_senter_y - self.coords_unit*2.5 + self.map_compass_vect_rotate[1][2],
10        self.frame_senter_x - self.coords_unit*3 + self.map_compass_vect_rotate[0][3] , self.
    ↪ frame_senter_y - self.coords_unit*2.5 + self.map_compass_vect_rotate[1][3])

```

Figur 7.28: Rotering av komasspil

7.5.3 Python

For å øke hastigheten på programmeringen benyttes Python som programmeringsspråket til styresystemet. Dette utviklingsprosessens hastighet fordi Python er et språk på et høyere nivå. Det implementeres pakker fra eksterne kilder for å gjøre det enklere å designe brukergrensesnitt og for å redusere koden som kreves for å behandle data fra Leap Motion Controller.

CustomTkinter hentes inn til design av brukergrensesnitt. Denne pakken er designet for å enkelt sette opp HMI på en datamaskin og fungerer på både mac og windows, men har forskjellige funksjoner avhengig av operativsystemet. Det blir designet brukergrensesnitt basert på funksjonaliteten som eksisterer i windows.

Python kan benyttes som et objekt orientert programmeringsspråk. Dette utnyttes for å øke kodekvaliteten med å sette opp klasser. Det eksisterer allerede en klasse som hentes inn fra driveren til Leap Motion Controller.

Klassen Leap.Controller hentes inn og objektet inneholder all dataen fra driveren. Gjennom SDK-en finnes det eksempler på mulige måter å sette opp dette objektet og det velges et ferdig designet program som setter opp en klasse for databehandling av data fra driveren. objektet Listener legges til som lytteobjekt i objektet Leap_controller. Dette fører til at vår egen definerte kode kan kjøres gjennom driveren. Klassen Listener inneholder callback funksjoner som kjører ved spesifiserte hendelser. Koden for behandling av data fra Leap Motion Controller og videresending over XBee legges inn i on_frame callback funksjonen til Listener. Denne koden kjører hver gang det mottas en datapakke fra Leap Motion Controller.

Objektet ROV inneholder informasjon om fartøyet. Dette inkluderer hastighet, distanse kjørt, vinkelen til fartøyet og posisjonen. Denne dataen blir beregnet fra antallet takometerpulser på motorene og lagres til objektet i on_frame funksjonen til Listener objektet. Informasjonen lagret i ROV objektet benyttes til å kartlegge ruten som er kjørt og tegne denne i brukergrensesnittet.

Alle objektene hentes inn i main funksjonen figur 7.29 som kun kjører hvis if testen registrerer at programmet kjøres som hovedprogram.

```

1 def main():
2     global rov
3     rov = ROV()
4     global hmi
5     hmi = HMI()
6
7     # Create a sample listener and controller
8     listener = SampleListener()
9     leap_controller = Leap.Controller()
10
11    # Have the sample listener receive events from the controller
12    leap_controller.add_listener(listener)
13
14    hmi.mainloop()
15
16    # Remove the sample listener when done
17    leap_controller.remove_listener(listener)
18
19 if __name__ == "__main__":
20     main()

```

Figur 7.29: Python main

For kommunikasjon med XBee skrives data til en COM port med tillkoblet seriel driver. Driv-

eren kobles opp mot windows og data kan sendes over TX med ser.write og hentes inn fra RX med ser.read. Koden i figur 7.30 viser funksjonen som benyttes til å sende hastighetskommandoer og oppdatere brukergrensesnittets hastighets piler.

```
1 def ROV_drive(self, left_number, right_number):
2     # Sending speed from 0-200 to xbee and drawing coresporonding arrows.
3     left_control = struct.pack("B", int(left_number))
4     right_control = struct.pack("B", int(right_number))
5     ser.write(start + left_control + right_control)
6     HMI.Draw_arrows_movement(hmi, left_number, right_number)
```

Figur 7.30: ROV drive

7.5.4 Leap Motion Controller

Enheten er som beskrevet tidligere et ferdig produkt. Produktet er designet for VR og SDK er designet deretter[2, 20, 31, 32]. Dette fører til mye unødvendig prosessering fra vårt ståsted, men gir muligheten for integrering av VR styring ved et senere stadiet av utviklingen. Driveren er basert på C++ og oversettes til et objekt i Pythen gjennom LeapC++.dll og LeapPython.pyd. På grun av .dll filen kan styresystemet kun benyttes på windows.

8 Resultater Og Diskusjon

Det oppsto utfordringer under hele utviklingsprosessen. I forbindelse med disse hindringene utførte gruppen tester for å løse problemene. Mot slutten av utviklingsprosessen fungerte demonstrasjonen bedre og gruppen fikk mulighet til å sette opp funksjonstester.

8.1 LeapMotion testing

Leap motion er ikke like nøyaktig når hånden er lukket. Gruppen ønsket originalt å designe styringen slik at fartøyet kjører når hånden er lukket, grunnet konsekvenser som kan oppstå når brukeren står med en hevet utstrakt hånd. Dette har vist seg å gi mer unøyaktige koordinater enn når hånden er åpen. Det virker som om at Leap Motion Controlleren leser bedre med åpen hånd. Dette fordi det er god kontrast mellom refleksjonen i huden og bakgrunnen som er lengre vekke (taket), og fordi omrisset av hånden med spredte fingre er enklere å analysere en omrisset av knyttet neve. Dette gir kameraet muligheten til å placere alle fingrene mer nøyaktig som fører til at håndflatens senter placeres korekt. Når hånden er lukket kan den se mer ut som en kule uten store kontraster og tydelige former. Vi har dermed valgt å lage en "nødstopp funksjon" som stopper bilen hvis styre hånden lukkes eller kameraet mister hånden.

8.2 Testing og utregning

Det har blitt gjort tester for å finne ut hvilken måte som er best for å kartlegge fartøyets bevegelser. For å kunne lage et nøyaktig kart med oversikt over hvor bilen har kjørt, er det flere alternativer.

En mulighet er å bruke et akselerometer som monteres på fartøyet. Et akselerometer er en sensor som måler akselerasjon ved å måle den relative endringen i bevegelsesmengde, som er en kombinasjon av akselerasjonen og gravitasjonskraften, som påvirker sensoren.

