Universitetet
i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering: Bachelor i ingeniørfag / Automatisering og Elektronikkdesign	Vårsemesteret 2021 Åpen
Forfatter(e): Lasse Ånestad og Ruben Skjæveland	
Fagansvarlig: Karl Skretting Veileder(e): Karl Skretting og John Breiland	
Tittel på bacheloroppgaven: Flaskefylling, små flasker Engelsk tittel: Bottle filling, small bottles	
Studiepoeng: 20	
Emneord: Robot, Signalbehandling, Styring, PLS, Modbus, HMI, OPC UA, Instrumentering, Bildebehandling,	Sidetall: 86 + vedlegg/annet: 191 Stavanger 15. mai 2021

Forord

Denne rapporten er skrevet av Ruben Skjæveland og Lasse Ånestad som begge studerer til bachelor i automatisering og elektronikkdesign ved Universitetet i Stavanger. Begge har fagbrev i bunn, Ruben har et fagbrev som tavlemontør og Lasse et fagbrev som automatiker.

Vi har fått tildelt en spennende og omfattende oppgave som har krevet mye tid og energi for å gjennomføre. Det har vært mye nytt å sette seg inn i som har resultert i en meget lærerik prosess.

En stor takk til Karl Skretting for god veileding til rapportskriving og til John Breiland for teknisk veiledning og oppgaveforslaget. Omron skal også takkes for lån av robotarm, og Roald Klingsheim for teknisk hjelp med robotarm. Videre må det også rettes en takk til Romuald Karol Bernacki, Didrik Efjestad Fjereide og Ståle Freyer for gode innspill, diskusjoner og hjelp til oppgaveløsning og generelt organisatorisk i forhold til laboratorie.

Sammendrag

Norwegian Oilfield Supply(NOS) AS har en utdatert produksjonslinje for flaskefylling som behøver oppgradering. Nåværende funksjon til produksjonslinjen er automatisk flaskefylling og korking. Det er hovedesaklig tre komplikasjoner som kan oppstå under produksjon, feil innholdsmengde, ikke tilstrekkelig påskrudd kork og skjev eller mangel på kork.

I denne oppgaven blir det presentert metoder for detektering av de tre spesifikke produksjonsfeilene nevnt overfor, omprogrammering av eksisterende funksjonaliteter på ny PLS og implementering av brukervennlig HMI samt OPC UA kommunikasjon for prosessovervåking fra kontor. De ulike metodene er fordelt inn i tre uavhengige system, robotarm for detektering om flaskekorken er tilstrekkelig fastskrudd, lastcelle for detektering av feil innholdsmengde og bruk av smartkamera til bildeanalyse for detektering av skjev eller mangel på kork. Rapporten gir eksempel på hvordan en robotarm kan brukes for korkmoment testing, utføre signalbehandling av støyforstyrret målesignal fra lastcellen samt fremgangsmetode og PLS programmering for å oppnå presise måleresultat ved dynamisk veiing og bildeanalyse for vurdering av korkposisjonen av flasker i bevegelse. Det blir fremstillet kommunikasjonsmetoder for innhenting av data fra smartkameraet og robotarmen til PLSen for vurderinger av resultatene. Resultat fra simuleringer for de respektive systemene blir vist på enten videoformat eller grafer fra loggført data. Det har blitt utviklet en brukervennlig HMI som gir en god oversikt over prosessen. Ved feil i prosessen vil relevant informasjon kunne leses av på HMI. Det er også muligheter for generell prosesstyring via HMI.

Robotarmen kan programmeres til å utføre korkmoment testing, men behøver tilleggsverktøy og utstyr for denne type oppgave. Lastcellen gir raske og nøyaktige måleresultat ved dynamisk veiing hvis kalibrert og optimalisert for den spesifikke prosessen. Smartkameraet for bildeanalysen gir konsistente vurderinger av flaskekork til PLS hvis flasker passerer i samme posisjon. OPC UA kommunikasjon er mulig å danne både på PLS NX102-1000 og CP1L. For mest robust og enklest oppkobling anbefales NX102-1000 da denne har en egen modul for OPC UA kommunikasjon.

