PROSJEKTRAPPORT

HØGSKOLEN I ÅLESUND
Aalesund University College

TITTEL :
STOR PLATTFORM

KANDIDATNUMMER(E): IVAN FLATVAL ,HÅKON EIKREM, EDVARD JOHAN DYB, ØRJAN GJELSETH			
DATO: 29.11.2013	FAGKODE: IE203512	FAGNAVN: Industrielle Styresystemer	
STUDIUM: AU2 / Automatiseringsteknikk		ANT SIDER/VEDLEGG: 25/13	

VEILEDER(E) :
Ottar Osen

SAMMENDRAG:

I denne rapporten blir prosjektoppgaven stor plattform presentert. Denne oppgaven var en av mange som gruppene i faget industrielle styresystemer kunne velge mellom. Oppgaven skulle løses med bruk av utstyr tilgjengelig på skolen. Det ble satt av 8 uker til dette prosjektet. Oppgaven skulle bestå av et automatisert system med blant annet et styreprogram, remote I/O, kommunikasjon med tredjepart og visualisering.

Plattformen består av 3 elektromotorer, som styrer hvert sitt hjørne av toppen. Motorene blir styrt av 3 frekvensomformere som igjen blir styrt av en PLS.

Målene gruppen ville oppnå var, styring av plattform ved hjelp av en mobilapplikasjon, et bilspill, et vilkårlig bølgemønster og ved manuell kjøring. Bevegelsene av plattformen skulle også simuleres ved hjelp av en 3D modell i sann tid.

Med disse målene var problemstillingene mange og lærerike. Det første som måtte komme på plass var kommunikasjon mellom PLS og frekvensomformere ved hjelp av profibus. En kommunikasjonsapplikasjon ble nødvendig for håndtering av data mellom de forskjellige systemene i prosjektet.

Resultatet som ble oppnådd var en plattform som beveget seg fint etter innput fra de forskjellige metodene for styring. 3D simuleringen av plattformen var og tilfredsstillende. Etter denne prosjektoppgaven har medlemmene i gruppen oppnådd en større forståelse for hva det innebærer å sette opp et styresystem, og alle utfordringer som dette byr på. Oppgaven har vist at planlegging av oppgaver, arbeidsmetodikk og en gruppe som jobber sammen er viktig.

Postadresse
Høgskolen i Ålesund
N-6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
Internett
www.hials.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
Foretaksregisteret
NO 971 572 140

INNHOLD

1 SAMMENDRAG

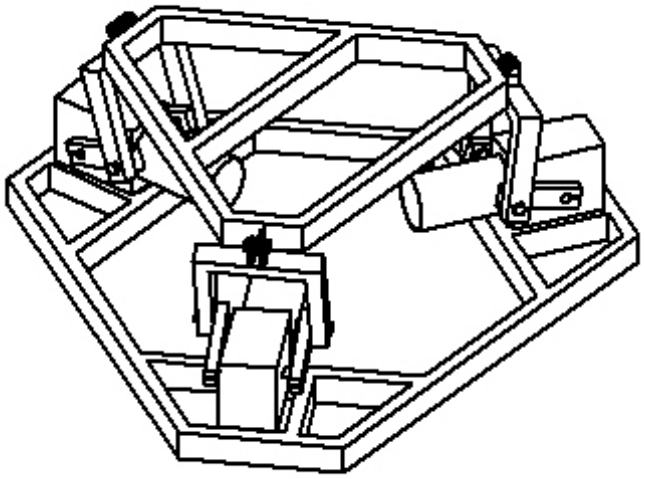
I denne rapporten blir prosjektoppgaven stor plattform presentert. Denne oppgaven var en av mange som gruppene i faget industrielle styresystemer kunne velge mellom. Oppgaven skulle løses med bruk av utstyr tilgjengelig på skolen. Det ble satt av 8 uker til dette prosjektet. Oppgaven skulle bestå av et automatisert system med blant annet et styreprogram, remote I/O, kommunikasjon med tredjepart og visualisering.

Plattformen består av 3 elektromotorer, som styrer hvert sitt hjørne av toppen. Motorene blir styrt av 3 frekvensomformere som igjen blir styrt av en PLS.

Målene gruppen ville oppnå var, styring av plattform ved hjelp av en mobilapplikasjon, et bilspill, et vilkårlig bølgemønster og ved manuell kjøring. Bevegelsene av plattformen skulle også simuleres ved hjelp av en 3D modell i sann tid.

Med disse målene var problemstillingene mange og lærerike. Det første som måtte komme på plass var kommunikasjon mellom PLS og frekvensomformere ved hjelp av profibus. En kommunikasjonsapplikasjon ble nødvendig for håndtering av data mellom de forskjellige systemene i prosjektet.

Resultatet som ble oppnådd var en plattform som beveget seg fint etter innput fra de forskjellige metodene for styring. 3D simuleringen av plattformen var og tilfredsstillende. Etter denne prosjektoppgaven har medlemmene i gruppen oppnådd en større forståelse for hva det innebærer å sette opp et styresystem, og alle utfordringer som dette byr på. Oppgaven har vist at planlegging av oppgaver, arbeidsmetodikk og en gruppe som jobber sammen er viktig.



2 INNLEDNING

Som siste øving i faget industrielle styresystemer skal ett større prosjekt gjennomføres, dette prosjektet skal være i tidsrommet uke 41 Til uke 49. Det ble oppdelt i grupper tidligere i semesteret av faglærer og det har vært en gruppeoppgave før prosjektet skulle starte.

Valget sto mellom to forskjellige plattformer, en «stor» og en «liten». Valget falt på den «store» plattformen som var ferdig laget her på skolen, selve plattformen er trekantet og har en motor i hvert hjørne slik at vinklene skal kunne stilles på plattformen ved hjelp av motorene. Målsetningen med denne er å få den til å kjøre til ønsket vinkler, via forskjellige modus som kan velges i ett program. De forskjellige modusene som ble valgt var:

Manuell styring: Der skal hver enkelt motor til kjøres opp og ned.

Bølgemønster: Plattformen skal kjøres i ett bølgemønster.

Bilspill: Her skal plattformen følge bevegelsene fra en bil i ett dataspill, her blir det festet ett bilsete, ratt og pedaler på plattformen slik vi får følge bevegelsene når vi kjører bilspillet på PC-en.

3 TEORETISK GRUNNLAG

3.1 Plattform

3.1.1 Mekanisk konstruksjon

Plattformen er en kraftig aluminiums konstruksjon, der er en bunnramme som er trekantet med en motor i hvert hjørne. Der hver motor har ett ledd som løfter i hver sitt hjørne av toppdelen av plattformen. Her løfter hver motor sitt hjørne fra bunn til topp ved hjelp av å rotere 125 grader. Det er slik at leddene kan gå over dette, men da vil plattformen begynne å synke igjen fordi leddene står rett opp når motorene har rotert 125 grader. Derfor brukes ikke noe mer rotasjon enn dette når plattformen er i bevegelse.

3.1.2 Elektrisk styring

Styringen er også klar til bruk, denne er bygd opp med 230V tilførsel. Her er det bare PLS som mangler, denne koples på ved hjelp av Profibus. Her er 3 frekvensomformere som styrer hver sin motor og endestoppere er koplet inn på frekvensomformerne, her er også motstander for å få varmen ut av frekvensomformerne når motorene bremses. Det er også montert et kommunikasjons kort på hver frekvensomformer slik at de kan snakke over Profibus. Kommunikasjons kortene kan byttes ut hvis man vil bruke andre kommunikasjons typer.

Posisjonen på motorene blir avlest av encodere som har 1024 steg per runde. Disse er også koplet inn på frekvensomformeren, så man slipper å ta inn input på plsen for å få posisjonen til motorene, denne leser man rett fra Profibus kommunikasjonen.

3.1.3 Komponenter

Frekvensomformere:
3 x Sew-Eurodrive: Movidrive MDX61B + Profibus kommunikasjons kort: DFP21B.

Motorer:
3 x Sew-Eurodrive: KA47 DRS90M4/TF/ES7S 1,5 kW.

Motor Encodere:
3 x OG 73 SN 1024 Sine Encoder.