For å finne fart og posisjon må denne dataen integreres. Dette kan være utfordrende på grunn av støy i sensoren, og integrasjon kan føre til feil over tid. I tillegg kan akselerometeret gi mye data som må overføres over XBee. Dette kan ta opp mye båndbredde, noe som kan sinke kommunikasjonen og gå ut over hyppigheten av styringen til fartøyet.

Vi oppererer på et todimensjonalt plan i denne demonstrasjonen og kan dermed ignorere gravitasjon og høyde i posisjoneringen. Det kan derfor benyttes metoder som er designet til å måle retning eller rotasjon

Rotasjon kan måles ved hjelp av gyroskop. Et gyroskop baserer seg på treghetsmomentet i et roterende legeme og kan måle rotasjonen påført dette legemet ved å måle endringen i kretene som påvirker det. Gyroskop gir høy nøyaktighet og krever mindre data en akselerometer, men dataen må behandles kontinuerlig for å opprettholde nøyaktigheten.

En absolut retningsmåler kan implementeres ved hjelp av et kompass. Kompass baserer seg på jordens magnetiske felt og er derfor følsome for elektromagnetisk støy. De gir derimot en absolutt retningsmåling og muligheten til å synkronisere andre retningsmålere.

en annen absolutt posisjonsmåler er GPS (Global Positioning System) som benytter satellitter til å bestemme posisjonen. GPS krever åpen sikt til satellitter eller begrenset med hindringer.

Gruppen har jobbet innendørs og GPS vil ikke kunne kommunisere med satellitter fra testlokalene.

Motorene i fartøyet har som beskrevet tidligere innebygde takometere. Dette gir pulser annen-hver grad for rotasjon. Fartspulsene gir direkte informasjon om farten og avstanden fartøyet har beveget seg. Dette er mer nøyaktig og enklere å implementere enn å bruke akselerometerdata.

En svak side med å bruke forflytningen per belte i steden for akselerometer er at den faktiske bevegelsen av fartøyet ikke blir målt. Dette fordi endring i friksjon ikke måles slik at styresystemet ikke kan dedektere om fartøyet eller sitter fast. Den matematiske modellen er basert på en perfekt verden hvor hvert belte forflytter seg like langt som målingene tilliser. I virkeligheten kan ett eller begge beltene skli mot underlaget og rotasjonen kan dermed bli andeledes en matematisk beregnet

To tester ble utført for å teste om den matematiske modellen samsvarer med virkeligheten. Avstand per puls ble målt og sammenlignet med matematisk beregnet lengde. Derrefter ble rotasjon ved forskjellige hastigheter målt og sammenlignet med matematisk beregnet rotasjon.

Avstandstest								
Målt avstand (cm)			Pulser			Avstand per puls (cm/puls)		
L	R	Snitt	L	R	Snitt	Snitt		
28	27,5	27,75	502	486	494		0,056174089	
29,8	29,5	29,65	511	496	503,5		0,058887786	
27,6	27,4	27,5	506	493	499,5		0,055055055	
28,6	27,5	28,05	506	503	504,5		0,055599604	
28,7	28	28,35	500	490	495		0,057272727	
28,3	27,5	27,9	507	496	501,5		0,055633101	
28	27,5	27,75	505	499	502		0,055278884	
		28.13			500		0,056271	

Table 3: Avstandstest

Fra tabell 3 kan vi se at gjennomsnittlig avstand per puls er ca 0.0563cm mens den beregnede avstanden er 0.0646cm. Beregningen er basert på radiusen til beltene og små unøaktigheter i diametermålet fører til stor feil på avstanden. Det er dermed mer nøyaktig å benytte målt avstand i stedenfor beregnet avstand.

Hvis vi skalerer avstanden kjørt på begge beltene likt til vinkelen beregnet ut fra dette forblir den samme. Fordi vinkelberegningen er basert på forholdet mellom beltene kjørte distanse. Dette betyr at vi kan teste vinkelberegningen uavhengig av estimatet for avstand.

For å få et bedre overblikk over hvordan systemet fungerer gjøres det noen tester ved forskjellige hastigheter figur 4. Fartskomandoene er pådraget i % som sendes til motordriveren. Radiusen er sirkelen som kan tegnes langs banen til det inderste beltet. Den høyre motoren defineres som R og den venstre som L. Det som kan observeres fra tabellen er at større differanse på beltene hastigheter gir kraftigere sving. Det kommer også frem at den beregnede rotasjonen er lik den målte, hvis det inkluderes en målefeil på $\pm 2^\circ$ grunnet måle utstyr.

Fartøyet er konstruert av LEGO. LEGO er bøyelig og gir derfor etter under kjøring. Batteriene veier nok til å deformere bærekonstruksjonen, som endrer egenskapene til fartøyet, Med å vinkle beltene oppover og øke avstanden mellom beltene. selve beltene er også laget av LEGO som betyr at disse har slark. Dette betyr at motorene kan rotere litt frem og tillbake uten at

Rotasjonstest						
Fartskommando		Radius (cm)	Grader kjørt	Pulsecount		Grader beregnet
L	R			L	R	
40	100	23,4	68	463	845	70.4
40	100	23,4	65	467	845	69.6
40	100	23,4	65	465	837	68.5
40	100	23,4	68	470	840	68.2
40	100	23,4	66	476	846	68.2
30	100	15,3	87	395	848	83.5
30	100	15,3	84	388	844	84
30	100	15,3	85	396	847	83.1
30	100	15,3	85	394	845	83.9
30	100	15,3	86	411	835	78.1
20	100	10,1	70	239	638	73.5
20	100	10,1	74	242	639	73.1
20	100	10,1	76	267	637	68.2
20	100	10,1	78	262	647	70.9
20	100	10,1	76	241	635	72.6
0	100	1	75	0	433	79.8
0	100	1	76	0	438	80.7
0	100	1	73	8	436	78.9
0	100	1	77	5	442	80.5
0	100	1	73	2	433	790.4

Table 4: Rotasjonstest

fartøyet flytter på seg. Målingene er derfor forurensset av eksterne kilder som stammer fra dårlig mekanisk utforming.