Innhold

Nomenklatur

AOI Automatisk optisk inspeksjon

CBoxDI Controller Box Digital Input

CBoxDO Controller Box Digital Output

CCW CounterClockWise

CW ClockWise

DI Digital Input

DO Digital Output

FH Flaskeholdere

FIR Finite impulse response

FS Flaskestoppere

HMI Human Machine Interface

HTTP Hypertext Transport Protocol

HTTPS Hypertext Transport Protocol Secure

IS Induktiv sensor

NOS Norwegian Oilfield Supply

OPC UA Open Platform Communication Unified Architecture

PLS Programbar Logisk Styring

RS Reflektiv sensor

RSR Reflektiv sensor robot

SSL Secure Sockets Layer

ST Structure Text

TCP Transmission Control Protocol

Kapittel 1

Introduksjon

1.1 Innledning

Automatiserte systemer som mer eller mindre styrer seg selv med liten eller ingen menneskelig interaksjon er ofte ønskelig og et behov i ulike virksomheter. Bruk av automatisering kan gi mange fordeler, eksempelvis økt effektivitet og lønnsomhet. I produksjonsprosesser som denne oppgaven baserer seg på hvor større mengder av like produkter skal leveres, er bruk av automatiserte systemer å foretrekke.

Denne oppgaven er gitt av John Breiland i samarbeid med bedriften Norwegian Oilfield Supply (NOS). NOS er en bedrift som blant annet produserer og distribuerer ulike væsker til industrien. For å kunne produsere sine produkter i større kvantum til sine kunder, på en optimal og effektiv måte, bruker bedriften automatiserte flaskefyllingslinjer som fyller væske i flaskebeholdere. Utfordringer bedriften nå støter på er at inspeksjon av væskemengde og sjekking av kork må gjøres manuelt. John ønsker nå å se på muligheter for å øke automatiseringen samt utarbeide en solid tilbakemeldingsstruktur for prosessfeil knyttet til en udatert flaskefyllingslinje som bedriften opererer med. Det er denne flaskefyllingslinjen som er utgangspunktet for oppgaven.

Rapportoppbygning

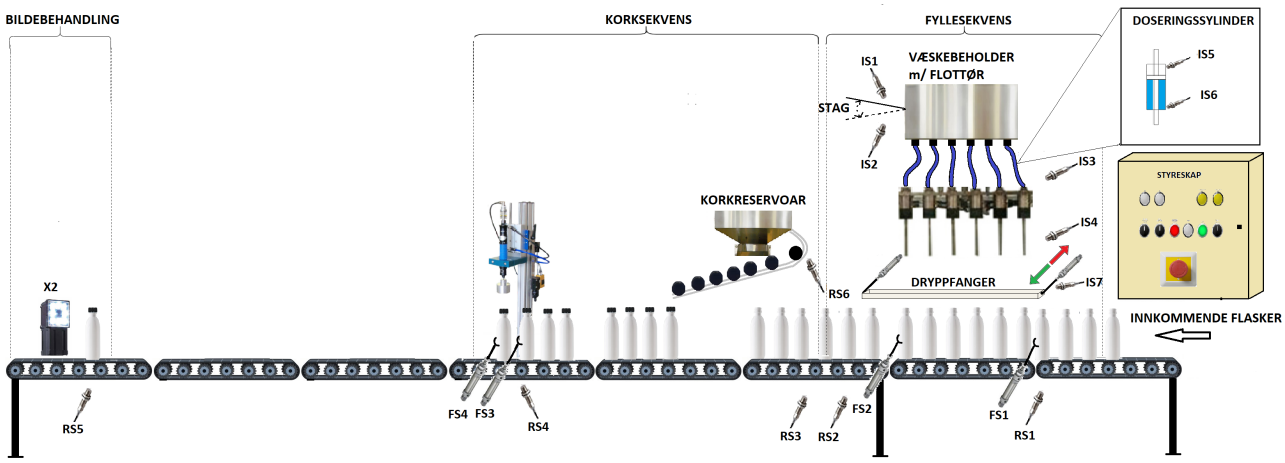
Rapporten er delt inn i fire kapitler. Første kapittel er introduksjon der oppgaven presenteres. Kapittel to er hoveddelen til rapporten som beskriver og presenterer fremgangsmåter og resultat av de ulike deloppgavene knyttet til prosjektet. I kapittel tre er det en diskusjon av de resultat som er blitt presentert og forslag til endringer utover det som kommer frem i rapporten. Kapittel 4 konkluderes rapporten. I slutten av dokumentet finnes relevant vedlegg knyttet til oppgaven.

