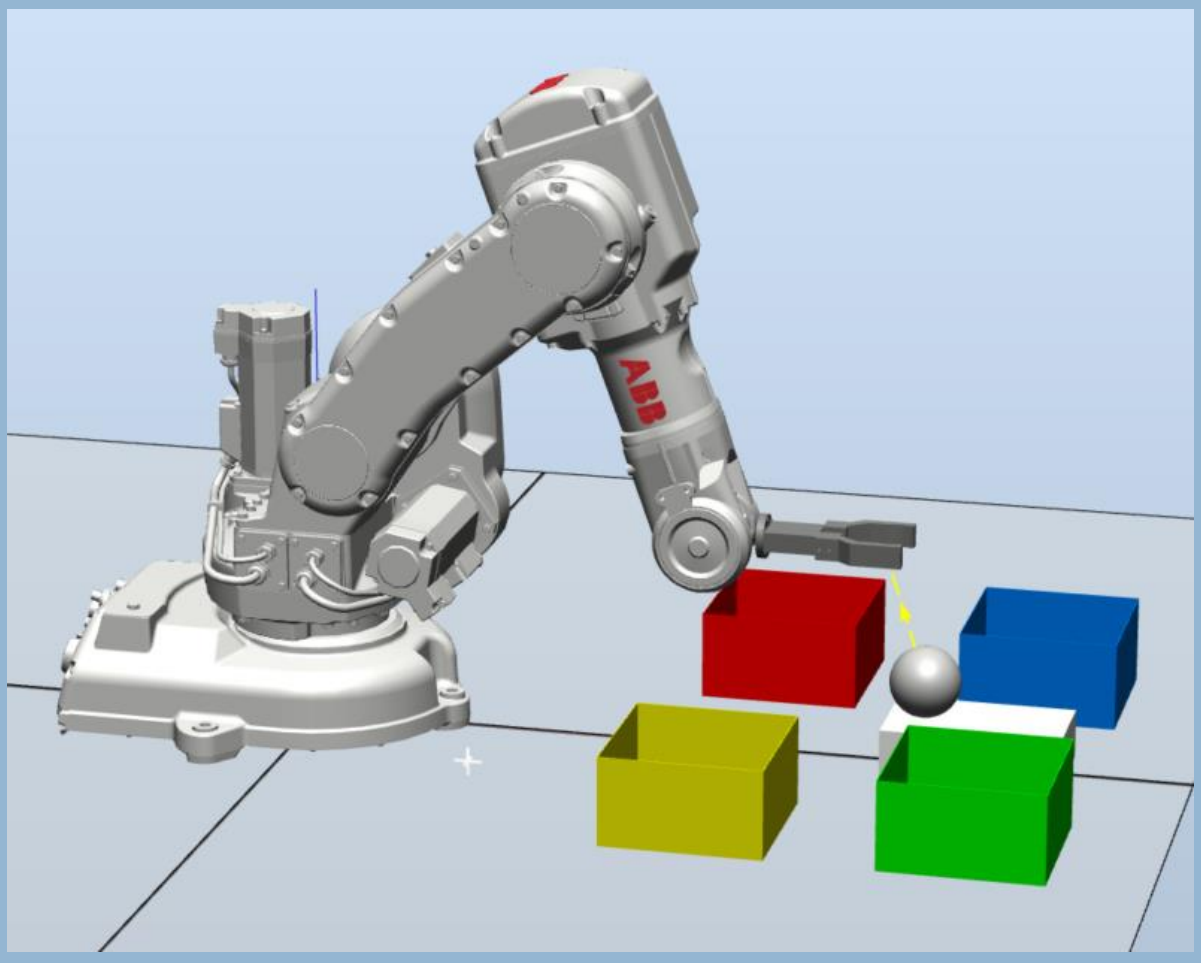
AUTOMATISK SORTERING MED ROBOT



FORORD

Prosjektoppgaven er gjennomført i perioden 20.10.2016 til 20.11.2016 av ingeniørfag – elektronikk og informasjonsteknologi studenter på femte semester ved Høgskolen i Oslo og Akershus. Prosjektoppgaven er teller 30 prosent av karakteren i emnet ELVE3610 Robotteknikk. Oppgaven er todelt, og består av en programmeringsdel og en rapport del. Prosjektbesvarelsen er resultatet av et samarbeid mellom Kristoffer Hegg, Mats Karlsen og Emil Årum.

Prosjektoppgaven var ikke forhåndsdefinert. Vi stod derfor fritt til å velge hva vi ønsket å lage og oppnå under prosjektet. Valget falt på å et realistisk industriproblem, nemlig å benytte roboten til å automatisk sortere objekter. Oppgaven inneholdt en rekke utfordringer som krevde bruk av en rekke kjente og ukjente momenter. Dette gjorte at vi fikk benyttet, repetert og mestret mye av pensumet samt utforsket og lært mye nytt.

Oslo, 18. November 2016

Kristoffer Hegg	Mats Karlsen	Emil Årum

Innholdsfortegnelse

SAMMENDRAG

Denne rapporten redegjør for hvordan vi har løst den obligatoriske prosjektoppgaven i emnet ELVE3610 Robotteknikk. Prosjektoppgaven gikk ut på å benytte en roboten til å automatisk sortere objekter med forskjellig farge i forskjellige beholdere. Dette blir gjort ved at LabVIEW (klient) med LEGO MINDSTORMS fargesensor identifiserer fargen på objektet og sender forespørsel til roboten (tjener) om å flytte objektet. Dette krever blant annet dataoverføring med klient-server arkitektur, flytkontroll, forflytning og manipulasjon av griper.