PLS:
Wago 758-870-000-011

PC:
Når det skal kjøres Input fra Live for Speed, må en pc kjøre spillet og sende ut feedback fra spillet, som igjen blir sendt videre til plsen via Modbus. Det er også satt opp en visualisering i Blender som kjører med samme inputen som sendes til programmet. Og følger samme bevegelser som den ordinære plattformen.

Switch:
D-link DIR 615.

Ratt og pedaler:
Logitech ratt og pedaler, som koples rett i pc-en og brukes når det skal kjøres bilspill på plattformen.

3.2 PLS

PLS står for Programmerbar logisk styring på engelsk Programmable logic controller(PLC). Den har sin opprinnelse i relebaserte kontrollsystemer. Før PLS ble vanlig var det releer, kontaktorer, tidsur og tellere som var vanlig å bruke for automatisk styring. Et slikt styresystem krevde mye kabling og arbeid for elektrikere både under oppbygning og utvidelse. PLS har nå tatt over oppgaven som hundrevis av releer gjorde før.

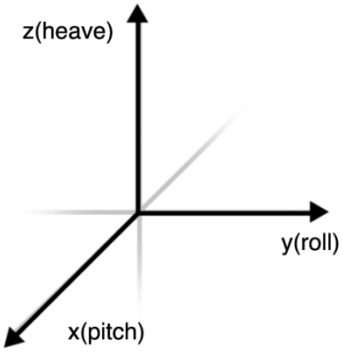
Generelt kan en se på en PLS som en datamaskin, på grunn av virkemåten. Hoveddelen består av en sentral prosesseringsenhet (CPU), minne, strømforsyning, kretsmoduler(I/O-enheter) og kommunikasjonsmoduler. Hovedenhetene er knyttet sammen med ledninger/kopperbaner som kalles busser. En PLS vil typisk ha 4 busser, adressebuss, databuss, kontrollbuss og systembuss. En buss er en samling av ledere.

En PLS må programmeres for at den skal ha noen funksjon, eller skal kunne utføre en oppgave. Dette gjøres med et dataverktøy beregnet på den enkelte type PLS. I dette verktøyet skriver brukeren programkoden, for deretter å laste den opp til PLS-en fra PC-en. Programkoden skrives i ulike språk og med strenge regler for oppbygning og struktur.

[1]

3.3 Kinematikk

For å sette plattformen i bevegelse, benyttes kinematikk.
En bruker to typer kinematikk alt etter hvilken retning en skal beregne, fremover og invers kinematikk.



Fremover kinematikk benyttes for å finne posisjoner ved gitte vinkler på leddene.
Invers kinematikk benyttes når man har en posisjon og en ønsker å vite hvilke vinkler leddene kan settes i for å oppnå dette.

Designet på plattformen i dette prosjektet gir mulighet for bevegelse opp og ned i Z-aksen, og rotasjon i X- og Y-aksen.
Dette gir 3 graders frihet for bevegelse; heave, pitch og roll.

De 3 leddenes utforming, gjør at alle plattformens mulige posisjoner kan oppnås med opptil flere ulike vinkler på hvert ledd.

Ved heave bevegelse i z-aksen, kan en gitt høyde oppnås ved opptil åtte ulike måter, da det er tre ledd, hvor hvert av disse leddene kan nå en gitt posisjon på to ulike måter, $2^3=8$.

[2],

3.4 Programvare

3.4.1 CoDeSys

CoDeSys er et programmeringsverktøy utviklet av Smart Software Solutions GmbH. Programmet følger IEC 61131-3 programspråk standarden, og er utstyrsuavhengig programmeringssystem rettet mot PLS-er, kontrollere og annet hardware. Programmet inneholder simuleringsverktøy og et grafisk visualiseringsverktøy som muliggjør testing av software uten oppkobling mot hardware.

Programmet kan lastes ned gratis fra leverandørens nettside mot registrering.
[1], [3]

3.4.2 Live For Speed

Live for Speed (LFS) er en racing simulator hvor fokuset er at kjøreopplevelsen skal være så realistisk som mulig, det er ingen «hjelpemidler», noe en finner i mer arkadeorienterte bilspill. Det er også tilrettelagt for uthenting av den informasjonen en trenger for å simulere bevegelsene over til en bevegelsessimulator.

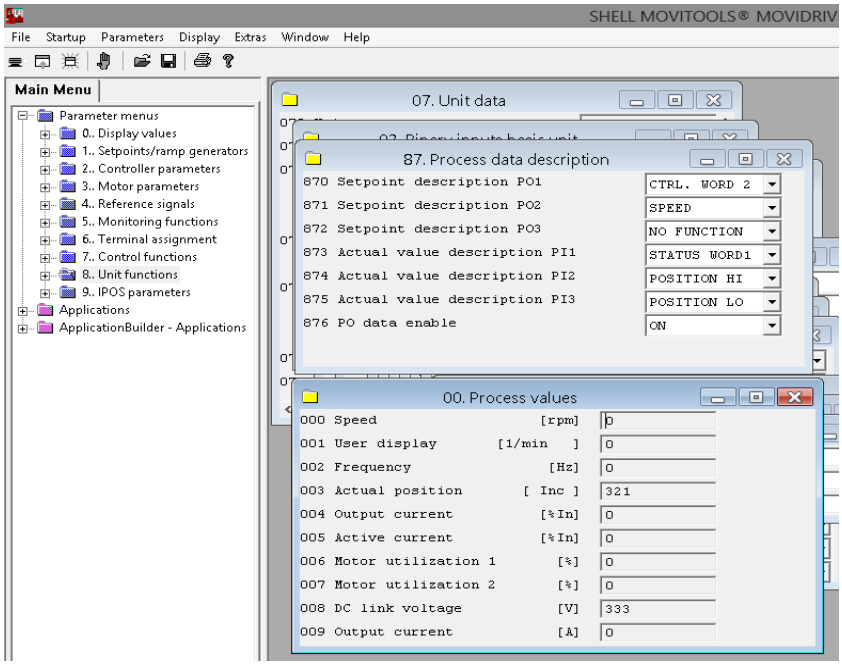
3.4.3 IMU

En Inertial Measurement Unit (IMU) er et apparat som ved hjelp av gyroskop og akselerometer kan beregne apparatets hastighet og orientering i rommet. Siden dagens mobiler har alt dette, kan dette brukes for å styre plattforme. Mobil appen «Sensorstream IMU+GPS» av Axel Lorentz som tar i bruk mobilens IMU egenskaper og sender ut mobilens orientering og hastighet ved hjelp av UDP pakker, ble valgt til å styre plattformen med.

[4]

3.4.4 MoviTools

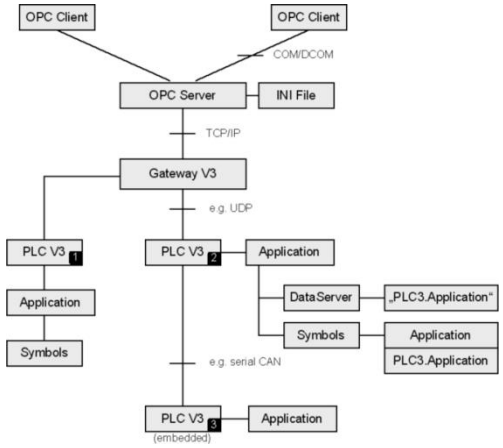
Programvaren brukes for å programmere frekvensomformerne, her kan du endre alt av parameter til frekvensomformerne. Dette ble brukt til å stille inn hvilke prosess ord som skulle brukes og start/stopp rampe på motorene. Dette kan også overvåke bus-kommunikasjonen, slik at en ser hva som blir aktivert når det blir sendt dataord fra PLSen. Dette er gratis Software som kan lastes ned fra Sew-eurodrive.com. Utklipp av Movitools programvaren, der vi kan velge kategorier, underkategorier, justere og følge med på verdier i «realtime» via seriellkommunikasjon i vinduene som kommer opp, til høyre.



[5]

3.4.5 OPC

OPC, kort for OLE for Process Control(OLE - Object linking and embeding), er en industristandard innen automasjon som brukes for åpen kommunikasjon mellom forskjellige typer utstyr, uavhengige av leverandør.
OPC kan beskrives som en tolk som gjør at ulike systemer og sensorer kommuniserer med hver andre på tvers av plattformer.[6]



CODESYS inneholder et eget OPC-Server grensesnitt, som muliggjør utveksling av variabler, både lesing og skriving.[7]

En typisk operasjon der en benytter OPC kan ved være en visualisering, der en OPC server tar seg av oppdatering av variabler levert fra en kontroller, gjerne en PLC, og en klient som henter ut data og presenterer disse.