1.2 Opprinnelig funksjonalitet

I oppstarten av prosjektet ble det utført et bedriftsbesøk til NOS hvor flaskefyllingslinjen ble presentert. John Breiland, som har programert funksjonaliteten til linjen slik den fungerer idag, beskrev prosessen under besøket.

På grunn av ulike komplikasjoner så var ikke prosessen i sin helhet operativ under besøket. Det som ble demonstrert var forflytning av flasker på transportsbånd, væskefylling i flasker og noe sylinderstyring. Funksjonalitet for korking ble beskrevet. Funksjonalitet for smartkameraer i eksisterende prosess er ukjent.

Slik prosessen ser ut idag kan visualiseres som i figur 1.1



Figur 1.1: Prisippiell skisse av prosessen slik den er idag. Flaskene kommer inn på høyre side. Seks flasker blir skilt ut for fylling. Dryppfanger posisjonerer seg under fylledyser etter fylling for å forhindre søl. Korkreservoar sørger for at korker blir lagt riktig vei på flasketut. En og en flaske blir skilt ut for korking ved korkmaskin. I enden av transportsbåndet er det montert to smartkameraer, et kamera på hver side av transportsbåndet, for bildebehandling av kork. Langs transportslinjen er det montert opp sylindere for å stoppe og å skille ut flasker. Ulike sensorer er montert på nødvendige plasser.

John gav tilgang på relevante skjemategninger som har blitt brukt for grundigere analyser av prosessens virkemåte. De opprinnelige skjemategningene er å finne i vedlegg B.1.

1.3 Oppgavebeskrivelse

Det er et ønske om å se på ulike tilleggsfunksjonaliteter som eventuelt kan implementeres til prosessen for å oppnå høyere grad av automatiseringen og sikre at levert produkt er i henhold til gitte standarder.

Tilleggsfunksjonaliteter som har blitt presentert i prosjektet:

- Bruke robotarm til å utføre testing om kork er tilstrekkelig påskrudd
- Bruke lastcelle for å verifisere at det er tilført riktig mengde væske i flaske
- Bruke smartkamera til bildebehandling av kork. Få tilbakemelding hvis kork er skjev eller mangel på kork
- Utforme en oversiktlig og brukervennlig HMI (Human Machine Interface)
- Etablere OPC UA kommunikasjon mellom PLS og datamaskin for å få oversikt over prosessen fra bedriftens kontorer

Eksisterende funksjonaliteter er også blitt omprogrammert til ny PLS.

1.4 Litt om utstyr, utvikling og utførelse

Prosessens funksjonaliteter blir idag styrt gjennom en PLS av typen CP1L [4]. De eksisterende funksjonalitetene har blitt omprogrammert utifra skjemategninger gitt i B.1, observasjon av prosessen og antatt virkemåte til å fungere på NX102-1000 [14].

Oppgaveforslaget ble utviklet på ett koblingsoppsett med PLS av typen NX102-1000 som har en integrert modul for OPC UA kommunikasjon, og en HMI skjerm av typen NA5 [5]. PLS av typen CP1L og en Raspberry PI [15], ble brukt for å teste om OPC UA kommunikasjon kan etableres til det eksisterende systemet. Lastcelle AL6B [36] [3] ble brukt for veiing og smartkamera FQ2 [8] brukt til bildebehandling. Fra Omron ble det lånt en kollaborativ robotarm i en begrenset periode. Programmet Sysmac Studio [16] ble brukt for å programmere NX102 og CX-programmer [13] ble brukt for programmere CP1L.

Alt av utført arbeid har blitt simulert og testet på universitets laboratorie, så fysisk oppkobling og testing på selve produksjonlinjen har ikke vært mulig. Derfor har arbeidet basert seg på en del antagelser om hvordan prosessen er tenkt å operere. Uforutsette endringer er å forvente om løsningsforslaget presentert i oppgaven blir iverksatt.