Resultatet av prosjektet er et funksjonelt, brukervennlig robotsystemet som illustrerer en mulig realisering av en sorteringsrobot på en enkel, men samtidig profesjonell måte. Rapporten forklarer hvordan og hvorfor vi har løst robotsystemet som vi har gjort.

1 INNLEDNING

Oppgaven er basert på undervisningen, øvingene, og laboratorieoppgavene i emnet samt noe selvlært kunnskap. Prosjektoppgaven var ikke forhåndsdefinert. Prosjektgruppen valgte å konstruere et robotsystem som automatisk sorterte objekter med forskjellig farge i forskjellige beholdere. Dette blir gjort ved at LabVIEW (klient) med LEGO MINDSTORMS fargesensor identifiserer fargen på objektet og sender forespørsel til roboten (tjener) om å flytte objektet. Dette krever blant annet dataoverføring med klient-server arkitektur, flytkontroll, forflytning og manipulasjon av griper.

2 ROBOTMANIPULATOR

En robot er en programmerbar, multifunksjonell manipulator designet for å flytte materialer, deler, verktøy, eller spesialiserte enheter via variable programmerte bevegelser for å utføre en rekke oppgaver.

Robotsystemet består av armen, kraftkilden, verktøyet, eksterne og interne sensorer, grensesnittet, datamaskinen og programmet. Roboten klassifiseres etter kraftkilde, geometri, kontrollmetode, og bruksområde. Klassifikasjonene er nyttige for å avgjøre hvilken robot som er riktig for oppgaven. Robotmanipulatorer er en kinematisk kjede satt sammen av koblinger og ledd. Leddene er typisk roterende eller lineære.

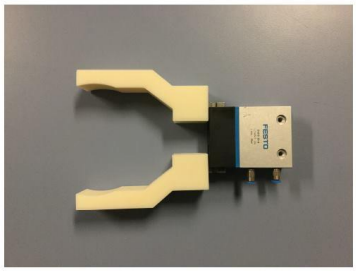
En industrirobot har normalt seks eller færre frihetsgrader (DOF) og klassifiseres kinematisk av de tre leddene på armen. Håndleddet blir beskrevet separat. Leddene i den kinematiske kjeden mellom armen og verktøyet er referert til som håndleddet. Sfærisk håndledd har tre rotasjonsledd som skjærer hverandre i et midtpunkt. Arm og håndledd brukes først og fremst til å posisjonere hånden og verktøyet. Det er verktøyet som utfører oppgaven.

2.1 ABB IRB 140

ABB IRB 140 er en kompakt, kraftig og rask industrirobot med seks akser. IRB 140 er egnet for en rekke bruksområder. Den kan håndtere en nyttelast på 6kg med en rekkevidde på 810mm (til akse fem). Robust utforming med integrerte kabler og kollisjon deteksjon med tilbaketrekking sørger for at roboten er pålitelig og trygg. IRB 140 er tilgjengelig i mange versjoner og kan monteres i alle vinkler. Roboten på materiallaboratoriet er gulv montert og av typen standard og vask.

2.2 VERKTØYET

Griper som kan åpne og lukke er den enkleste typen verktøy, og kun tilstrekkelig for materiale flytting, dele håndtering, eller gripe enkle verktøy. Det designes derfor rekke verktøy som kan utføre spesielle oppgaver og raskt endres som oppgaven tilsier.

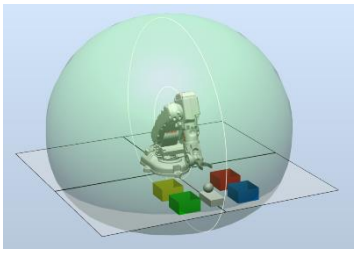


Figur 1 - Griper som er tilkoblet roboten.

Automatisk sortering med robot benytter en griper som verktøy. Griperen er pneumatisk operert av en magnetventil som styrtes av et digitalt signal. Vi har valgt at magnetventilen er normalt lukket. Dette betyr at griperen åpner ved aktivering. Dette øker sikkerheten siden griperen ikke slipper objektet ved strømbrudd.

2.3 ARBEIDSOMRÅDET

Arbeidsområdet til roboten er det totale volumet som er innenfor verktøyets rekkevidde. Arbeidsområdet er begrenset av geometrien til roboten og mekaniske begrensninger på leddene.



Figur 2 - 3D visning av arbeidsområdet til roboten.

3 ROBOTSTUDIO

Robotstudio er et program for modellering, offline programmering og simulering av robotceller. Robotstudio øker lønnsomheten til robotcellen ved å gjøre det mulig å programmere, optimalisere og drive opplæring uten å forstyrre produksjonen.

Robotstudio er en eksakt kopi av programvaren som kjører roboten i produksjonen. Dette gjør det mulig å utføre meget realistiske simuleringer med samme program og konfigureringer som er på roboten i produksjonen. Fordelene med dette er risikoreduksjon, raskere oppstart, kortere produktbytte og økt produktivitet.