3.4.6 LabVIEW

LabVIEW, utviklet av National Instruments, er et verktøy som benyttes til å utvikle applikasjoner for visualsisering, måling og styring.
Verktøykassen i LabVIEW har store bibliotek med instrumenter, og har støtte for et mangfold av hardware, programmeringsspråk og protokoller, blandt annet OPC, Modbus, UDP.

[8]

3.5 Programmeringsspråk

3.5.1 IEC 61131-3

IEC 61131-3 kom ut i 1993 og er en standard for programspråk.
Denne standarden følges i mer eller mindre grad av de fleste større PLS produsenter.
Standarden omfatter 5 ulike programmeringsspråk, *Structured Text (ST), Function Block Diagram (FBD), Ladder Diagram (LD), Instruction List (IL)* og *Sequential Function Chart (SFC).*

Strukturert Tekst
Structured Text(ST) er et tekstbasert høynivåspråk med likhetstrekk til programmeringsspråkene Pascal og C.
Et program i Strukturert Tekst består av valgsetninger, løkker, og instruksjoner.

PUOs
Programmstrukturen er bygget opp av 3 typer Programmorganiseringsenheter (POU); Funksjoner(FUN), funksjonsblokker(FB) og program(PRG).

Funksjoner
En *funksjon (FUN)* er definert som en PUO som gir ut kun ett dataelemet når den instansieres.
En funksjon som instansieres flere ganger med samme argument vil gi samme resultat hver gang.
Et eksempel på en funksjon kan være en matematisk funksjon som sinus.

Funksjonsblokker
En *funksjonsblokk (FB)* er definert som en POU som i forskjell fra funksjoner, har intern hukommelse.
I tillegg kan en funksjonsblokk ha opptil flere utgangsvariabler.
Et eksempel på en funksjonsblokk kan være en teller.
En funksjonsblokk kan instansiere funksjoner.

Program
Et *program (PRG)* er definert som en POU som er satt sammen av ulike elementer i programmeringsspråket på en slik måte at det utgjør en kjørbar kode som kan løse en styringsoppgave.
Programmer er overordnet funksjonsblokker og kan instansiere både funksjoner, funksjonsblokker og andre programmer.

[1]

3.5.2 Java

Java er et høynivå, plattform uavhengig, objektorientert programmeringsspråk. Språket ble skrevet av James Gosling og Sun Microsystems, Java tilhører nå Oracle Corporation. Filosofien til Java er «write once, run anywhere», det vil si at koden skal kunne skrives på hvilken som helst operativsystem og kjøres på hvilke som helst operativsystem, uten at koden må endres. Det er mulig fordi Java koden blir kompilert til en Java bytekode, istedenfor direkte til en plattform-spesifikk maskinkode, og kjørt på en plattform-tilpasset Java Virtual Machine (JVM) som videre kompilerer en plattform-spesifikk maskinkode.

[9]

3.5.3 Python

Python er et mye brukt objektorientert programmeringsspråk. Programmet var i utgangspunktet tenkt som et scriptspråk for Amoeba OS for å lage systemendringer. Perl, Ruby og til dels Java blir sett på som alternativer til dette programmeringsspråket. Python er utviklet som en fri programvare. Fri programvare er sterkt relatert til åpen kildekode.
[10]

3.6 Feltbus

3.6.1 Profibus

Profibus er en av de ledende feltbusene, den kommuniserer via seriell kommunikasjon (RS485) og har en overføringshastighet på 9,6 – 12000Kbit/s, men dette begrenses igjen av avstanden mellom nodene. Som man ser i tabellen til venstre har man en rask nedgang i overføringshastighet når distansen blir lang.

Det benyttes tvinnet kabel og det brukes ofte db9 kontakter, men dette er ingen standard. Det finnes mange typer forskjellige kontakter på markedet, alt etter hvilke leverandører man bruker.

Det kan også brukes fiberkabel, og med dette økes kommunikasjons-avstanden opp mot 15Km, men da med fare for redusert redundans.

[11]

Transmission rate [Kbit/s]	Transmission range per segment [m]	Applies to
9,6 19,2 45,45 93,75	1200	RS485
187,5	1000	RS485
500	400	RS485
1500	200	RS485
3000 6000 12000	100	RS485
31,25	1900	MBP
The values above apply to cable type A with the following properties		
Wave resistance	135 ... 165 Ω	
Capacitance per unit	≤ 30 pf/m	
Loop resistance	≤ 110 Ω/km	
Core diameter	> 0.64 mm	
Core cross-section	> 0,34 mm²	

Table 2: Transmission values of RS485 and MBP

Big /Little endian Byte order
«Little endian» betyr at bytet med laveste orden blir lest av på første adresse, og bytet med høyeste orden blir lest av på høyeste adresse.

«Big endian» betyr at bytet med laveste orden blir lest av på høyeste adresse, og bytet med laveste orden blir lest av på laveste adresse.

Adresse i F.eks. Codesys	«Big endian»	«Little endian»
%QB500	Byte 3	Byte 0
%QB501	Byte 2	Byte 1
%QB502	Byte 1	Byte 2
%QB503	Byte 0	Byte 3

[12]

3.6.2 Modbus

Modbus er en mye brukt kommunikasjons protokoll i industrien for å kommunisere til og mellom PLSer og annet utstyr. Den originale protokollen, som ble opprettet i 1979 av Modicon, kommuniserer serielt og har et master-slave oppsett. Modbus TCP/IP protokollen, som blir brukt i dette prosjektet, ble opprettet i 1999 og tar i bruk TCP/IP protokollen, det fører til at en lett kan sette opp kommunikasjon over Ethernet.

[13]

3.6.3 Ethernet

Ethernet brukes til kommunikasjon mellom PC og PLS og kommunikasjonen til mobiltelefonen går over WLAN / trådløst nettverk. Dette er satt opp med en egen switch som har trådløs muligheter.

[14]

3.7 Transportprotokoller

3.7.1 UDP

UDP er en lettvekts meldingsprotokoll som sender ut pakker til en port og IP adresse uten å bry seg om det er noen på den andre enden som tar imot pakkene, det fører til at protokollen er rask og relativ lett å lage servere og klienter til.

[14]

3.7.2 TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) er en pakke med protokoller som blir brukt til å transportere data på et nettverk, og i motsetning til UDP protokollen så sørger TCP/IP for at der er noen på den andre ender før den starter å sende pakker, den passer og på at pakkene kommer fram i rett rekkefølge.

[14]

3.8 3D Simulering/ Visualisering

3.8.1 SolidWorks

SolidWorks er et komplett 3D – tegneprogram som gir deg muligheten til å lage, simulere, publisere og behandle 3D modeller. SolidWorks er utviklet av Dassault Systemes SolidWorks Corp. Programmet er et såkalt CAD - Computer Aided Design verktøy, eller dataassistert konstruksjon som er den norske betegnelsen, ofte forkortet DAK. Programmet brukes for konstruksjon og teknisk tegning. Det er ofte brukt av ingeniører, arkitekter og andre designere innen ulike industrier som bygg og anlegg.

SolidWorks simuleringspakke tilbyr løsninger hvor en kan skape virtuelle virkelighetsmiljøer, der en kan teste produkter. En kan teste de ulike produktene opp mot en rekke parametere som holdbarhet, statisk- og dynamisk respons, sammenstillingens bevegelse, varmeoverføring og væskedynamikk. Dette for å vurdere designytelsen og forbedre sammenstillingen.

[15]

3.8.2 Blender

Blender er et open source 3d-modellering-, animasjon- og renderings-program. Dette programmet er tilgjengelig på alle de vanligste operativsystemene. Med dette programmet kan en lage 3D modeller, spill og animasjoner. Blender er gratis og har mange gode nettsider og forum som formidler opplæringsvideoer. Blender har også en innbygget spillmotor (bge- blender game engine) som kan brukes for simulering. Verktøyet støtter også bruk av python scripting for å lage spill og komplekse interaktive animasjoner.