Relevant utstyr

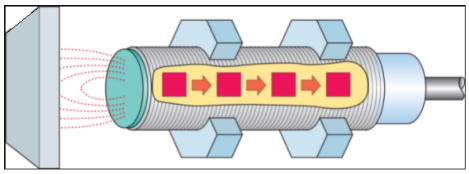
I kommende del blir relevant utstyr knyttet til prosessen og løsning av oppgaven presentert. Det deles opp i "Hardware" for å beskrive fysisk utstyr og "Software" for å beskrive de ulike programmeringsprogrammene.

1.4.1 Hardware

Sensorer

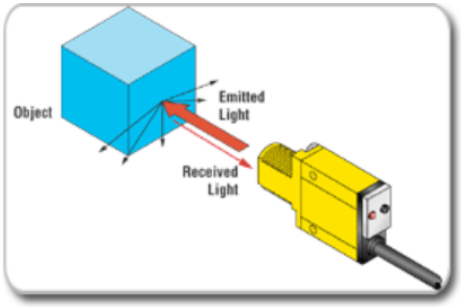
I prosessen er det koblet opp flere induktive og reflektive sensorer. Litt om forskjellene mellom de blir nå beskrevet.

Induktive sensorer
Induktive sensorer kan brukes til å detektere metalliske objekter. Prinsippet baserer seg på en induktor som utvikler elektromagnetisk induksjon. Når et metall er innen rekkevidde for den dannede induksjonen så registrerer sensoren metallet grunnet metallet endrer den induserte spenningen. Siden den induktive sensorer ikke interagerer med det våte element så fungerer sensoren utmerket også under våte forhold [19]. Figur 1.2 viser en induktiv sensor som detekterer et metallisk objekt.



Figur 1.2: Induktiv sensor som detekterer et metallisk objekt. Bildet hentet fra [20]

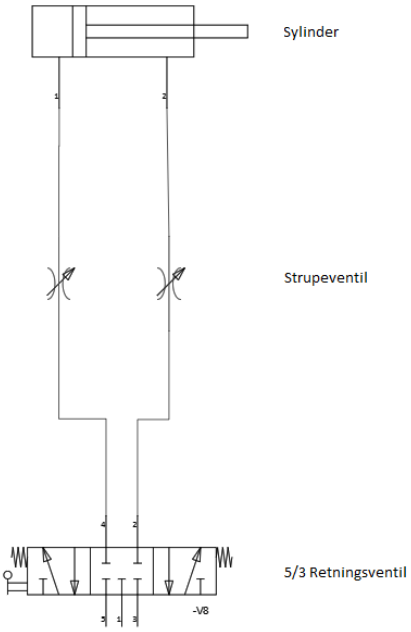
Reflektive sensorer
De reflektive sensorene som er brukt i denne prosessen er av typen diffuse sensorer. Prinsippet baserer seg på en sender og en mottaker. En lysstråle sendes fra sender, når lyset treffer et objekt så reflekteres lysstrålen tilbake til mottaker som registrerer lyset og gir en tilbakemelding. Både sender og mottaker er i samme "hus" [33]. Figur 1.3 viser den prinsipielle virkemåten.



Figur 1.3: Diffus sensor som detekterer objekt ved hjelp av reflektert lys. Bildet hentet fra [33]