3.1 MODELLERING OG SIMMULERING

Modellering og simulering gjør det mulig å visualisere geometrien, bevegelsen og oppførselen til roboten og verktøyet. Visualisering av interaksjonen mellom roboten og omgivelsene øker sikkerheten ved å gjøre det mulig å forutse og unngå eventuelle kollisjoner. Visualiseringen bekrefter også at robotbevegelsen og åpne-lukke verktøyet fungerer som ønskelig. Geometrien gjør det mulig å visualisere robotens arbeidsområde og sørge for at alle «targets» er innenfor robotens rekkevidde.

Griperen er modellert med CAD filene til Festo griperen som er tilgjengelig på fronter og vi benyttet under øving.

4 PROGRAMMET

Roboten skal sortere objekter med forskjellig farge i forskjellige beholdere. Dette blir gjort ved at LabVIEW (klient) med LEGO MINDSTORMS fargesensor identifiserer fargen på objektet og sender forespørsel til roboten (tjener) om å flytte objektet. Dette krever blant annet dataoverføring med klient-server arkitektur, flytkontroll, forflytning og manipulasjon av griper.

Programmet er løst med brukervennlighet, skalering og sikkerhet i fokus. I kapittel 5 og 6 redegjør vi for hvordan og hvorfor vi har løst deler av programmet slik vi har gjort. Hver gang (linje.nr) nevnes i teksten refererer vi til en eller flere linjer i programmet. Bildene er utdrag fra programmet.

5 RAPID

RAPID er et høynivå programmeringsspråk som brukes til å kontrollere ABB industriroboter. Programmet består av en sekvens kommandoer. Programstrukturen likner C, men med litt annerledes syntaks. Programmet deles inn i moduler og kan inneholder prosedyrer, funksjoner og trap rutiner. Sammenligninger, tester og løkker er noen av de tilgjengelige kommandoene.

5.1 ROBTARGET

Robtarget definerer posisjonen til roboten og blir brukt i bevegelse instruksen for å definere posisjonen som roboten skal flytte til. Robtarget definerer posisjonen (x, y, z) og orienteringen (rotasjonen) til verktøyet. Roboten kan oppnå den samme posisjonen på flere forskjellige måter, derfor defineres også akse konfigurasjonen. Det er også mulig å definere posisjonen til ytre akser.

Matriser (Array) er en metode for å organisere og lagre data. Matrisen kan lagre forskjellige datatyper. Hvert element i en matrise har en nummerert posisjon som vi kan benytte for å få tilgang til elementet. Matrisen er Null-indeksert.

Robtarget er definert (line nr. 18 – 23) og samlet i en matrise (linje nr. 24). Dette gjør at vi kan benytte element nummereringen til å enkelt og effektivt bevege roboten. Alle «target» er rotert horisontalt for at bevegelsen til objektet skal være forutsigbar når verktøyet griper og slipper. Alle «target» har samme konfigurering for å minimere nødvendig rotasjon for å bevege roboten til alle «target».

5.2 DATAOVERFØRING

Se avsnitt 6.2.1 Klient-Server arkitektur, 6.2.2 TCP og 6.2.3 Dataoverføring som beskriver kommunikasjonen mellom LabVIEW (klienten) og roboten (tjeneren). Se også avsnitt 6.1 Fargesensor som beskriver signalet som overføres.

TCP er basert på streng kommunikasjon. Dette krever at dataen som skal overføres må konverteres frem og tilbake.

Variabler tillater oss å tildele data et navn som vi kan bruke i programmet i stedet for dataen. Dette gjør at hvis verdien til variabelen endres behøver vi bare å endre verdien til variabelen. Vi behøver ikke å skrive om programmet.

5.3 FLYTKONTROLL

Avgjørelser bestemt av kode kalles for flytkontroll i programmering. IF-test gjør det mulig for programmet å avgjøre hvilken kodeblokk som skal utføres basert på en betingelse. Logiske- og sammenligningsoperatorer sammenligner to variabler for å kontrollere om en bestemt tilstand er oppfylt og returnerer true eller false.

Flytkontroll kontrollerer om det er gyldig farge på objektet og kaller funksjonene for å sortere objektet. Se avsnitt 6.1 Fargesensor som beskriver hvordan fargesensoren fungerer og hvordan fargene blir representert.

5.4 FORFLYTNING

Funksjoner bryter opp store problemer til små håndterbare problemer. En funksjon er en gjenbrukbar kodeblokk designet for å utføre en bestemt oppgave. En funksjon tar noen argumenter, utføre en oppgave på disse argumentene, og deretter returnere et resultat. Dette gjør koden mer fleksibel og vedlikeholds vennlig ved at vi kan bruke samme kode med forskjellige argumenter for å produsere forskjellige resultat. Bruk av instruksjoner som resulterer i output inne i funksjonen er vanligvis ikke beste praksis.

Programmet benytter funksjoner til å plukke (linje nr. 90-95) og sortere (linje nr. 104-111) objektene. Funksjonen benytter «robtarget» i matrisen som argumenter. Dette gjør at vi kan bevege roboten til alle «robtarget» bare ved å endre argumentet. Det er hovedsakelig en lineær forflytning av verktøy midtpunktet (TCP).

Funksjonen benytter «offset» instruksjonen som endrer posisjonen på «robtarget». Dette reduserer antallet «robtarget» nødvendig for å realisere bevegelsen og øker fleksibiliteten.