Programvaren til Blender blir utviklet av et samfunn som består av frivillige programmerere, studioer, fagfolk og amatører.

[16]

4 METODE

4.1 PLS Programmering

Programmet for å kjøre WAGO PLSen er skrevet i CODESYS med strukturert tekst, programmet består av flere program, funksjoner og funksjonsblokker for å redusere gjentatt koding.

Hovedprogram

PLC_PRG (PRG)
Hovedklassen i programmet som der en velger hvilken modus som skal kjøres.

READ_POS(PRG)
Dette programmet leser av verdiene fra encoderene for å finne den aktuelle posisjonen til motorene.

SET_STEP(PRG)
Dette programmet kjører opp, ned eller bremser hver enkelt av motorene.
Ut ifra ønsket posisjon beregnes motorhastigheten for å unngå skarpe bevegelse.

Moduser

INPUT_MODE (PRG)
Denne modusen tar imot roll og pitch verdier med modbus, og kjører deretter plattformen til ønsket posisjon.

For å oppnå myke bevegelser og unngå at falske inn-verdier skaper uønskede bevegelser, har dette programmet innebygget et IIR filter. [19]
Programmet har også innebygget en simuleringsfunksjon der en kan sette roll og pitch manuelt for testing.

MANUAL (PRG)
Med denne modusen kjøres hver enkelt motor opp og ned manuelt.

WAVE_V3 (PRG)
Denne modusen kjører plattformen i et bølgemønster.
Hver enkelt motor kjøres opp og ned med sinusfunksjoner som er forskjøvet i forhold til hverandre.
Hastigheten kan justeres på hver enkelt motor som for å skape mer variasjon.

Kalkuleringer

CALCULATED_ACTUALS (FB)
Tar inn avleste steg i fra hver av motorene og regner om til nåværende vinkel og høyde på hver av motorene, og pitch og roll på plattformen.
Bruker funksjonsblokken STEP_TO_HIGHT for høyde og vinkel.

HIGHT_TO_STEP (FB)
Inn: length
Ut: angle, step

Tar inn høyde og regner ut vinkel og steg som kan settes på hver motor for å oppnå denne høyden.

ROLL_PITCH_TO_STEP(FB)
Inn: roll_angle, pitch_angle
Ut: step1, step2, step3

Tar inn vinkler for roll og pitch, kalkulerer så høyden på hver enkelt motor for å oppnå gitte vinkel.
Bruker HIGHT_TO_STEP funksjonsblokken for å regne ut setpunkt til hver av de tre motorene.

STEP_TO_HIGHT(FB)
Inn: step
Ut: h(høyde), a(vinkel)
Tar inn step verdi lest av fra encoder og regner ut høyde og vinkel på hver av motorene.

Funksjoner

M1,M2,M3(FUN)
Inn: hastighet/retning
Kjører motoren i ønsket hastighet og retning.

GUI

HOME
Velkomst-meny med valg av modus.
Manual_mode
Grensesnitt for manuell kjøring av plattform.
UDP
Grensesnitt for kjøring av plattform i input modus.
Wave_mode
Grensesnitt for kjøring av plattform i bølgemønster.

GRAF
En grafisk representasjon av motorens bevegelse.

4.2 Kommunikasjon mellom PC og PLS

Kommunikasjonen mellom pc og PLS går over Ethernet, der brukes det en egen switch med mulighet for å koble seg til med trådløs tilkobling. Alle enhetene på dette nettverket er satt opp på adresseområdet 158.38.140.xxx, PLSen er satt opp med 158.38.140.113. Alle enhetene som er koblet på dette nettverket har fått egne reserverte IP addresser for å unngå kommunikasjonsfeil.

[14]

4.2.1 Modbus

Når PLSen tar inn signalene fra UDP stream og fra bilspillet, blir dette distribuert over Modbus. Dette blir sendt rett til en standard adresse i PLS-en, slik at det ikke er nødvendig å lage til noe ekstern kode i programmet for å motta Modbus verdiene.

Dette er på PLS-en adresse «%MW0», «%MW1» osv. Dette tilsvarer adressene 12288 til og med 247575 sett fra java applikasjonens ståsted.

Deretter blir adressene linket til variabler og brukt som input i programmet.
Java server programmet skriver pitch og roll til PLSens adresser MW0 og MW1.
Applikasjonen leser også plattformens vinkler på adressene MW2 og MW3 for å sende disse videre til Blender som presenterer dette visuelt. [17]

4.3 Kommunikasjon Mellom PLS og Frekvensomformere

Kommunikasjonen mellom PLS-en og frekvensomformerne går over Profibus DP med en baudrate på 12mbit/s, der PLS-en er Master og Frekvensomformerne blir Slaver. Kommunikasjonen foregår med å utveksle Data ord, der et ord er 16Bit eller 2 Bytes. Første steg for å få dette opp og gå er å legge inn en GSD fil i programmet til plsen, den gjør slik at det kan velges hvor mange inn og ut ord som skal brukes.

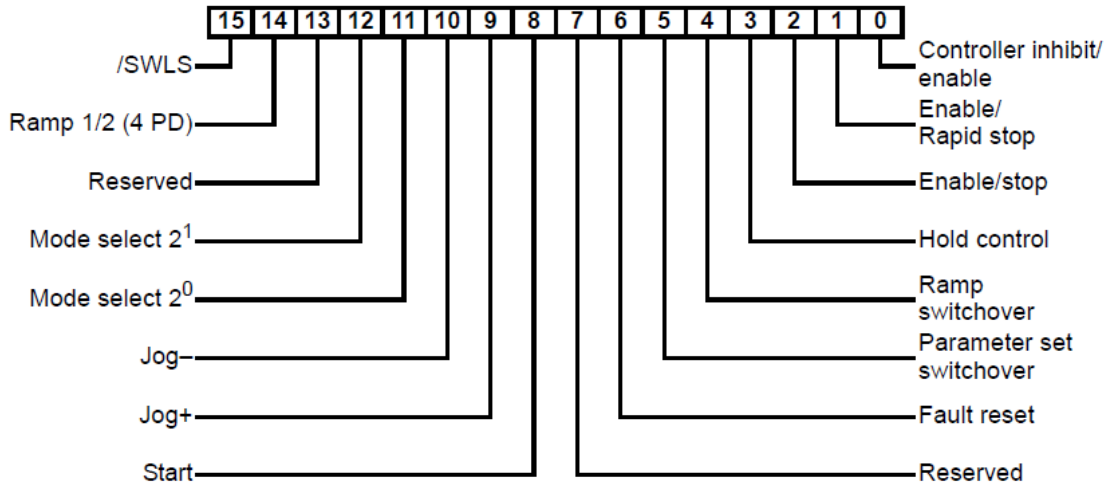
Så må adressene til omformerne settes, her er adressene 2,3,4, disse stilles inn ved hjelp av dipswitcher på frekvensomformerne. Adressene til ordene blir laget automatisk etter at det har blitt lagt inn hvilke inn- og utganger som skal brukes (F.eks. %QW2400). I denne oppgaven er det brukt 7 Inn og 7 ut ord, der de første 4 ordene er en parameter kanal der du kan lese/skrive ut hver enkelt parameter. De 3 siste inn og ut ordene er prosess ord. Ordene bruker «Big Endian» rekkefølge.

4.3.1 PROSESS ORD:

Prosess Output Word:
Prosess ordene blir brukt til å starte og stoppe motoren. Der PO1 er ett kontrollord som aktiverer frekvensomformeren, og gjør slik at den er klar til bruk. PO2 er brukt til å bestemme hastigheten og retningen på motoren.

Når motorene skal kjøres må det sendes inn 2 prosess ord, PO1: som aktiverer frekvensomformeren, og PO2: som setter hastigheten.