Sylinderer

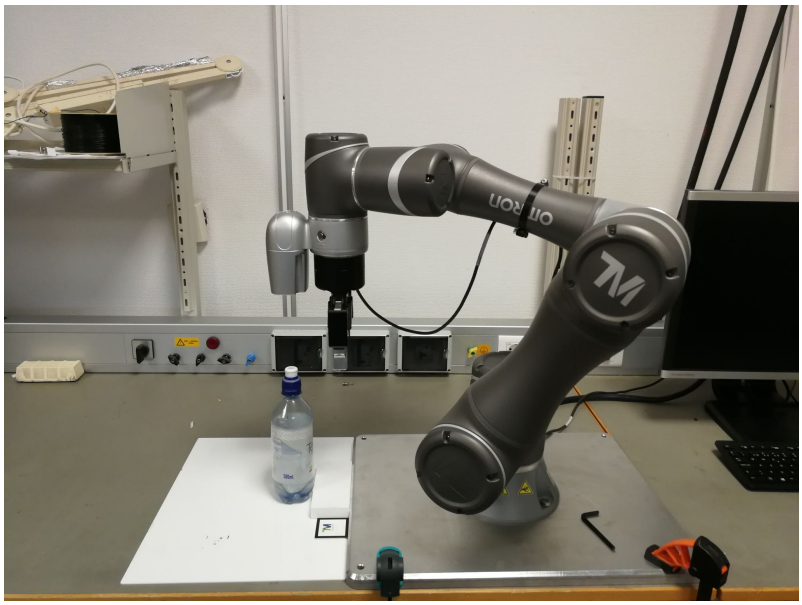
Det er montert opp sylindere langs linjen for å stoppe flaskene i enkelte av sekvensene. Funksjonaliteten til sylinderne er basert på pneumatisk (luft) styring ved hjelp av en retningsventil. Sylinderne er bygget opp med et sylinderhus og et stempel med stempelstag som er på innsiden av huset. På hver side av huset er det tilkoblingspunkter for luft. Retningsventilens posisjon bestemmer i hvilken side sylinderen skal tilføres luft, ergo, om stempelet blir presset inn eller ut. Skjematisk tegning av sylinder, strupeventil og retningsventil er vist i figur 1.4.



Figur 1.4: Skjematisk tegning av sylinder, strupeventil og retningsventil. Bildet hentet fra B.1 s.11

TM kollaborativ robotarm

For å verifisere at kork er blitt tilstrekkelig påskrudd etter korking vil det bli brukt en seksakset kollaborativ robotarm fra Techman Robot Incorporation levert av Omron [17], se figur 1.5. Robotarmen har et integrert kamera for bildebehandling og påmontert en griper fra ROBOTIQ [11].



Figur 1.5: Viser robotarmen over flaske. Bildet tatt med eget kamera

Veiecelle

For prinsippets skyld blir denne lastcellen, som vist i figur 1.6 brukt under labtesting. AL6B [36] bruker null defleksjons målesprinsippet med hjelp av strekklapper i en målebro "wheatstone bridge"[3] for å konvertere mekanisk energi om til spenningssignal. Lastcellen AL6B har en fullskala output på 0.9±0.1mV/V [16] hvor i dette tilfellet exitasjonsspenning tilført er 5V. Dermed kan fullskala måleområde til lastcellen fremstilles i et intervall på [0,4.5±0.5]mV.

Figur 1.6: AL6B-3.0kg-0.4B. Bilde hentet fra [36]

FQ2 smartkamera

I oppgaven skal det brukes et smartkamera av typen FQ2 [8] fra Omron. FQ2 kan brukes til ulike oppgaver knyttet til bildebehandling. I dette prosjektet er kameraet bruk for å detektere om kork er tilstede på flaske og om kork er riktig montert. Figur 1.7 viser hvordan smartkameraet ser ut.

Figur 1.7: FQ2 smartkamera. Bilde hentet fra [28]

1.4.2 Software

TM Flow

TMflow [2] er et programmeringsverktøy utviklet av Techman Robot Inc. Det er et grafisk HMI grensesnitt som gjør det relativt enkelt og praktisk å programmere roboten. Programmeringen er hovedsaklig basert på flytskjemategning som knytter sammen ulike noder. For å strukturere programmet og å unngå for mye spagettikode så kan "SubFlow"-noden benyttes. "SubFlow"-noden brukes for å pakke seksjoner av programmet inn i individuelle noder, dette vil redusere størrelsen og skaper en bedre oversikt av det fullstendige programmet.

Touchfinder for PC

"Touchfinder for PC" [8] er et program utviklet av Omron. "Touchfinder for PC" brukes til å konfigurere smartkameraet som er brukt i denne oppgaven. Programmet brukes for å lære kameraet opp til å detektere ønsket detaljer i bildet. Det er flere funksjoner som kan brukes for å lære kameraet til å detektere detaljer i bildet, men i dette prosjektet så var "Edge position" hovedsaklig brukt. Denne funksjonen ser etter kanter/fargeoverganger i bildet på definerte områder.