«Path_Home» funksjonen (linje nr. 78-82) kjøres ved oppstart for å sikre at roboten er plassert i «Target_Home» og verktøyet er åpnet. Dette gjør at programmet kan starte med roboten i en tilfeldig posisjon. Dette krever at forflytningen ikke er lineær.

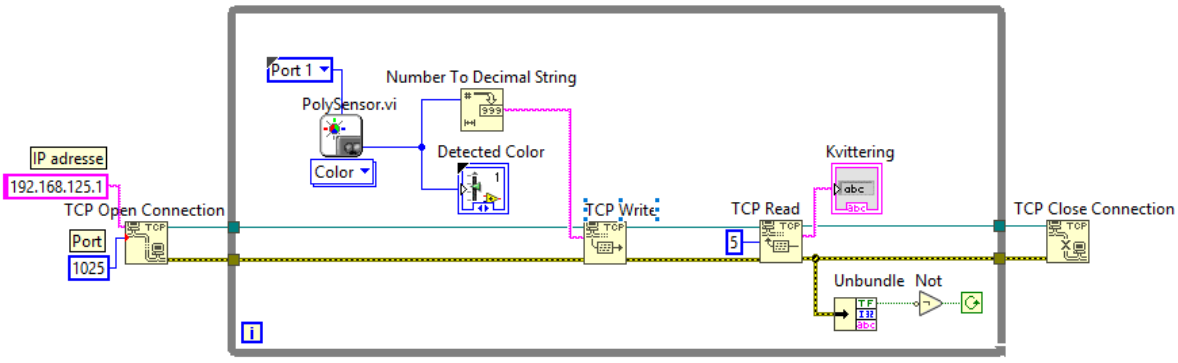
5.5 INTERRUPT

Roboten har en funksjon for stopp løst med en «trap» rutine. Dette gjør det mulig å pause systemet for å tillate for eksempel tømming eller vedlikehold av en sorteringsstasjon. Dette er en viktig og ofte brukt funksjon som er nødvendig om sorteringssystemet skal virkeliggjøres.

Når bryteren DI10_15 aktiveres så aktiveres trap rutinen. Funksjonen definerer en robtarget variabel for huske nåværende posisjon. Funksjonen stopper sorteringssekvensen og lagre robotens nåværende posisjon. Roboten blir stående i påventer av bekreftelse fra bryteren DI10_16 for å gjenoppta sorteringssekvensen fra punktet den ble stoppet.

6 LABVIEW

LabVIEW er et grafisk programmeringsverktøy som kombinerer fleksibiliteten til et programmeringsspråk med styrken til et avansert engineeringsverktøy. LabVIEW passer til ett bredt spekter og applikasjoner og industrier. LabVIEW har en rekke utviklingspakker som kan benyttes for enkelt og raskt løse store kompliserte prosjekter. Signaler og data fra eksternt utstyr kan kobles til og brukes av LabVIEW.



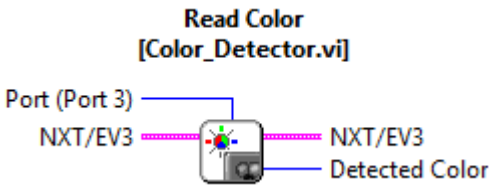
Figur 3 - Utklipp av LabVIEW programmet.

6.1 FARGESENSOR

LEGO MINDSTORMS RGB fargesensor brukes sammen med LEGO MINDSTORMS NXT hjerne. Den kan oppdage 6 forskjellige farger (Rød, Gul, Grønn, Blå, Hvit og sort). Sensoren skiller mellom de forskjellige fargene ved hjelp av refleksjon og absorbsjon. For eksempel vil et rødt objekt reflekterer rødt lys og absorberer grønt lys. RGB-sensoren har en rød, grønn og blå LED og en og en sensor som måler reflektert lys. LED lampene lyser en og en (i et høyt

tempo) og fargen beregnes ut ved hjelp av en algoritme som baserer seg på det reflekterte lyset.

Automatiske sortering med robot benytter en RGB fargesensor til å identifisere fargen på objektet. Fra tidligere er det kjent at LabVIEW kan brukes til å behandle eksternt utstyr og har mange biblioteker. Fargesensoren er koblet til en legohjerne, som videre kommuniserer med datamaskinen via USB. Tredjepartsbiblioteket «Mindstorms Robotics» inneholder «Read NXT Color - Detect color» VI. Dette gjør det mulig å detektere fargen til et objekt og returnere et heltall som representerer den detekterte fargen. Det er forhånds bestemt at: Svart = 1, Blå = 2, Grønn = 3, Gul = 4, Rød = 5, og hvit = 6.



Figur 4 - Read Color - Color_Detector.vi

6.2 KOMMUNIKASJON MELLOM KLIENT OG SERVER

6.2.1 KLIENT-SERVER ARKITEKTUR

Kommunikasjonen har en klient-server arkitektur. Serveren (roboten) lytter på en fast port klar til å motta henvendelser fra klienten (LabVIEW). Serveren vil ofte ha en fast IP adresse. Klienten kan ha varierende portnummer og IP adresse.

Serveren oppretter en tilkobling på porten og adressen. Serveren lytter på denne porten og adressen etter innkommende tilkoblinger, aksepterer tilkoblingen og etablerer forbindelse med klienten. LabVIEW sender forespørsel til roboten om å sortere et objekt og venter på at roboten skal flytte objektet og sende kvittering før LabVIEW sender neste forespørsel.