PO1: Control word 2

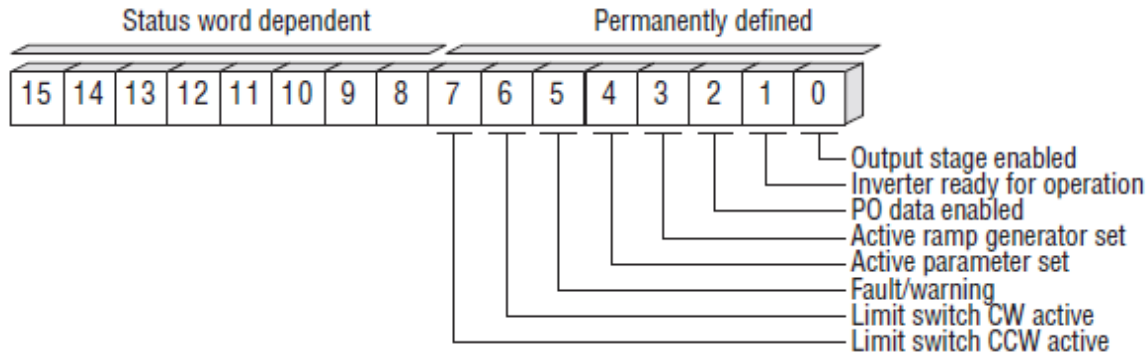


[Sew Eurodrove Manual – Movidrive MD_60A: DriveInverters: Appendum to system manual: BUS positioning - edition 02/2000]

For å aktivere frekvensomformerne må Bit 1 og Bit 2 settes «TRUE» og resten kan man egentlig se bort ifra. For at dette skal skje må bit9 og bit 10 settes fra plsen på grunn av byte rekkefølgen som brukes.

«PO2: Setpoint Speed»: Skal settes til ett tall mellom 0 – 4000, men dette er begrenset til 2000, på grunn av konstruksjonen på plattformen.

Prosess Input Word:
«Prosess input Word» er tilbakemeldingen fra frekvensomformerne. Det mottas 3 ord fra frekvensomformeren, der ord 1 er «Status Word» som gir tilbakemelding på hvilken tilstand frekvensomformerne er i, status på endebrytere, feil, osv. Det kan også programmeres på frekvensomformerne hvilke tilbake meldinger du skal ha.
«Status Word»:



De to neste ordene er brukt til å sende posisjonen til motorene. Posisjonen er sendt over to data ord, det vil si at den sender en «pos index hi» og en «pos index low», det vil si at de to ordene må plusses sammen før vi kan lese av nøyaktig posisjon. Her må «Byte order» også tas hensyn til.

[18]

4.3.2 Posisjons avlesning

Posisjons input fra frekvensomformerne kommer på prosess ord 2 og 3, som har adresse «%IW2405» og «%IW2406» for motor 1.

```
(*Motor 1*)
    In_data_1 AT %IW2405: WORD;
    In_data_2 AT %IW2406: WORD;
    Data_in_m1_temp: WORD;
```

Variablene til avlesning av posisjon til motor 1.

Deretter blir en ny variable satt til summen av begge inndataene, som også blir skjøvet 8 plasser mot venstre, dette gjøres for å bytte plass på Bytene slik de kommer i riktig rekkefølge.

```
(*Leser av posisjonsdata til motor 1*)
Data_in_m1_temp:= ROL(In_data_1 + In_data_2, 8);
DATA_IN_M1 := WORD_TO_INT(Data_in_m1_temp);
```

programmet som gjør om fra 2 data ord til en Int verdi med posisjonen til motor 1.

Deretter blir data_in_m1_temp gjort om fra WORD til INT, og satt til DATA_IN_M1 som er en global variabel. Denne er nå av typen INT og gir oss posisjonen til plattformen.

Detter er gjort på samme måte for alle motorene i ett og samme program som kjører hele tiden, slik at posisjonen til plattformen alltid er tilgjengelig.

Nullstilling av posisjon

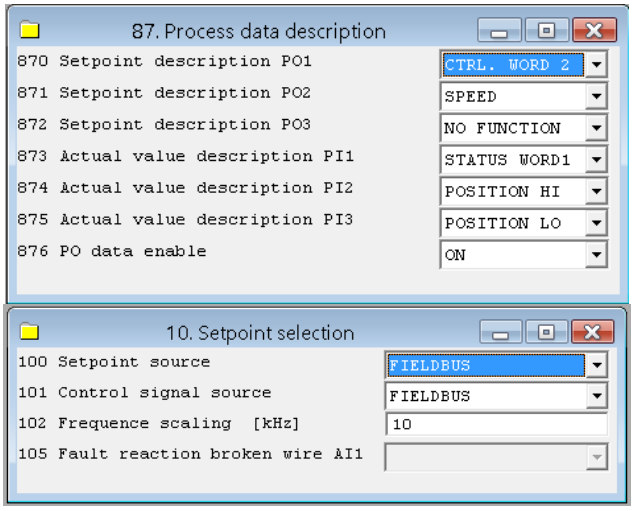
Når frekvensomformerne starter opp blir nullpunktene til encoderene satt til den posisjonen de befinner seg i, dette kan gi utslag på plattformens bevegelse da hvert ledd ikke står i lik posisjon.
For å utbedre dette brukes noen klosser til å støtte opp mellom rammen og leddene før strømmen slås på.

4.4 Oppsett frekvensomformere

Frekvensomformerne er satt opp via Movitools programvaren, der er noen kritiske punkter som må settes opp før frekvensomformerne kan kjøres. Det ble satt opp til å bruke prosess ordene: «PO1: Controll word 2», «PO2: Speed» og PO3: No Function. «PI1: Status word 1», «PI2: actual position high» og «PI3: Actual position low».

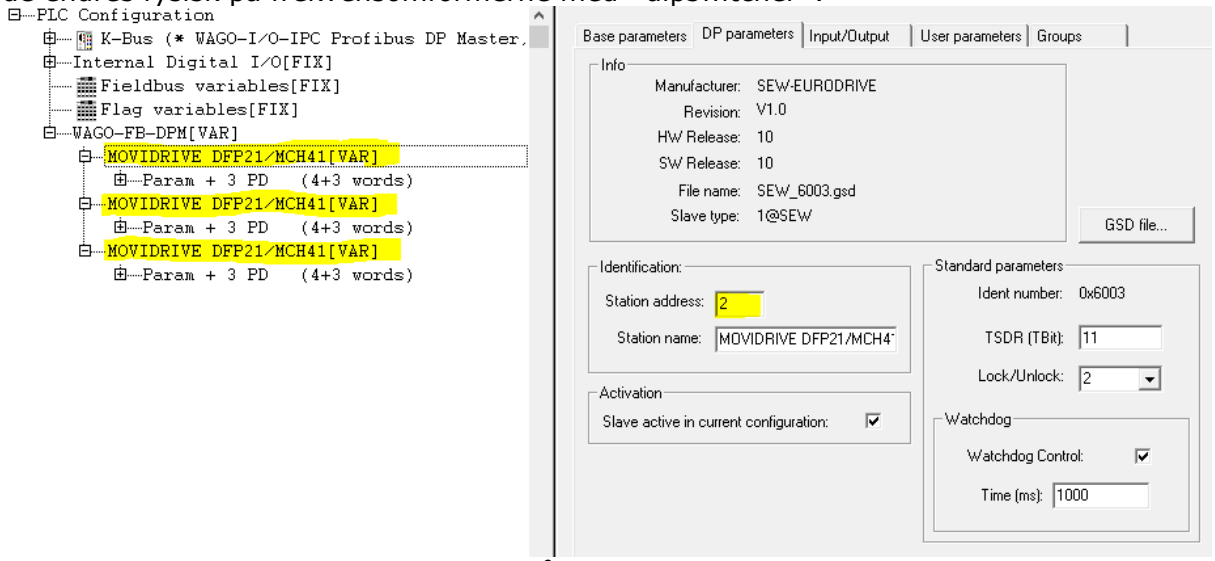
Innstillinger for «Setpoint selection» må også velges:
Her skal «Setpoint source» stå til «fieldbus» og «Control signal source» skal stå til «fieldbus».
Dette er innstillingene til hvilken kilde frekvensomformeren skal hente informasjon fra.

Dette er de mest kritiske innstillingene for å kunne styre motorene. Der er mange andre innstillinger som kan endres men dette er ikke en nødvendighet for å kjøre programmet.



4.5 Oppsett PLS

Oppsettet til PLS-en slik at den kommuniserer med frekvensomformerne må settes opp i «PLC-Config» i Codesys programvaren. Her må det legges inn en GSD fil, legge til en Profibus master, så legge inn frekvensomformerne som slaver under profibus masteren. Profibus masteren har adresse 1, og frekvensomformerne har adresse 2, 3 og 4. Adressene til frekvensomformerne kan ikke endres her, skal andre adresser brukes må de endres fysisk på frekvensomformerne med «dipswitcher».