8.3 XBee kommunikasjon

Det er viktig at kommunikasjonen opprettholdes under kjøring for å oppnå korekte målinger og avstandsberegninger. Å oppnå hurtig kommunikasjon med rom for videre utvikling vil redusere arbeidet ved implementering av mer funksjonalitet. trådløs kommunikasjon testes for å forsikre at det ikke oppstår problemer i sluttfasen.

UART kommunikasjonen som ble implementert først baserte seg på en baudrate på 9600 som er relativt tregt. Ved observere kommunikasjonen med oscilloskop var det mulig å observere at kommandoer på 100Hz bruker opp 30% av potensiell bånbredde. Matematisk blir dette $\frac{9600bit/s}{1+8+1} = 960Byte/s$ fordi det benyttes start og stop bit før og etter hver Byte. Dette fører til maksimalt 9.6 Byte per datapakke ved 100Hz kommunikasjon. Kommunikasjonen krever 3Byte per datapakke. baudrate på 115200 blir implementert for å gjøre det mulig å øke størrelsene på datapakkene større en 9 Byte.

Xbee kommuniserer over UART med baudrate på 9600. Dette testes ved å koble sammen to ESP32 med ledninger og sende data mellom dem med UART, for så å koble opp to XBee i stedetfor direkte tillkobling med ledere. Det blir observert at XBee kommunikasjonen har samme funksjonalitet som ledninger. XBee benytter en broadcast struktur som standard. Alle XBee

enheter i nerheten mottar dataen som om det var direkte tilkobling. For å unngå at kommunikasjonen påvirkes av støy eller skaper støy for andre, settes XBee enhetene til å kommunisere privat. Direkte kommunikasjon oppnås med spesifikke adresser på en mindre populær frekvens. Dermed vil data kun mottas av den korekte XBee enheten. Baudraten endres også til 115200 for å samsvare med tidligere implementert UART kommunikasjon med fartøyet.

SPI, I2C og USART er alle kommunikasjon metoder som kan benyttes av XBee. SPI med sin master slave struktur er en mer praktisk kommunikasjonsmetode fordi det i denne demonstrasjonen kun skal sendes og mottas tallverdier. I2C har de samme egenskapene som SPI for overføring av tallverdier. Denne standarden er bedre egnet til kommunikasjon med mange slaver men kan implementeres nesten like enkelt. Det kreves nemlig flere mekaniske komponenter ved I2C kommunikasjon for å unngå støy. Argumentet for å starte med UART kommunikasjon er fordi dette benyttes av debugger på AVR og fordi det kan enkelt settes opp USART kommunikasjon i Python. Det er derfor naturlig å vurdere å endre kommunikasjonen til SPI.

8.4 Signalproblematikk på Motorer

Motorene som benyttes har takometer med styrekrets. Transistorene benyttet til å deddekkere endringer i lysstyrken på de lys sensitive diodene har stor motstand til jord. Dette fører til at pull upp motstanden på LLC fungerer som en spenningsdeler og at signalet til AVR ikke blir 0V ved logisk lavt signal. Forsterkeren på styrekretsen til LEGO motorene har ca $10K\Omega$ til jord som fører til 2.5V på HV siden av LLC. Ved å sette HV til GND blir pullupmotstanden på HV siden til en pull down motstand. 0V inn på LLC blir nå 0V til AVR og 5V inn på LLC blir nesten 3.3V til AVR.

Kommander til EV3 fra Raspberry PI Som beskrevet tidligere kan EV3 styres på flere måter [6]. En av disse metodene er å sende kommander over USART i form av json objekter. Disse objektene samsvarer med predefinert kode i operativsystemet til EV3 og fører til endringer på utgangene eller avlesninger fra sensorer. Objektene er store kompliserte og abstrakte. Disse er designet til å bli benyttet av drivere innebygget i andre programmer slik som LEGO EV3 software pakken. Å dekode disse kommandoene for å finne en måte å styre EV3 gjennom Python krever dermed mye tid.

Det velges dermed å skifte ut EV3 med en egen styreenhet. Grunnes tråd responsid på EV3 og kompleksiteten med å designe egen driver til EV3. Selve legobilens beholdes fordi dette er et ferdig konstruert fartøy med motorer. Gruppen vurderte muligheten for å bygge egen radiostyrt bil. Det ble valgt bort fordi lego motorene har posisjonsmåling og er robust designet for å unngå varmgang ved stillstans [17].

8.5 Brukergrensesnitt

Før utviklingen av brukergrensesnittet startet ble det gjennomført digitalt grunnkurs i UX design. Viktigheten av et selvforklarende styresystem med intuitive figurer og lettleselig tekst ble gjordt klart for gruppen. Dette la grunnlaget for brukergrensesnittet beskrevet i gjennomføringskapittelet 7.5.

For å sikre at brukergrensesnittet er selvforklarende for andre en de som har designet det ble medstudenter hentet inn for testing. Mange antok at rotasjon av fartøyet kom av rotasjon av

håndflaten. Dette er selvfølgelig mer intuitivt for fartøyet som er designet til demonstrasjonen fordi rotasjon oppnås med å endre hastigheten på beltene. Fartøyet kan rotere på stedet og bevegelse til siden er dermed ikke logisk når fartøyet blir stasjonert og roterer. Tanken bak dette designvalget var at dette ligner mer på styring med en spak hvor frem og bak er hastighet og sidelangs er rotasjon. Det eksisterer radiostyrte biler som benytter en linneer spak for hastighet og en roterende spak for rotasjon. Denne styringen er mer intuitiv for demonstrasjonsfeltet i denne oppgaven.

Hvis brukergrensesnittet skal benyttes til styring av andre typer fartøy må styrekommandoene endres til mer intuitive kommander. En blindtest hvor tilfeldige personer blir plassert foran brukergrensesnittet vil gi muligheten til å observere hvordan noen som aldri har benyttet demonstrasjonen tror den burde fungere. Det er mulig å dra begrensede antakelser fra nåværende observasjoner. Disse antakelsene tillsier følgende styrekommandoer.