6.2.2 TCP

Transport kontroll protokoll «TCP» sikrer pålitelig dataoverføring mellom klienten og serveren ved å nummerere byte og kvitere med neste forventede byte. TCP bruker glidende vindu for å optimalisere sendehastigheten.

6.2.3 DATAOVERFØRING

TCP er basert på streng kommunikasjon. Dette krever at dataen som skal overføres må konverteres frem og tilbake.

LabVIEW skriver data til en TCP nettverkstilkobling med «TCP Write». «TCP Read» leser et bestemt antall bytes fra en TCP nettverkstilkobling. Lengden på strengen må defineres.

Det er ingen innebygd mekanisme i rapid for å håndtere flere sendte meldinger. Det er derfor nødvendig at roboten sender en kvittering før LabVIEW sender neste forespørsel. Dette hindrer LabVIEW i å sende en rekke gjentagende forespørsler som ikke roboten kan håndtere og dermed bygger opp buffer som må behandles før neste virkelige forespørsel blir behandlet. Siden LabVIEW programmet ikke sender en ny forespørsel før roboten har fullført den som pågår blir det også mulig å legge på nye baller mens en sortering pågår.

7 DOKUMENTASJON

Dokumentasjon er en viktig del av alt ingeniørarbeid. Dokumentasjonen må være enkel og oversiktlig for hjelpe deg og andre ingeniører å forstå programmet. Dokumenteringen bør være standardisert. Programmene våre er dokumentert med en modifisert standard for

dokumentering av tekstbaserte programmer. Standarden består av generell dokumentasjon etterfulgt av spesifikk dokumentasjon av funksjonene.

```
!-------------------------------------------------------------------------------
!    Function: Overordnet styring av programmet
!    Description: Konfigurerer Trap og TCP/IP kommunikasjon. Kald start av roboten.
!                 Motar forespørsel fra klient. Styrer plukking og plassering av objekt.
!    Parameters:
!    Global: string receive_string - Streng fra klient
!            string parta - delstreng av recieve_string
!            num val - numerisk representasjon av fargen på objektet
!    See Also: <Path_Home>, <Path_Pick>, <Path_Place>, <Trap_Hold>
!-------------------------------------------------------------------------------
```

Figur 5 - Eksempel på dokumentasjon av en funksjon i rapid.

```
Function: TCP kommunikasjon fra LabVIEW til roboten ABB IRB 140
Description:  RGB sensor behandling for detektering av et objekt. Dataen blir pakket og videresendt til
              ABB IRB 140 ved bruk av TCP kommunkasjon. Mottar pakke som bestemmer når neste
              data skal sendes.
Parameters: Write: Polysensor.vi - "Color"  [Unsigned word] Skalert tallverdi i fra RGB sensor
            Read:  Kvittering [String] Kvitteringsmeldning i fra rapid
Global:
See Also: Automatisk_sortering.rspag
```

Figur 6 - Eksempel på dokumentasjon av en funksjon o LabVIEW.

8 KONKLUSJON

Programmet for automatisk sortere objekter med forskjellig farge i forskjellige beholdere fungerer som planlagt. Programmet oppfyller oppgavens spesifikasjoner og gruppens ambisjoner. Prosjektbesvarelsen illustrerer et mulig robotsystem for å realisere en sorteringsrobot på en enkel, men samtidig profesjonell måte. Robotsystemet tilfredsstiller kravene til sikkerhet, dataoverføring og muligheter for utvidelser.

Robotsystemet kan forbedres ved å utvikle et styresystem som gjør det mulig å overvåke og kontrollere roboten. Det kunne også vært mulighet til å sortere etter flere kriterier, for eksempel etter vekt eller form. Vi burde også anskaffe en mer nøyaktig fargesensor, da Lego RGB-sensoren ikke alltid var så konsis som vi ønsket.

REFERANSER

ABB. (2016). «IRB 140 Small powerful and fast 6-axes robot». Hentet fra http://new.abb.com/products/robotics/industrial-robots/irb-140

ABB. (2016). «RobotStudio® It's just like having the real robot on your PC!» Hentet fra http://new.abb.com/products/robotics/robotstudio

ABB. (2010). «Technical reference manual RAPID Instructions, Functions and Data types». Hentet fra https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf

Mark W.Spong, Seth Hutchinson, M.Vidyasagar. (2006). *ROBOT MODELING AND CONTROL*. John Wiley & Sons, Inc.

VEDLEGG

Vedlegg 1: Bruksanvisning

Vedlegg 2: Rapid kode

Vedlegg 3: LabVIEW Program

Vedlegg 4: Flytskjema Robot

Vedlegg 5: Flytskjema LabVIEW

VEDLEGG 1: BRUKSANVISNING

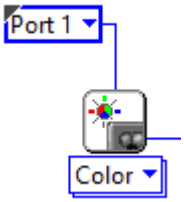
Oppstart prosedyre og tilleggsinformasjon

Brukeren har tilgang til klientens frontpanel i LabVIEW og flexpendanten som representerer som server for roboten ABB IRB 140.

For kommunikasjon mellom klient og server programmene over TCP/IP trenger begge samme port nr. Vanligvis porter mellom 1025-4999 kan brukes. Porter under 1025 kan allerede være tatt. Derfor er **1025** valgt som default port. En feil genereres hvis den valgfrie angitte porten er allerede i bruk. I tillegg trenger klienten og serveren samme IP adresse. Roboten har en fast IP adresse, som er **192.168.125.1.** De eneste gyldige adressene er noen offentlige LAN adresser eller kontrolleren serviceport adresse 192.168.125.1. Dermed er serviceport adressen satt som default adresse.