Skjermbilde av konfigurasjon av oppsettet på PLS-en.

Baudraten må også settes, dette gjøres ved å velge den Wago-FB-DPM[VAR] → «Bus parameters» på oppgave linjen → Da kan denne endres. Den skal være satt til 12Mbit/s. (12000Kbit/s).

4.6 Kinematikk

Plattformens bevegelser baserer seg hovedsakelig på input med pitch og roll. Gitt en ønsket posisjon skal hvert enkelt av hjørnene heves eller senkes for å imøtekomme posisjonen.

Beregninger for vinkel for hvert ledd:

$$h = l1\cos(a) + \sqrt{l2^2 - l1^2\sin^2(a)}$$

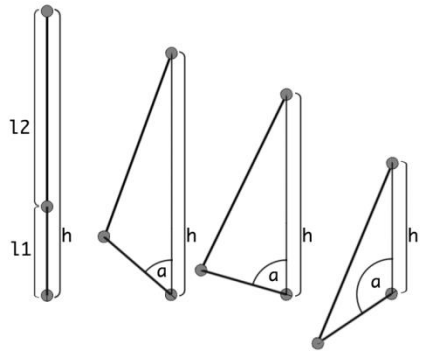
$h_{max} = l1 + l2 = 480mm$

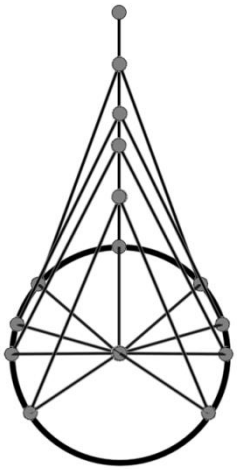
$h_{min} = 220mm$

$l1 = 150mm$

$l2 = 330mm$

$$a = \cos^{-1}\left(\frac{h + l1 - l2}{2 * h * l1}\right)$$





Beregninger for rotasjon i X og Y akse:

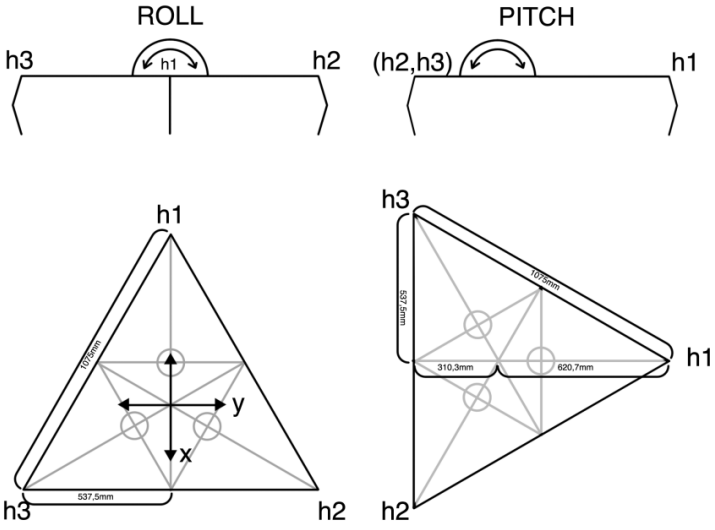
Ut ifra pitch og roll beregnes høydeforskjellen mellom hver av 3 leddene.

Rotasjon i X-aksen(pitch) beregnes med h1 i front, og h2 og h3 i bakkant.

Rotasjonssentrum for X er forskjøvet 620,7 mm bakover i fra ledd h1.

Høydeforskjellen blir da fordelt mellom hjørne h1 og hele plattformens bakdel, mellom hjørne h2 og h3.

Ved rotasjon i Y-retning(roll) beregnes en høydeforskjell i hjørne h2 og h3.



roll_Angle = $\sin^{-1}(\Delta h2\text{-}h3 / 1075)$
$\Delta h2\text{-}h3 = \sin(roll\_Angle) \times 1075$

pitch_Angle = $\sin^{-1}(\Delta h23\text{-}h1 / 931)$
$h1 = \sin(pitch\_Angle) \times 620{,}7$
$h23 = -\sin(pitch\_Angle) \times 310{,}3$

[20]

4.7 Live for Speed

LFS har en innebygd OutSim funksjon som, vist aktivert, sender ut den informasjonen en trenger for å simulere bevegelsene fra spillet til en bevegelsessimulator, informasjonen blir sendt via UDP protokollen. For å aktivere OutSim funksjonen må konfigurasjonsfilen(cfg.txt) endres for å bestemme hvilken IP-adresse og port nummer pakkene skal sendes til, samt hyppighet.

4.8 3D Simulering

4.8.1 SolidWorks

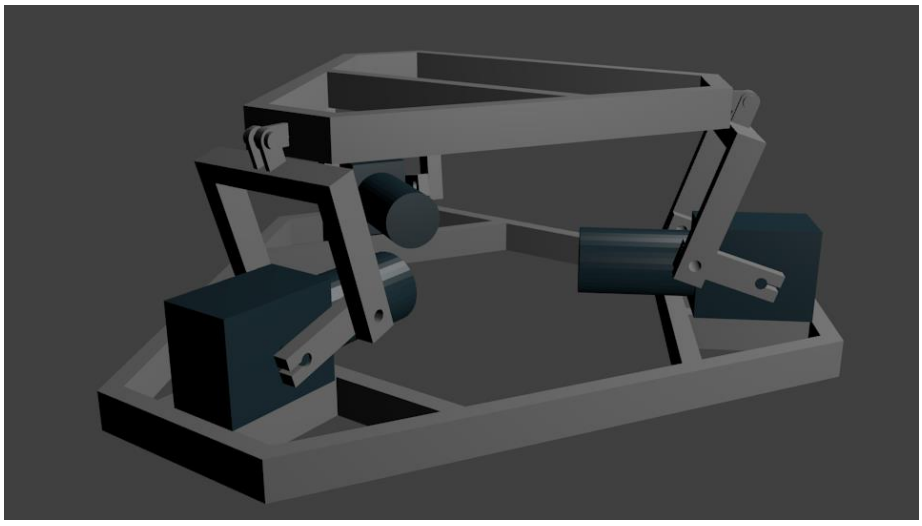
En del av prosjektet var å lage en visualisering. Det ble bestemt at en 3D modell av plattformen skulle lages i SolidWorks. Det var på dette tidspunktet kjent for gruppen at SolidWorks ikke støttet sanntid-simulering. Verktøyet ble likevel benyttet for tegning og sammensetting av modellen. Dermed var en nøyaktig og rett sammensetting av modellen tilgjengelig for videre arbeid. For at denne modellen skulle være mest mulig lik originalen og den skulle oppføre seg på samme måte, var det nødvendig med nøyaktige mål og vinkler. Når arbeidstegningen var på plass var det klart for 3D – tegning. Siden programmet er et populært og mye brukt verktøy fantes det mange gode videoer og tutorials på nettet som en kunne lære fra.

Måten en bygger opp en sammensetting i Solidworks på er delt i to metoder. Det første en må gjøre er å skille alle deler og tegne de hver for seg. Dette ble for gruppen sin del slik att plattformens topp, bunn, motorer, bolter og ledd ble tegnet hver for seg. Når alle de ulike delene var på plass var det klart for sammensetting. Sammensettingen av deler gjøres slik at en lager et geometrisk forhold mellom komponentene. Det kan for eksempel være at de skal være parallelle konsentriske, eller at flater skal ha en maksimum/minimum avstand fra hverandre.

Siden Solidworks ikke støtter den form for simulering som gruppen var ute etter, ble det bestemt at modellen som ble tegnet skulle importeres til et annet program som støttet simulering av 3D modeller i sanntid. Simuleringsverktøyet trengte også en mulighet for henting og behandling av data. Etter en del undersøkelse ble det bestemt å bruke et program som heter Blender.