- Håndflate delvis apen under kjøring
- hånd fjernes eller ristes bort når bruker føler tap av kontroll
- hånd forflyttes i en retning fører til at fartøy flyttes likt
- rotasjon eller vipping fører til lik rotasjon eller vipp på fartøyet

Disse grunnleggende prinsippene vil gi brukeren en bedre evne til å visualisere hvordan hen ønsker fartøyet skal bevege seg. Dette fordi brukeren kan sammenligne styresystemet med andre dagligdagse situasjoner og fartøyet kan bli som en utvidelse av ens egen arm.

PI-regulator En drone kan stå stille selv om det blåser svak vind. Og kan fly like raskt mot vinden som med vinden under rolig styring. Det ble observert at friksjon i beltenes konstruksjon og ujevnhet på testflaten førte til at fartøyet fikk forskjellig hastighet på beltene med samme pådrag på motorene. Ved å regulere hastighetene på motorene kan det garantere at begge motorene kjører med samme hastighet og at fartøyet kjører rett frem. Dette oppnår den samme jevne følelsen som droneflyvning.

for å gjøre styresystemer allsidig ble det bestemt at regulatoren skal designes på fartøyet. Styresystemet trenger ikke regulere fartøyet men sender kun hastigheten fartøyet skal opprettholde. AVR mikrokontrolleren er basert på en 8 bit datamaskin, som betyr at store komplekse utregninger kan bli utfordrige. Alle utregninger foregår i binærtallsystemet og alle desimaltall avrundes nedover. Denne programeringsutfordringen har vist seg å være en større tidstyp en forventet.

Primitive regulatorer ble implementert først. Her besto regulatoren av et lite P-ledd og et større I-ledd som ga pådraget. Denne regulatoren fungerte men var ikke basert på matematisk bevitste reguleringsverdier. Den var også veldig treg og det dominerende I-leddet førte til at regulatoren brukte lang tid før stegresponsen ble observert i pådraget. Reguleringsverdiene samsvarer med verdiene som ble beregnet senere men utregningene i binærtallsystemet førte til dårlig kodekvalitet og lite oversikt.

For å øke kodekvaliteten ble det benyttet Z-transformasjon for å designe tidsdiskre regulator. Denne regulatoren er matematisk optimalisert og utregningene foregår i desimalsystemet.

Funksjonen består av flere summasjoner og multiplikasjoner, men bare en divisjon. Med å redusere antallet konstanter og flytte rundt på variablene er det mulig å redusere regnfeilen ved divisjonen til ubetydelig liten.

Den teoretiske regulatoren er ikke implementert på fartøyet grunnet liten tid til feilsøking.

9 Miljøkonsekvensanalyse

Formålet med denne analysen er å identifisere potensielle miljøpåvirkninger som knytter seg til prosjektet. I utgangspunktet er det benyttet elektriske komponenter og enheter som oppdragsgiver allerede hadde kjøpt inn. Det eneste vi har bestilt er et kretskort med enkelte tilhørende komponenter, dette ble bestilt fra Kina. I tillegg bruker fartøyet oppladbare batterier som oppdragsgiver hadde fra før. Med bakgrunn fra dette vil FNs bærekraftsmål kunne betraktes[34]. Figur 9.1 viser et overblikk over målene som er drøftet i denne analysen.

FNs bærekraftsmål 7 figur 9.1a handler om ren energi og økonomisk vekst. Bruk av oppladbare batterier kan bidra til å redusere forbruket av engangsbatterier. I tillegg blir batteriene ladet opp av strøm fra det norske strømnettet, hvor store deler kommer fra fornybar energi.

Bærekraftsmål 9 figur 9.1b handler om industri, innovasjon og infrastruktur. Prosjektet i denne oppgaven kan bidra positivt til dette bærekraftsmålet med tanke på innovasjon av fjernstyringsteknologi eller utvikling av industri. Produksjon av kretskort i Kina vil imidlertid kunne ha negative konsekvenser for blant annet menneskers helse. Dette får prosjektet også innom bærekraftsmål nummer 12 figur 9.1c som omhandler ansvarlig forbruk og produksjon. Kretskortet som ble bestilt ble nøyne planlagt, og bestilt sammen med en stor bestilling til flere studenter hos UiT. Det vurderes derfor til et ansvarlig forbruk, selv om det er lang transportvei fra Kina til Norge.



(a) Bærekraftsmål 7

(b) Bærekraftsmål 9

(c) Bærekraftsmål 12

Figur 9.1: FNs Bærekraftsmål

Samlet sett så kan prosjektet både ha positive og negative påvirkninger for miljøet. Det blir likevel konkludert med at prosjektet er innenfor fornuftens grenser med tanke på FNs bærekraftsmål.

10 Konklusjon

Å demonstrere et styresystem basert på dataen fra en Leap Motion Controller, og benytte dette systemet til styring av et fjernstyrt fartøy. Var målet med denne oppgaven. Demonstrasjonen som er utviklet fullfører denne problemstillingen.

Gruppen har utviklet et fjernstyrt fartøy som kan styres med kontaktløs styring. Det er integrert intuitivt brukergrensesnitt med veiledende anvisninger og feilmeldinger som hjelper brukeren med å sette seg inn i systemet. Fartøyet har god rekkevidde og lang levetid på batteriene. som nevnt i resultat og diskusjon er det store muligheter for forbedring.

Etter første fase hadde fullførte demonstrasjonen alle kravene til fartøyet og styresystemet. I andre fase satte ble det definert større krav, og det har vist seg å være utfordrende å fulføre alle disse. Det er utviklet PI-regulator og implementeringen av denne er igangsatt. Det er utviklet retningsmåling, hastighetsmåling og lengdemåling og dette er satt sammen til et kart som viser den kjørte strekningen. Dette kartet er ikke nøyaktig nokk og videre arbeid må gjøres for å oppnå bedre nøyaktighet. Det er utviklet kretskort til styresystemet på fartøyet og dette oppfyller kraven i fase to men det er flere forbedringsmuligheter her som kan redusere strømforbruket og gjøre systemet mer fysisk robust og brukervennlig.

10.1 Videre arbeid

PI-regulator må implementeres og testes. Det må testes at regulatoren oppfyller kravene til stabilitet og at det oppnås en hurtig nokk respons.