Før noen av programmene kan starte er det viktig at RGB sensoren er tilkoblet riktig, ellers vil ikke LabVIEW programmet kjøre (får feilmeldingen: «*Cannot find an NXT/EV3 device*.»)

Så sørg for at RGB sensoren er tilkoblet riktig port på NXT hjernen, og den porten samsvarer med porten i LabVIEW programmet (for eksempel port 1). USB kablene kobles mellom USB inngangen på NXT hjernen og USB inngangen til en pc. I tillegg må NXT hjernen være på mens den er tilkoblet (aktiveres ved å trykke den oransje knappen).

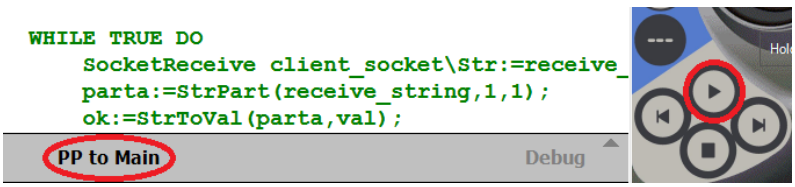


Figur 7 – Tilkoblingsport «port 1» i LabVIEW.

Figur 8 – RGB sensoren er koblet til «port 1» til NXT hjernen.

For at roboten skal være kjørbar og kommunikasjonen med LabVIEW skal fungere, må begge være i run mode. Det er viktig at roboten starter først, da den er server. Før flexpendanten settes i run mode har det blitt en vane å ta «PP to main» under fanen «production window» før det trykkes «run». For da lagres den nyeste versjon av programmet og klargjør programmet som skal utføres.

For start av klient programmet i LabVIEW trykkes «run» knappen. «Abort execution» fungerer som en stop av programmet. Hvis man enten stopper programmet fra LabVIEW eller flexpendanten er systemet lagd slik at programmene må startes i riktig rekkefølge igjen pga automatisk innebygd «timeout» i programvarene.



Figur 9 - "PP to main" og" run" knappene ifra flexpendanten.

Figur 7 - "Run" knapp i LabVIEW.

Roboten er nå klar til å sortere de fargede ballene. Legges det på en valgfri farget ball, sorteres roboten den til riktig boks for den fargen. Når en sortering pågår kan en ny ball legges på, og når roboten kommer tilbake til hjem posisjon starter den straks å utføre den nye sorteringen. Det er også to fysiske digitale innsignaler tilkoblet roboten; DI10_15 og DI10_16. DI10_15 kan brukes til å stoppe roboten hvor som helst når det kjører, og DI10_16 gjenopptar kjøring av roboten fra det punktet den ble stoppet.

VEDLEGG 2: RAPID KODE

```
MODULE Module1
!-----------------------------------------------------------------------------------------------------------------
!    Company:
!    Engineer: Kristoffer Hegg, Mats Karlsen, Emil Årum
!
!    Create Date: 20.10.2016
!    Module Name: Module 1
!    Procject Name: Automatisk sortering
!    Target Devices: ABB IRB 140
!    Description: Sortering av objekter med forskellig farge etter forespørsel fra klient.
!                 Se Automatisk Sortering med Robot rapport for flere detaljer.
!
!    Dependencies: Labview: Automatisk_Sortering_Klient.vi
!
!    Revision:
!    Revision 0.01 - File Created
!    Aditional Coments:
!-----------------------------------------------------------------------------------------------------------------
    CONST robtarget Target_Red:=[[500,250,250],[0.70717,0,0.70717,0],[0,0,-1,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget Target_Blue:=[[800,250,250],[0.70717,0,0.70717,0],[0,0,-1,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget Target_Home:=[[650,0,250],[0.70717,0,0.70717,0],[0,0,0,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget Target_Yellow:=[[500,-250,250],[0.70717,0,0.70717,0],[-1,-1,0,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget Target_Green:=[[800,-250,250],[0.70717,0,0.70717,0],[-1,-1,0,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
    CONST robtarget Target_Pick:=[[700,0,95],[0.70717,0,0.70717,0],[0,0,0,1],[9E9,9E9,9E9,9E9,9E9,9E9]];
    VAR robtarget Target_Array{5}:=[Target_Home,Target_Blue,Target_Green,Target_Yellow,Target_Red];

    VAR socketdev server_socket;
    VAR socketdev client_socket;
    VAR string client_ip:="empty";
    VAR string receive_string:="empty";

    VAR intnum Hold;

    VAR string parta:="empty";
    VAR bool ok;
    VAR num val;
!-----------------------------------------------------------------------------------------------------------------
!    Function: Overordnet styring av programmet
!    Description: Konfigurerer Trap og TCP/IP kommunikasjon. Kald start av roboten.
!                 Motar forespørsel fra klient. Styrer plukking og plassering av objekt.
!    Parameters:
!    Global: string receive_string - Streng fra klient
!            string parta - delstreng av recieve_string
!            num val - numerisk representasjon av fargen på objektet
!    See Also: <Path_Home>, <Path_Pick>, <Path_Place>, <Trap_Hold>
!-----------------------------------------------------------------------------------------------------------------
    PROC Main()
        CONNECT hold WITH Trap_Hold;
        ISignalDI DI10_15,1,Hold;

        SocketCreate server_socket;
        SocketBind server_socket,"127.0.0.1",1025; !!Simmulering: 127.0.0.1, IRB 140: 192.168.125.1
        SocketListen server_socket;
        SocketAccept server_socket, client_socket\ClientAddress:=client_ip;

        Path_Home;

        WHILE TRUE DO
            SocketReceive client_socket\Str:=receive_String;
            parta:=StrPart(receive_string,1,1);
            ok:=StrToVal(parta,val);

            IF  val >= 2 AND val <= 5 THEN
                Path_Pick;
                Path_Place;
            ENDIF

            val := 0; !!For å ungå gjentagelse ved simmulering med putty
            receive_string:="empty";
            SocketSend client_socket\Str:= "ready";
        ENDWHILE
    ENDPROC
```