Matlab

Matlab [22] er en programmeringsplatform designet for analysering og designing av systemer. Denne oppgaven blir Matlab brukt for matematiske beregninger for signalbehandling og analysering av data fra simuleringer.

UaExpert

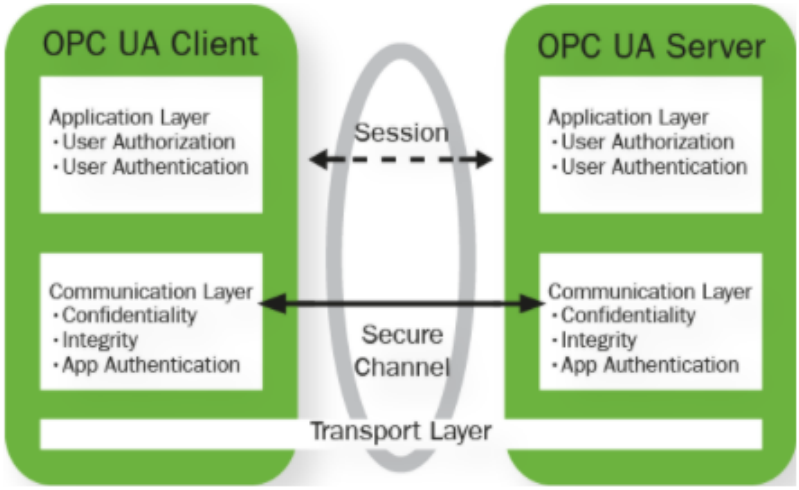
UaExpert [34] er et program som brukes for å konfigurere datamaskin som en klient for OPC UA kommunikasjon. I UaExpert så kobles datamaskin opp mot en server, som i dette tilfellet enten var Raspberry PI eller PLS. Når oppkoblingen er godkjent og korrekt utført så kan man lese av status på de variabler som er delt over OPC UA protokollen.

1.5 Grunnleggende teori om OPC UA

Open Platform Communication Unified Architecture (OPC UA) [30] er en plattformuavhengig kommunikasjonsprotokoll utviklet av OPC Foundation. Protokollen brukes for å opprette kommunikasjon med høy sikkerhet mellom ulike maskiner, eksempelvis mellom datamaskiner og industrielle maskiner. Kommunikasjonen kan brukes til ulike formål som overvåking og loggføring av et anlegg, endring av variabler i en prosess og lignende.

OPC UA baserer seg på en såkalt klient/server kommunikasjon der serverne sin oppgave er å vente på henvendelser fra klienter [37][29]. Når server mottar en henvendelse fra klient så sendes det data over til klienten. Oppsettet kan også konfigureres slik at når det innkommer nye data på server så skal data kontinuerlig sendes til klient.

Selve kommunikasjonstransporten er bygd opp av ulike lag. Transportlaget består av standard TCP/IP protokoll i bunn etterfølgt av protokollene SSL (Secure Sockets Layer), HTTP (Hypertext Transport Protocol) og HTTPS (HTTP Secure). Deretter er det to lag der et lag er ansvarlig for å monitorere kontakten mellom klient og server [35] og et lag som etablerer en sikker kanal mellom klient og server [32]. Figur 1.8 illustrerer i grove trekk de ulike lagene som inngår i kommunikasjonstransporten mellom klient og server.



Figur 1.8: OPC UA kommunikasjonslag. Bilde hentet fra [32]