Kart over kjørt strekning må forbedres slik at vinkelen på fartøyet blir representeret med større nøyaktighet.

Kretskort med innebyggede batteriholdere eller oppladbart batteri med ladekrets må designes og testes.

Mulighet for styring av drone må utvikles. Her kan det settes opp et objekt som inneholder kommandoer som opp, ned, frem, bak, høyre, venstre, rotasjon og vipping slik at disse kommandoene kan sendes til en rekke forskjellige fartøy som kan benytte disse slik det passer i forhold til fartøyenes utforming og fysiske begrensinger.

11 Litteraturliste

- [1] Syed Anwaarullah. *Leap Motion Controlled Remote Search and Disposal Robot*. 2015. URL: <https://www.hackster.io/Anwaarullah/leap-motion-controlled-remote-search-and-disposal-robot-4cb662> (visited on 01/23/2023).
- [2] ALAN DAVIS. *Leap Motion Getting Started*. 2014. URL: <https://blog.leapmotion.com/getting-started-leap-motion-sdk/> (visited on 04/30/2023).
- [3] SparkFun Electronics. *Bi-Directional Logic Level Converter Hookup Guide*. URL: <https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide> (visited on 02/18/2023).
- [4] SparkFun Electronics. *BSS138 N-Channel Logic Level Enhancement Mode Field Effect Transistor*. 2005. URL: <http://cdn.sparkfun.com/datasheets/BreakoutBoards/BSS138.pdf> (visited on 02/18/2023).
- [5] SparkFun Electronics. *Logic Level Bi-Directional*. 2013. URL: http://cdn.sparkfun.com/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf (visited on 02/18/2023).
- [6] Christoph Gaukel. *ev3-dc (Use python3 to program your LEGO Mindstorms EV3)*. 2020. URL: <https://ev3-dc.readthedocs.io/en/latest/index.html> (visited on 01/18/2023).
- [7] Digi International Inc. *Xbee®/XBee-PRO® 802.15.4 (Legacy) Professional Kit Getting Started Guide*. 2015. URL: <https://hub.digi.com/dp/path=/support/asset/xbee-s1-802154-legacy-starter-kit-getting-started-guide/> (visited on 02/18/2023).
- [8] Digi International Inc. *XBee®/XBee-PRO® RF Modules - 802.15.4 - v1.xEx [2009.09.23]*. 2009. URL: <https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf> (visited on 02/18/2023).
- [9] DigiInternational Inc. *XBee/XBee-PRO S1 802.15.4 (Legacy)*. 2018. URL: https://www.digi.com/resources/documentation/digidocs/pdfs/90000982.pdf?fbclid=IwAR2v8GxmX8ENxqt8-6yGAli6Sad9tbcCAalAyyJ5-6a3O6NvU_X4eziOgU (visited on 03/13/2023).
- [10] Microchip Technology Inc. *AVR128DB28/32/48/64*. 2023. URL: <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/AVR128DB28-32-48-64-DataSheet-DS40002247.pdf> (visited on 02/18/2023).
- [11] Microchip Technology Inc. *AVR128DB48 Curiosity Nano Hardware User Guide*. 2020. URL: <https://ww1.microchip.com/downloads/en/DeviceDoc/AVR128DB48-Curiosity-Nano-HW-UserG-DS50003037A.pdf> (visited on 02/18/2023).
- [12] Digi International Inc. *Channels, Zigbee*. 2018. URL: https://www.digi.com/resources/documentation/digidocs/90001537/references/r_channels_zigbee.htm (visited on 05/08/2023).
- [13] Kim Aleksander Hammer Iversen. *Valg av modell for tidligfaseaktiviteter som skaper en strukturert produktutviklingsprosess*. 2017. URL: https://nmbu.brage.unit.no/nmbu-xmlui/bitstream/handle/11250/2464530/Masteroppgave_Kim_Aleksander_Hammer_Iversen_2017_endelig.pdf?sequence=1 (visited on 03/29/2023).
- [14] G.B. Jens, A. Trond, and A.F. Bjarne. *Reguleringssteknikk*. 2016. URL: <https://folk.ntnu.no/tronda/regtek-kurs/> (visited on 04/18/2023).

- [15] JIMBLOM and TONI_K. *Exploring XBeees and XCTU*. URL: <https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu/configuring-networks> (visited on 05/08/2023).
- [16] N. Jinji et al. *Teleoperation Control for Mobile Robot via Leap Motion Device*. 2022. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10023804&tag=1> (visited on 05/08/2023).
- [17] LEGO. *LEGO® MINDSTORMS® Education EV3*. 2023. URL: <https://education.lego.com/en-au/product-resources/mindstorms-ev3/downloads/developer-kits> (visited on 05/08/2023).
- [18] Millennium Circuits Limited. *PCB TRACE WIDTH CALCULATOR*. 2023. URL: <https://www.mclpcb.com/blog/pcb-trace-width-vs-current-table/> (visited on 02/28/2023).
- [19] MetaGeek. *Why channels 1, 6 and 11?* 2023. URL: <https://www.metageek.com/training/resources/why-channels-1-6-11/> (visited on 05/08/2023).
- [20] Zachariah Peterson. *Implementation of Leap Motion Based RC Car Controller*. 2016. URL: https://ijiset.com/vol4/v4s2/IJISET_V4_I02_25.pdf (visited on 05/08/2023).
- [21] Zachariah Peterson. *PCB Trace Width vs. Current Table for High Power Designs*. 2019. URL: <https://resources.altium.com/p/pcb-trace-width-vs-current-table-high-voltage-design> (visited on 05/08/2023).
- [22] RandomNerdTutorials. *RandomNerdTutorials*. URL: <https://randomnerdtutorials.com/> (visited on 05/08/2023).
- [23] Sanjay and Arvind Seshan. *EV3 tutorials Moving Straight*. 2019. URL: https://ev3lessons.com/en/ProgrammingLessons/beginner/py-Turning.pdf?fbclid=IwAR0ccj85SBW3FVZMXAMMeB7-zlQhrguYbv3Ba-_wAQ5Wc2ctfDWjuuY2DCK (visited on 03/13/2023).