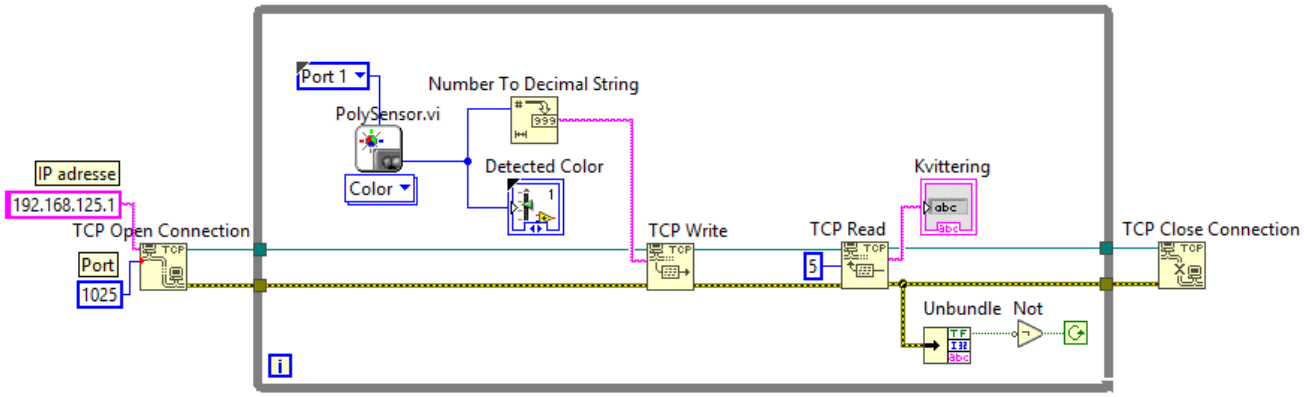
```
!-----------------------------------------------------------------------------------------------------------------
!    Function: Cold start
!    Description: Kjøres ved oppstart for å sikre korrekt opprinnelig verdi. Robotarmen plaasert i Home
!                 og verktøyet åpent.
!    Parameters:
!    Global: Altered: DO10_2 - Åpne(set)/lukke(reset) griper.
!    See Also: <Main>
!-----------------------------------------------------------------------------------------------------------------
    PROC Path_Home()
        MoveJ Target_Home,v100,fine,My_Mechanism_1\WObj:=wobj0;
        Set DO10_2;
    ENDPROC
!-----------------------------------------------------------------------------------------------------------------
!    Function: Plukk object
!    Description: Plukker opp objektet.
!    Parameters:
!    Global: Altered: DO10_2 - Åpne(set)/lukke(reset) griper.
!    See Also: <Main>
!-----------------------------------------------------------------------------------------------------------------
    PROC Path_Pick()
        MoveJ Target_Pick,v100,fine,My_Mechanism_1\WObj:=wobj0;
        WaitTime 0.25;
        Reset DO10_2;
        MoveJ Target_Home,v100,fine,My_Mechanism_1\WObj:=wobj0;
    ENDPROC
!-----------------------------------------------------------------------------------------------------------------
!    Function: Plasser object
!    Description: Plasserer objektet i ønsket beholder.
!    Parameters:
!    Global: Target_Array{} - Sammling av target
!            Altered: DO10_2 - Åpne(set)/lukke(reset) griper.
!    See Also: <Main>
!-----------------------------------------------------------------------------------------------------------------
    PROC Path_Place()
        MoveL Target_Array{val},v100,fine,My_Mechanism_1\WObj:=wobj0;
        MoveL Offs(Target_Array{val},0,0,-50),v100,fine,My_Mechanism_1\WObj:=wobj0;
        WaitTime 0.25;
        Set DO10_2;
        MoveL Offs(Target_Array{val},0,0,0),v100,fine,My_Mechanism_1\WObj:=wobj0;
        MoveL Target_Home,v100,fine,My_Mechanism_1\WObj:=wobj0;
    ENDPROC
!-----------------------------------------------------------------------------------------------------------------
!    Function: Pause roboten
!    Description: Manuell bryter DI10_15 aktiverer trap funksjonen som avbryter oppgaven og blir stående
!                 til manuell bryter DI10_16 tillater fortsettelse.
!    Parameters: robtarget p1 - Robotens nåværende plassering.
!    Global: DI10_15 - Set bryter
!            DI10_16 - Reset bryter
!    See Also: <Main>
!-----------------------------------------------------------------------------------------------------------------
TRAP Trap_Hold
    VAR robtarget p1;
    StopMove;
    StorePath;
    p1:=CRobT();
    WaitTime 0.25;
    WHILE DI10_16=0 DO

    ENDWHILE
    MoveJ p1,v100,fine,My_Mechanism_1\WObj:=wobj0;
    RestoPath;
    StartMove;
ENDTRAP
ENDMODULE
```