4.8.2 Blender

Etter at Solidworks modellen ble importert til blender så ble det skrevet et Python skript som har en UDP server for å få informasjon, Pythons egne matrisefunksjoner for rotasjon i rommet, og blender game engine for å få bevegelse på plattformen. For enkelhetens skyld så ble det bestemt for å lage koden slik at blender får inn hvilken «roll» og «pitch» vinkel plattformen skulle ha, og setter deretter plattformen direkte til de vinklene, for så å la bena følge naturlig etter, det vil si at bena følger etter plattforma og ikke omvendt som det er i den virkelige plattformen. For å få rotasjon på plattformen ble det brukt en rotasjonsmatrise funksjon fra Pythons matematiske bibliotek, den tar inn enn vinkel (roll eller pitch), hvilken akse det gjelder og størrelsen på matrisen den skal lage.



Siden Solidworks er konfigurert slik at y-aksen er opp og i blender er z-aksen opp så førte det til en del unødvendige problem. Det var ingen enkel løsning på å snu aksene, så det førte til at en måtte roterte plattformen i blender, noe som oppførte seg fint helt til blender game funksjonen ble startet, noe en må for å få bevegelse i animasjonen, for då snudde plattformen seg tilbake til sin originale importerte posisjon, løsningen vart tilslutt å gange den opprinnelige rotasjonsmatrisen med en ny rotasjonsmatrise som roterte plattformen tilbake slik at z-aksen er opp.

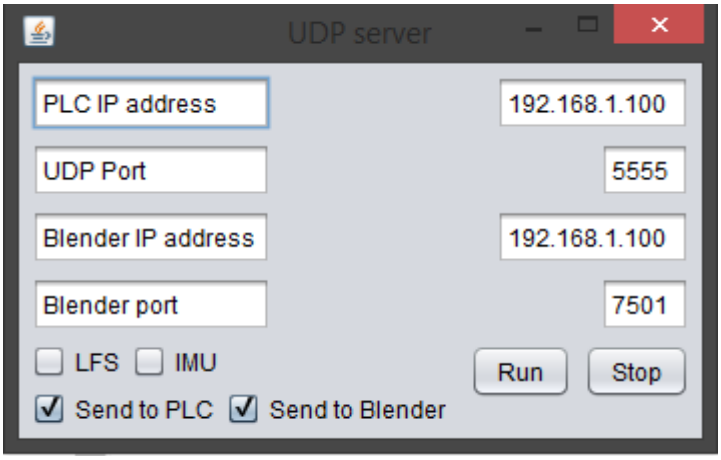
For å få bena til å følge etter så ble det brukt noe som heter «rigid body joint», det er en funksjon som lar to ledd ha et hengsleledd mellom seg rundt en bestemt akse, eller de kan henge samen ved hjelp av et kuleledd som kan bevege seg rundt alle aksene.

Blender får dataene som blir brukt i scriptet fra UDP servern. Serveren henter verdiene ”roll” og ”pitch” fra PLS programmet (CoDeSys). CoDeSys skriver verdiene som plattformen blir satt til, til modbus-adresser i CoDeSys. Disse verdiene blir så lest av Java - applikasjonen over modbus. Deretter sender applikasjonen verdiene videre til Python scriptet i blender via UDP for simulering i sanntid.

4.9 Java Server

For å få relevant informasjon fra spillet eller fra mobilen til PLSen så ble det laget en Java server med tilhørende GUI (grafisk grensesnitt). Den tar imot UDP pakker fra spillet eller mobilen og håndterer informasjonen i pakkene og sender relevant informasjon videre til PLSen og Blender.

Det ble laget tre klasser, en «GUI», en «UDP server» og en «Print consol» . Klassene er relativt sjølforklarende, «GUI» klassen tar seg av det grafiske, «UDP serveren» setter opp en «UDP server» og behandler dataen den får inn og sender deretter dataen videre, og «Print console» klassen oppretter et konsoll vindu som viser til brukeren at dataen blir behandlet og sendt videre.



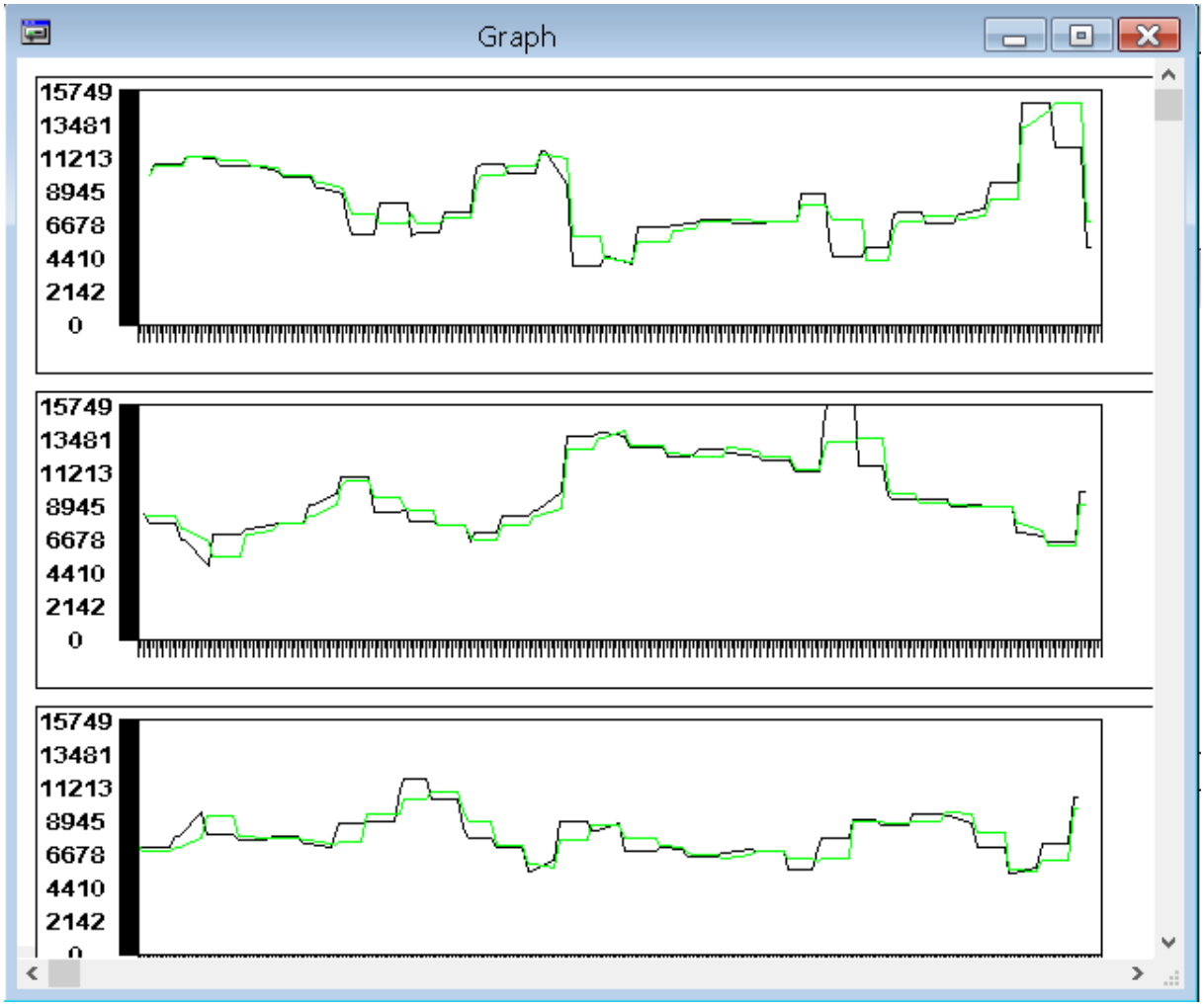
UDP Server GUI

For å sende informasjonen fra Java til PLSen via Modbus, ble Java biblioteket jamod importert og brukt. I jamod er det ferdige funksjoner for å sette opp kommunikasjonen og skriver til en bestemt plass i minnet til PLSen for å fortelle hvilke vinkler den skal gå til. Java serveren leser og fra PLSen, via modbus, hvilke vinkler plattformen har på det tidspunktet, og sender de verdiene til blender, slik at animasjonen i blender følger plattformen, ikke mobilen eller spillet. Modus har port 502 som standard, det er denne porten Java skriver til og leser fra.