- [24] Teo Yu Siang. *What is Interaction Design?* 2021. URL: <https://www.interaction-design.org/literature/article/what-is-interaction-design> (visited on 02/01/2023).
- [25] Safaric Stanislav and Malarić Krešimir. *ZigBee wireless standard*. 2006. URL: <https://ieeexplore.ieee.org/document/4127535> (visited on 05/12/2023).
- [26] STMicroelectronics. *Control Stepper Motor with L298N Motor Driver & Arduino*. 2022. URL: <https://lastminuteengineers.com/stepper-motor-l298n-arduino-tutorial/> (visited on 02/18/2023).
- [27] STMicroelectronics. *DUAL FULL-BRIDGE DRIVER*. 2000. URL: <https://www.st.com/resource/en/datasheet/l298.pdf> (visited on 02/18/2023).
- [28] Espressif Systems. *ESP32Series Datasheet*. 2023. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (visited on 05/08/2023).
- [29] MBA Skool Team. *Product Evolution*. 2021. URL: <https://www.mbaskool.com/business-concepts/marketing - and - strategy - terms/16612 - product - evolution.html> (visited on 04/04/2023).
- [30] Sigurd Trageton. *Innovasjon og markedsføring (SR-SRL vg2)*. 2023. URL: <https://ndl.no/nb/subject:1:a7c337ca-d3b6-492f-ace2-b05c45f54e93/topic:1:c332fc97-ba55-460b-ab03-9ef4b9a63077/> (visited on 03/29/2023).
- [31] Ultraleap. *Ultraleap For Developers*. 2023. URL: <https://developer.leapmotion.com/> (visited on 01/18/2023).
- [32] Ultraleap. *Ultraleap/Developer*. 2022. URL: <https://developer-archive.leapmotion.com/documentation/python/index.html> (visited on 01/18/2023).

- [33] Wikipedia. *H-bridge*. 2023. URL: <https://en.wikipedia.org/wiki/H-bridge> (visited on 04/29/2023).
- [34] Nicholas Wilkinson. *FNs bærekraftsmål*. 2020. URL: <https://www.fn.no/om-fn/fns-bærekraftsmaal> (visited on 05/08/2023).

12 Vedlegg

- A Prosjektfleksjon
- B Forenkling av mattematiske formler
- C PID Beegninger

A Prosjektrefleksjon

Dette dokumentet defineres som et vedlegg i prosjektrapportmalen. Gruppen har valgt å inkludere referanser til den originale rapporten i teksten for å koble valg og utfordringer til oppgaven.

Valg under utviklingsprosessen

Gruppen ble raskt overveldet av mange valg. Hvilke komponenter burde benyttes hvilke teknikker som kunne hjelpe og hvilke metoder som ville føre til den beste demonstrasjonen. For å lage en sti blant jungelen av mulige løsninger ble mange dyktige hoder på fakultetet kontaktet og planleggingsmøter ga et bedre overblikk over hvilke løsninger som var mulige hvilke utfordringer disse medførte men også hvilke styrker som kom med dem. I løpet av forprosjekt ble en generell plan utformet som vist i figur 7.2. Denne planen ga en ramme for demonstrasjonen som gjorde prosessen med valg av utstyr og metode enklere.

Innhenting av inspirasjon var viktig for å se mulige løsninger og velge den som passet best for gruppens problemstilling. Flere andre demonstrasjoner benytter Raspberry PI til å styre fartøyene [1, 16, 20]. Dette fordi Raspberry PI er enkelt å programmere. De fleste demonstrasjonene benytter også Python eller lignende høynivå programmer til å behandle Leap Motion Controller data før det videresendes til Raspberry PI.

Gruppen valgte bort Raspberry PI av et par grunner. Raspberry PI er en datamaskin og bruker mye strøm for å utføre enkle arbeidsoppgaver. Raspberry PI har trådløs kommunikasjon over WIFI som krever mye prosessor kraft for å sende og motta data. Dette betyr at en ekstern trådløs kommunikasjonsmodul må benyttes for å forenkle kommunikasjonen. Raspberry PI har ikke dedikerte perifere moduler til signalgenerering som betyr at PWM generering og takometer signal behandling må foregå på ekstern modul. Når det ble klart at EV3 ikke kunne benyttes som motorkontroller ble det bestemt å benytte AVR128DB48 curiosity nano som hovedsentral på fartøyet. Som beskrevet i kapittel 7.4.4 benyttes AVR til alt utenom trådløs kommunikasjon.

For å operette Trådløs kommunikasjon ønsket gruppen å benytte utstyr som oppdragsgiveren kunne supplere. Alternativene som var tillgjengelig var dermed XBee, Raspberry PI og i senere tid Bluetooth low energy. Gruppen hadde allerede konstantert at Raspberry PI ikke ville bli benyttet og valgte XBee kommunikasjon.

Leap Motion Controller er komponenten som legger grunnlaget for denne oppgaven. Det er ikke essensielt og benytte Leap Motion Controller for å tilfredsstille kravspesifikasjonene og problemstillingen. Dette fordi det eksisterer flere produkter som kan benyttes til hånddeteksjon. Oppdragsgiveren ønsket imidlertid å benytte Leap Motion Controller fordi disse kunne bli benyttet i undervisning. Gruppen har benyttet SDK V4 for å hente data fra enheten fordi SDK V2 og V3 kun er kompatibel med Python 2 og fordi SDK V5 er designet opp mot VR applikasjoner og er ikke kompatibel med Python. Det er mulig å designe egen konverterer som henter inn C++ objektet og oversetter dette til et Python objekt. Dette vil være en tungvindt prosess og det er dermed enklere å oversette nåværende program til C++, Unity eller Unreal.

Det er et par aspekter med demonstrasjonen som burde blitt implementert anderledes. Retnings- og posisjons-måling var en morsom utfordring og gjennomføring og testing av valgt metode var

lærerikt. Den er derimot unøyaktig og trenger mye raffinering før den kan benyttes til å finne den eksakte posisjonen. Andre metoder for posisjonsmåling har blitt utviklet over flere tiår og er raffinert ned til kompakte pålitelige systemer. Gruppen burde dermed valgt ferdig programert flight controller til å finne posisjonen. Dette hadde vært mye enklere å implementere og komponentene kunne blitt benyttet i andre typer fartøy. En annen utfordring har vært hastigheten på LEGO motorene. Disse motorene er saktegående og passer fint til utviklings- og opplæringsformål, men grunnet problematikken med lav samplingsrate på hastigheten ville raskere motorer med gir vært enklere å regulere.