VEDLEGG 3: LABVIEW PROGRAM

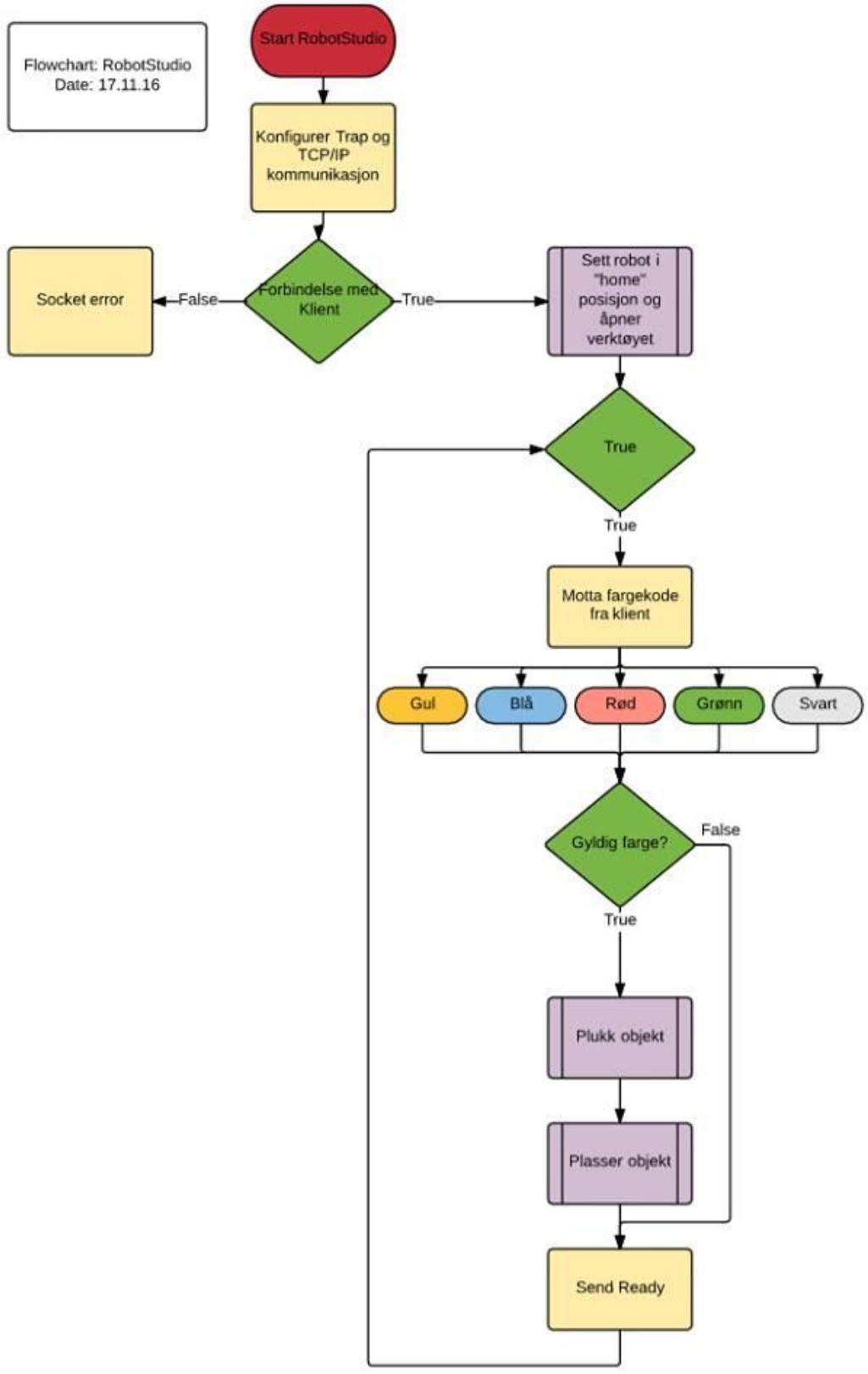
Company: Høyskolen i Oslo og Akershus, Automatisering
Engineer: Kristoffer Hegg, Emil Årum, Mats Karlsen

Create Date: 20.11.2016
Project Name: Automatisk Sortering
Target Devices: ABB IRB 140
Tool versions: LabVIEW 2016
Description: Sensor behandling for automatisk sortering med robot.
Revision:
Revisjon 0.01 - File Created
Additional Coments:



Function: TCP kommunikasjon fra LabVIEW til roboten ABB IRB 140
Description: RGB sensor behandling for detektering av et objekt. Dataen blir pakket og videresendt til ABB IRB 140 ved bruk av TCP kommunkasjon. Mottar pakke som bestemmer når neste data skal sendes.
Parameters: Write: Polysensor.vi - "Color" [Unsigned word] Skalert tallverdi i fra RGB sensor
Read: Kvittering [String] Kvitteringsmeldning i fra rapid
Global:
See Also: Automatisk_sortering.rspag

VEDLEGG 4: FLYTSKJEMA ROBOT



VEDLEGG 5: FLYTSKJEMA LABVIEW