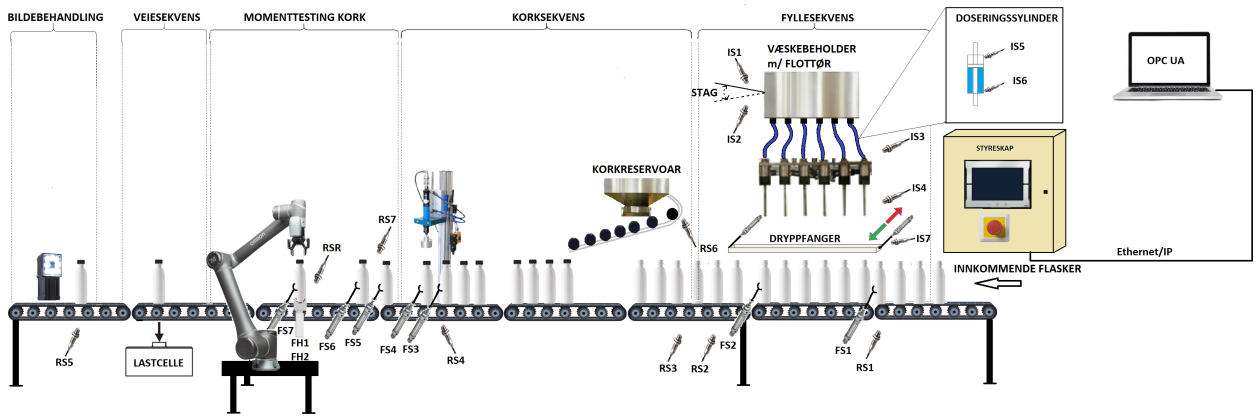
Kapittel 2

Flaskefyllingsprosessen inkludert tilleggsfunksjonaliteter

I dette kapitellet blir fremgangsmåter og resultater knyttet til oppgaven presentert. Først kommer det en oversikt over hvordan en ny tenkt prosess vil kunne se ut med tilleggsfunksjonaliteter. Så blir forslag for fyllesekvens og korkesekvens i ny PLS presentert. Deretter kommer løsningforslag for bruk av robotarm til sjekk av korkmoment, lastcelle for veiing av flasker og smartkamera for bildeanalyse av kork. Videre kommer det forslag til metoder for å etablere OPC UA for eksisterende PLS og for ny PLS. Siste del av kapitellet presenteres en oversiktlig HMI knyttet til prosessen.

2.1 Hvordan forløpet for fyllelinje med tilleggsfunksjonaliteter er tenkt

Prosessen med tilleggsfunksjonaliteter kan visualiseres slik som i figur 2.1. Fyllesekvens og korkesekvens er tenkt å fungere likt som idag.