Fremdriften

Under arbeidet med forprosjektet ble det satt av lite tid til planlegning grunnet andre fag og eksamener. Dette førte til at planleggingsfasen skle ut over starten av januar måned. Forsinkelsene forplantet seg litt ut mot påsken, men mue hardt arbeid hindret at det gikk ut over rapportskrivningen. fremdriften vises i figur 7.1

Den praktiske delen av oppgaven ble spist bit for bit. Gruppen hadde planleggingsmøter med omtrentlig 10 dagers mellomrom. Disse møtene ble benyttet til planlegning av videre arbeid og refleksjon på arbeidet som er utført. For å motivere medlemmene med arbeidet ble det definert mål som skulle være ferdig før neste møte. På denne måten ble hvert møte en refleksjon på sprinten som hadde vært og planlegningen av neste spurt.

Dagboken er en remse med dato og notater, og egner seg derfor ikke til rapporten. Grunnen til at dagboken ikke er pent strukturert er fordi den er et verktøy som ikke er ment for offentligheten. Det legges dermed inn en punktliste over rekkefølgen på utført arbeid.

- Første grove utkast av forprosjekt dokumentet.
- Første utkast forprosjekt ferdig
- Forprosjekt levert
- Samle data
- Velge utstyr
- Funksjonsteste utstyr
- Velge komponenter ut fra funksjonstester
- Sette opp systemet etter grov system skisse fra forprosjekt
- Plan for systemet i sin helhet med valgte komponenter
- AVR test
- XBee test
- Anskaffe SDK V4 til Leap Motion Controller
- Programere generelt styresystem i Python
- Kommunikasjon fra Python til AVR

- Mekanisk montasj og test av fartøy
- Innhenting av informasjon innen UX design
- XBee programering
- AVR programering
- Programering av brukergrensesnitt
- Mekanisk montasje og test av takometerpulser
- Toveiskommunikasjon med AVR
- Design av kart i brukergrensesnitt
- AVR programering av TCB
- PID test
- Planlegningsmøter og design av PI-regulator
- Rafinering av brukergrensesnitt
- Funksjonstesting av ferdig system
- Teste teoretiske beregninger opp mot virkeligheten
- Ferdigstille rapport mal
- Raportskriving
- Produksjon av poster fil
- Raport første utkast
- Presentasjonsdag
- Raport endelig utkast
- Ferdigstille rapport

Det har vært et langt og lærerikt semester, og den viktigste lerdommen gruppen tar med til neste prosjekt er viktigheten av struktur plan. Og en god rapport struktur, slik at arbeidet med utvikling av rapport mal ikke blir like omfattende.

B Forenkling av mattematiske formler

$$\beta = \alpha \cdot 2 = \arctan(\tan(\alpha)) \cdot 2 = \arctan\left(\frac{a}{b}\right) \cdot 2 = \arctan\left(\frac{a}{\tan(\theta) \cdot a}\right) \cdot 2 \quad (\text{B.1a})$$

$$= \arctan\left(\frac{\frac{R}{2}}{\tan(\theta) \cdot \frac{R}{2}}\right) \cdot 2 = \arctan\left(\frac{\frac{R}{2}}{\frac{e}{d} \cdot \frac{R}{2}}\right) \cdot 2 = \arctan\left(\frac{\frac{R}{2}}{\frac{e}{\frac{R-L}{2}} \cdot \frac{R}{2}}\right) \cdot 2 \quad (\text{B.1b})$$

$$= \arctan\left(\frac{\frac{R-L}{2}}{e}\right) \cdot 2 = \arctan\left(\frac{\frac{R}{2}}{\frac{e}{\frac{R-L}{2}} \cdot \frac{R}{2}}\right) \cdot 2 \quad (\text{B.1c})$$

$$= \arctan\left(\frac{\frac{R-L}{2}}{\sqrt{W^2 - d^2}}\right) \cdot 2 = \arctan\left(\frac{R-L}{\sqrt{W^2 - (\frac{R-L}{2})^2} \cdot 2}\right) \cdot 2 \quad (\text{B.1d})$$

$$= 2 \arctan\left(\frac{R-L}{\sqrt{4W^2 - (R-L)^2}}\right) \quad (\text{B.1e})$$

$$L = 0 \quad (\text{B.2a})$$

$$R = 0.0005627 \quad (\text{B.2b})$$

$$W = 0.175 \quad (\text{B.2c})$$

$$\beta = \arctan\left(\frac{0.0005627}{\sqrt{0.175^2 - \left(\frac{0.0005627}{2}\right)^2} \cdot 2}\right) \cdot 2 \quad (\text{B.2d})$$

$$= 0.003215 \text{ rad} \quad (\text{B.2e})$$

$$\beta_{tot} = \sum_0^n \beta_n \quad (\text{B.3a})$$

$$d_2 = \cos(\theta) \cdot \frac{W}{2} \quad (\text{B.3b})$$

$$= \cos(\arctan(\tan(\theta))) \cdot \frac{W}{2} \quad (\text{B.3c})$$