5 RESULTATER

5.1 Live For Speed

«Live for Speed» ble kjørt med tilfredsstillende resultater, uten noe betydelig forsinkelse til plattformen. Følelsen av å følge bilen er tilstede og utslaget på plattformen er reell i forhold til bevegelsene som faktisk skjer i spillet.



Bildet viser grafer av når Live For Speed ble kjørt, den svarte grafen er den verdien plattformen skal ha og den grønne er verdien plattformen har. Fra toppen og ned: Motor 1, Motor 2, Motor 3.

5.2 IMU – Mobil Bevegelse

Plattformen kan også styres via en IMU applikasjon som sende over akselerometer data til plattformen, her ble resultatet også ganske bra. Det som ikke var helt ideelt var at til å begynne med var bevegelsene veldig brå og hakkete, derfor måtte det inn ett IRR filter slik at bevegelsene ble myke og fine. Dette gikk litt utover responstiden på plattformen, slik at den ble hengende noen tiendedeler etter.

5.3 Bølge Mønster

Bølge mønsteret ble også bra etter hvert, det som ble best resultat var når vi kjørte hver enkelt motor med en sinuskurve hver. Dette førte til myke og rolige bevegelser på plattformen, her kan hastigheten på hver motor stilles. Slik det ble erfart av gruppen ble dette ganske realistisk i forhold til det man kan oppleve ved å være på sjøen i urolig sjø.

5.4 Manuell kjøring

Det ble også laget en funksjon for å kjøre plattformen manuelt. Der ble det laget en visualisering i Codesys som hele plattformen kan styres ved enten å kjøre hver enkelt motor, alle motorene samtidig eller at en motor blir satt til en posisjon.

5.5 3D modell i Blender

En 3D modell ble tegnet og satt opp med bevegelse i Blender, Blender tar imot data fra plattformen via modbus og den beveger seg i samme mønster som plattformen. Her er største problemet at plattformen i blender har litt mindre utslag på bevegelsene enn den faktiske plattformen, ellers er bevegelsene lik.

5.6 Kommunikasjon

Målet for kommunikasjon i oppgaven er oppnådd, ved at vi har Profibus som hoved kommunikasjon mellom PLS-en og frekvensomformerne.
Mellom PLS og PC når vi sender data fra IMU og bilspill blir det brukt Modbus og fra Java til Blender over UDP.

6 DISKUSJON

6.1 Java Server

Vi valgte å la all kommunikasjon mellom LFS, IMU, Blender og PLSen styres av en Java applikasjon, på grunn av dette så ble det ikke nødvendig å sette opp en OPC server eller lage en LabVIEW applikasjon.

Kommunikasjonen mellom PLSen og Blender kunne ha gått direkte fra PLSen til et Python skript, men vi valgte derimot å gå en liten omvei med å la Java applikasjonen styre også denne delen, slik at vi får en oversiktlig applikasjon som tar seg av all kommunikasjon.

6.2 Bilsimulator

Når spillet som skulle brukes til bilsimulatoren skulle velges ble Live For Speed valgt, grunnen til at dette spillet ble valgt er i hovedgrunnen at det er gratis, dette for å holde kostnadene nede på prosjektet. Det finnes mange andre spill som er mulig og bruke, f.eks. Racedriver GRID, Dirt 2, disse ble vurdert og ikke brukt. Programmet som tar imot inputen fra Live For Speed er konstruert slik at vi tar inn en X og Y verdi, det er derfor ikke noen stor sak og implementere andre bilspill til å fungere. Når plattformen tar inn bare en X og Y verdi kan den også brukes til f.eks. flysimulator, eller en annen form for simulering.

6.3 Kommunikasjon

Kommunikasjonen som ble valgt mellom PLS-en og frekvensomformerne var egentlig valgt på forhånd, fordi at om man skal bruke en annen type kommunikasjon her må det byttes ut deler i frekvensomformerne. Det ble aldri vurdert og bytte ut denne kommunikasjonen.

6.4 Generelt

Den løsningen vi har kommet fram til i prosjektet, er vi fornøyd med. Vi føler at med det utgangspunktet og kunnskapen vi har til nå, har vi oppnådd ett godt resultat med riktige og gode bevegelser på plattformen.

Det er noen deler av konstruksjonen av plattformen vi ikke er like fornøyd med:

- Konstruksjonen på leddene av bolter og muttere, disse løsner etter en liten stund med kjøring. Dette skulle absolutt blitt utbedret.
- Det er en del varmgang i skapet. Her bør det monteres inn en vifte slik at det blir luft gjennomstrømning i kabinettet og omformerne kjøles ned. Dette ble aldri noe problem når vi kjørte plattformen i korte perioder i slengen eller med kabinett døren åpen, men viss denne skal kjøres en hel dag kan dette fort bli ett problem.
- Ved stopp av plattform ved f.eks. nødstopp, faller plattformen rett i bakken og låser seg ofte fast i rammen. Her kan det eventuelt settes inn gummidempere for å dempe fallet.

7 KONKLUSJON

Gruppen har utført prosjektoppgaven som planlagt og oppnådd alle målene satt i starten av prosjektet.
Etter å ha brukt mye tid med å få plattformen til å bevege seg etter eget ønske, ble hovedmålet med å hente ut posisjonsdata fra en bil simulator og overføre dette til plattformen en overkommelig oppgave.
Måten programstrukturen er bygget opp på gjør at det uten mye modifikasjoner kan implementeres input fra andre kilder.

Kommunikasjonen mellom PLS og frekvensomformerne viste seg å være den største utfordringen, da beskrivelsen av hvordan kommunikasjonen skulle opprettes var vanskelig å forstå. Det viste seg også at PLS-en som ble brukt ikke støttet alle ønskede funksjoner og ble dermed en ukjent feilkilde.
Etter den PLS-en ble byttet ut løste mange av prosjektets problemer seg.

Gruppen er godt fornøyd med egen innsats, og alt i alt har dette vært et lærerikt prosjekt.

Link til video av plattformen:

http://youtu.be/H_eZfmxhzJY

8 REFERANSER

[1] [Programerbare logiske styringer, Dag Håkon Hanssen, ISBN 978-82-519-2644-7]

[2] [Vedlegg 11 og 12] (kinematikk)

[3] [http://www.codesys.com/]

[4] [http://sourceforge.net/projects/smartphone-imu/]

[5] [MOVIDRIVE MDX61B Extended Positioning via Bus Application Edition 04/2005 Manual]

[6] [http://www.opcfoundation.org/Default.aspx/01_about/

01_whatis.asp?MID=AboutOPC]

[7] [http://www.codesys.com/products/codesys-runtime/opc-server.html]

[8] [http://sine.ni.com/np/app/main/p/docid/nav-104/lang/no/fmid/1762/]

[9] [Objects First with Java A Practical Introduction using BlueJ, David J. Barnes & Michael Kölling, ISBN 978-013-283554-1]

[10] [http://www.python.org/about/]

[11] [Profibus MANUAL – System description: Technology and Application]

[12] [http://people.cs.umass.edu/~verts/cs32/endian.html]

[13] [http://www.modbus.org/faq.php]

[14] [Computer Networking: A top-Down Approach. Sixth edition, James F. Kurose, Keith W. Ross]

[15] [http://www.solidworks.no/sw/6453_NOR_HTML.htm]

[16] [http://www.blender.org/about/]

[17] [http://www.wago.com/wagoweb/documentation/app_note/a3000/a300003e.pdf]

[18] [Sew Eurodrove Manual – Movidrive MD_60A: DriveInverters: Appendum to system manual: BUS positioning - edition 02/2000]

[19] [http://www.mikroe.com/chapters/view/73/chapter-3-iir-filters/]

[20] [Calculus A complete Course Seventh Edition, Robert A. Adams, Christopher Essex]

9 VEDLEGG

[1] – Kildekode Codesys (PDF og PRO)

[2] – Koblingsskjema (PDF)

[3] – Manualer Sew Eurodrive (ZIP)

[4] – 3D modell Blender

[5] – UDP server 2.5 (JAR)

[6] – Kildekode Java server 2.5 (ZIP)

[7] – Jamod Library (PDF)

[8] – Prosjektplan (PDF)

[9] – Python kildekode (PDF)

[10] – 3D tegning plattform (JPEG)

[11] – Kinematikk 1

[12] – Kinematikk 2

[13] – GSD Fil, Movidrive