$$= \cos\left(\arctan\left(\frac{e}{d}\right)\right) \cdot \frac{W}{2} \quad (\text{B.3d})$$

$$= \cos\left(\arctan\left(\frac{\sqrt{w^2 - d^2}}{d}\right)\right) \cdot \frac{W}{2} \quad (\text{B.3e})$$

$$= \cos\left(\arctan\left(\frac{\sqrt{w^2 - \left(\frac{R-L}{2}\right)^2}}{\frac{R-L}{2}}\right)\right) \cdot \frac{W}{2} \quad (\text{B.3f})$$

$$C = L + d_2 \cdot 2 \quad (\text{B.4a})$$

$$= L + \cos\left(\arctan\left(\frac{\sqrt{w^2 - \left(\frac{R-L}{2}\right)^2}}{\frac{R-L}{2}}\right)\right) \cdot \frac{W}{2} \cdot 2 \quad (\text{B.4b})$$

$$= L + \cos\left(\arctan\left(\frac{\sqrt{W^2 - \left(\frac{R-L}{2}\right)^2}}{\frac{R-L}{2}}\right)\right) \cdot W \quad (\text{B.4c})$$

$$\cos(\arctan(x))) = \frac{\sqrt{1+x^2}}{1+x^2} \quad (\text{B.5a})$$

$$C = L + \frac{\sqrt{1 + \left(\frac{\sqrt{W^2 - \left(\frac{R-L}{2} \right)^2}}{\frac{R-L}{2}} \right)^2} \cdot W}{1 + \left(\frac{\sqrt{W^2 - \left(\frac{R-L}{2} \right)^2}}{\frac{R-L}{2}} \right)^2} \quad (\text{B.5b})$$

$$= L + \frac{\sqrt{1 + \left(2 \frac{\sqrt{W^2 - \left(\frac{R-L}{2} \right)^2}}{R-L} \right)^2} \cdot W}{1 + \left(2 \frac{\sqrt{W^2 - \left(\frac{R-L}{2} \right)^2}}{R-L} \right)^2} \quad (\text{B.5c})$$

$$= L + \frac{\sqrt{1 + \left(\frac{\sqrt{4W^2 - (R-L)^2}}{R-L} \right)^2} \cdot W}{1 + \left(\frac{\sqrt{4W^2 - (R-L)^2}}{R-L} \right)^2} \quad (\text{B.5d})$$

$$= L + \frac{\sqrt{1 + \frac{4W^2 - (R-L)^2}{(R-L)^2}} \cdot W}{1 + \frac{4W^2 - (R-L)^2}{(R-L)^2}} \quad (\text{B.5e})$$

$$= L + \frac{\sqrt{1 + \frac{4W^2}{(R-L)^2} - \frac{(R-L)^2}{(R-L)^2}} \cdot W}{1 + \frac{4W^2}{(R-L)^2} - \frac{(R-L)^2}{(R-L)^2}} \quad (\text{B.5f})$$

$$= L + \frac{\frac{\sqrt{4W^2}}{\sqrt{(R-L)^2}} \cdot W}{\frac{4W^2}{(R-L)^2}} \quad (\text{B.5g})$$

$$= L + \frac{2W}{(R-L) \cdot \frac{4W^2}{(R-L)^2}} \cdot W \quad (\text{B.6a})$$

$$= L + \frac{2W^2}{\frac{4W^2}{R-L}} \quad (\text{B.6b})$$

$$= L + \frac{R-L}{2} \quad (\text{B.6c})$$

$$= \frac{R-L+2L}{2} \quad (\text{B.6d})$$

$$= \frac{R+L}{2} \quad (\text{B.6e})$$

$$X = C \cdot \sin(\beta_{tot}) \quad (\text{B.6f})$$

$$Y = C \cdot \cos(\beta_{tot}) \quad (\text{B.6g})$$

C PID Beegninger

$$h_r(s) = K_p \frac{T_i s + 1}{T_i s} \quad (\text{C.1a})$$

$$(\text{C.1b})$$

$$s = \frac{2(z-1)}{T(z+1)} \quad (\text{C.2a})$$

$$h_r(z) = K_p \frac{T_i \left(\frac{2(z-1)}{T(z+1)} \right) + 1}{T_i \left(\frac{2(z-1)}{T(z+1)} \right)} \quad (\text{C.2b})$$

$$= \frac{K_p(T_i 2(z-1) + T(z+1))}{T_i 2(z-1)} \quad (\text{C.2c})$$

$$= \frac{K_p(2T_i + T)z + (T - 2T_i)}{2T_i z - 2T_i} \quad (\text{C.2d})$$

$$= \frac{u(z)}{e(z)} \Rightarrow u(z) = h_r(z) \cdot e(z) \quad (\text{C.3a})$$

$$u(z) = \frac{K_p(2T_i + T)z + (T - 2T_i)}{2T_i z - 2T_i} e(z) \quad (\text{C.3b})$$

$$u(z) \cdot (2T_i z - 2T_i) = (K_p(2T_i + T)z + (T - 2T_i))e(z) \quad (\text{C.3c})$$

$$2T_i u(z)z - 2T_i u(z) = K_p(2T_i + T)e(z)z + K_p(T - 2T_i)e(z) \quad (\text{C.3d})$$

$$u(z) \cdot z^n = u(k+n) \quad (\text{C.4a})$$

$$2T_i u(k+1) - 2T_i u(k) = K_p(2T_i + T)e(k+1) + K_p(T - 2T_i)e(k) \quad (\text{C.4b})$$

$$u(k+1) = \frac{2T_i}{2T_i} u(k) + \frac{K_p(2T_i + T)}{2T_i} e(k+1) + \frac{K_p(T - 2T_i)}{2T_i} e(k) \quad (\text{C.4c})$$

$$u(k) = u(k-1) + \frac{2T_i + T}{2T_i} e(k) + \frac{T - 2T_i}{2T_i} e(k-1) \quad (\text{C.4d})$$

$$T = \frac{y(k)}{1000000} \quad (\text{C.5a})$$

$$2T_i = 2 \cdot 0.017 = 0.034 \quad (\text{C.5b})$$

$$u(k) = u(k-1) + \left(\frac{T}{2T_i} + 1 \right) e(k) + \left(\frac{T}{2T_i} - 1 \right) e(k-1) \quad (\text{C.5c})$$

$$u(k) = u(k-1) + \left(\frac{\frac{y(k)}{1000000}}{0.034} + 1 \right) e(k) + \left(\frac{\frac{y(k)}{1000000}}{0.034} - 1 \right) e(k-1) \quad (\text{C.5d})$$

$$u(k) = u(k-1) + \frac{y(k) \cdot e(k)}{34000} + e(k) + \frac{y(k) \cdot e(k-1)}{34000} - e(k-1) \quad (\text{C.5e})$$

$$u(k) = u(k-1) + e(k) - e(k-1) + \frac{y(k)(e(k) + e(k-1))}{34000} \quad (\text{C.5f})$$