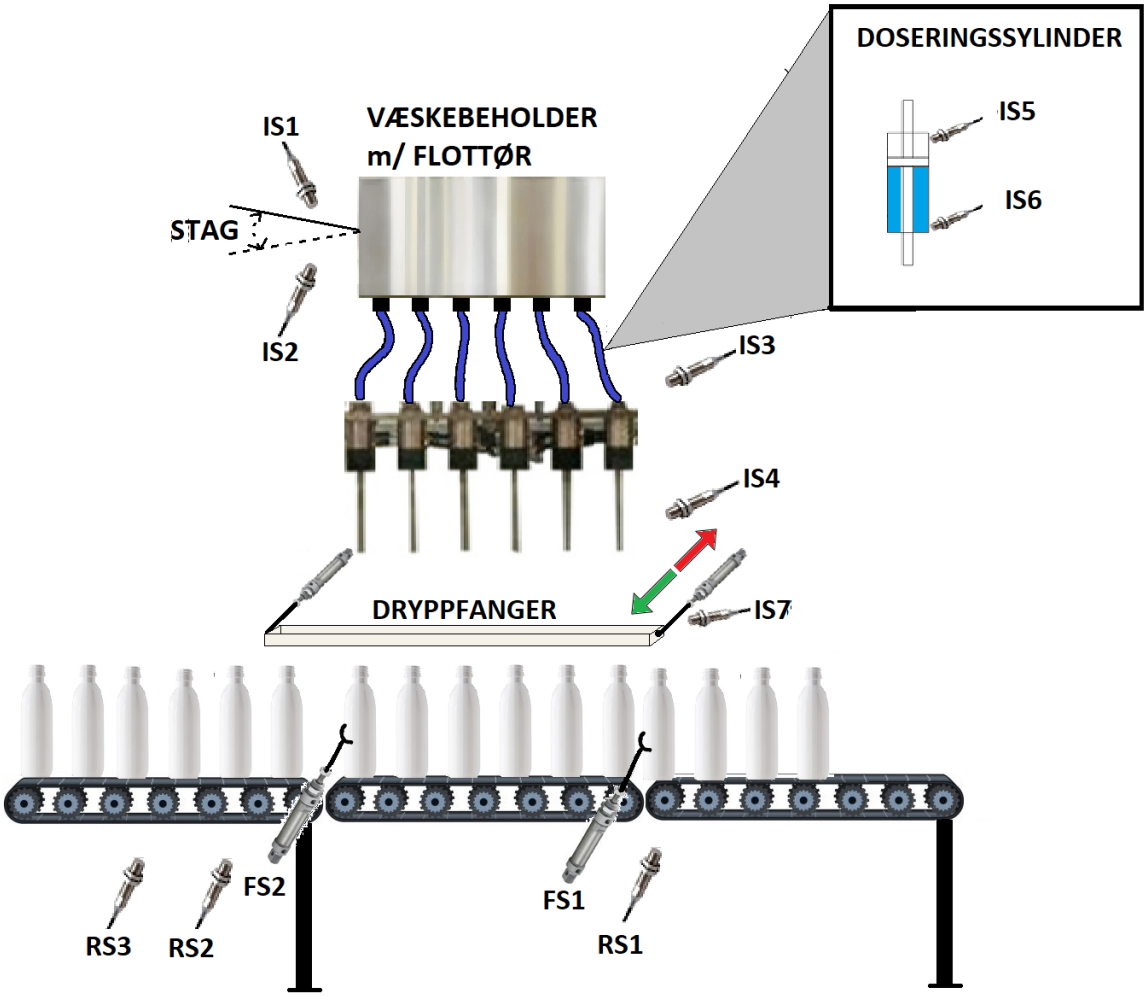
Figur 2.1: Prisippiell skisse av prosessen slik den er tenkt med inkluderte tilleggsfunksjonaliteter. Flaskene blir først plassert på et transportbånd som er i kontinuerlig bevegelse. Deretter går flaskene gjennom ulike sekvenser. Første sekvens er fyllesekvensen hvor flaskene skal fylles med ønsket væske. Neste sekvens er korksekvensen der flaskene blir forseglet med en kork. I sekvensen momenttesting av kork skal en robotarm anvendes for å sjekke at kork er skrudd til med rett moment. I veiesekvensen vil det bli brukt ei lastcelle for å verifisere riktig mengde væske i flaske. Etter veiesekvensen kommer flaskene til bildebehandling der et smartkamera skal sjekke om kork er tilstede og riktig montert. På HMI skjerm vil en få relevant informasjon fra prosessen, ha valget mellom å kjøre prosessen i auto eller manuell og muligheter for kalibrering av lastcelle. Prosessens forløp skal kunne bli observert på bedriftens kontorer med bruk av OPC UA kommunikasjonsprotokoll via ethernet/IP.

2.2 Fyllesekvens

2.2.1 Funksjonsbeskrivelse

I foreslått løsning beholdes denne delen av prosessen slik den er idag. Vårt arbeid her er da å dokumentere eksisterende prosess, signaler brukt for å styre denne, samt foreslå (nytt) program for å styre prosessen med NX-102.

Figur 2.2 viser hvordan fyllesekvensen er tenkt.



Figur 2.2: Fyllesekvens: Sekvensen har to sylinderstyrte flaskestoppere (FS), FS1 og FS2, som skiller ut seks flasker om gangen. Når seks flasker er skilt ut så fylles flaskene med væske via seks dyser som blir senket ned ved bruk av pneumatisk styring. Etter fyllingen så blir dysene hevet opp igjen og en dryppfanger kommer inn under dysene for å forhindre søl. Deretter går FS2 inn for å sleppe flaskene videre mens FS1 holder igjen kommende flasker. Etter tre sekunder går FS1 inn og når seks flasker har passert RS2 vil FS2 gå ut igjen. IS1 og IS2 er induktive sensorer som detekterer når stag er i høy eller lav posisjon. Høy posisjon indikerer lavt væskenivå og lav posisjon indikerer høyt væskenivå. Det er alarmer tilknyttet til IS1 og IS2, se seksjon 2.75. Detaljert beskrivelse av sekvensen og aktuelle sensorer blir beskrevet videre i kapitelet.

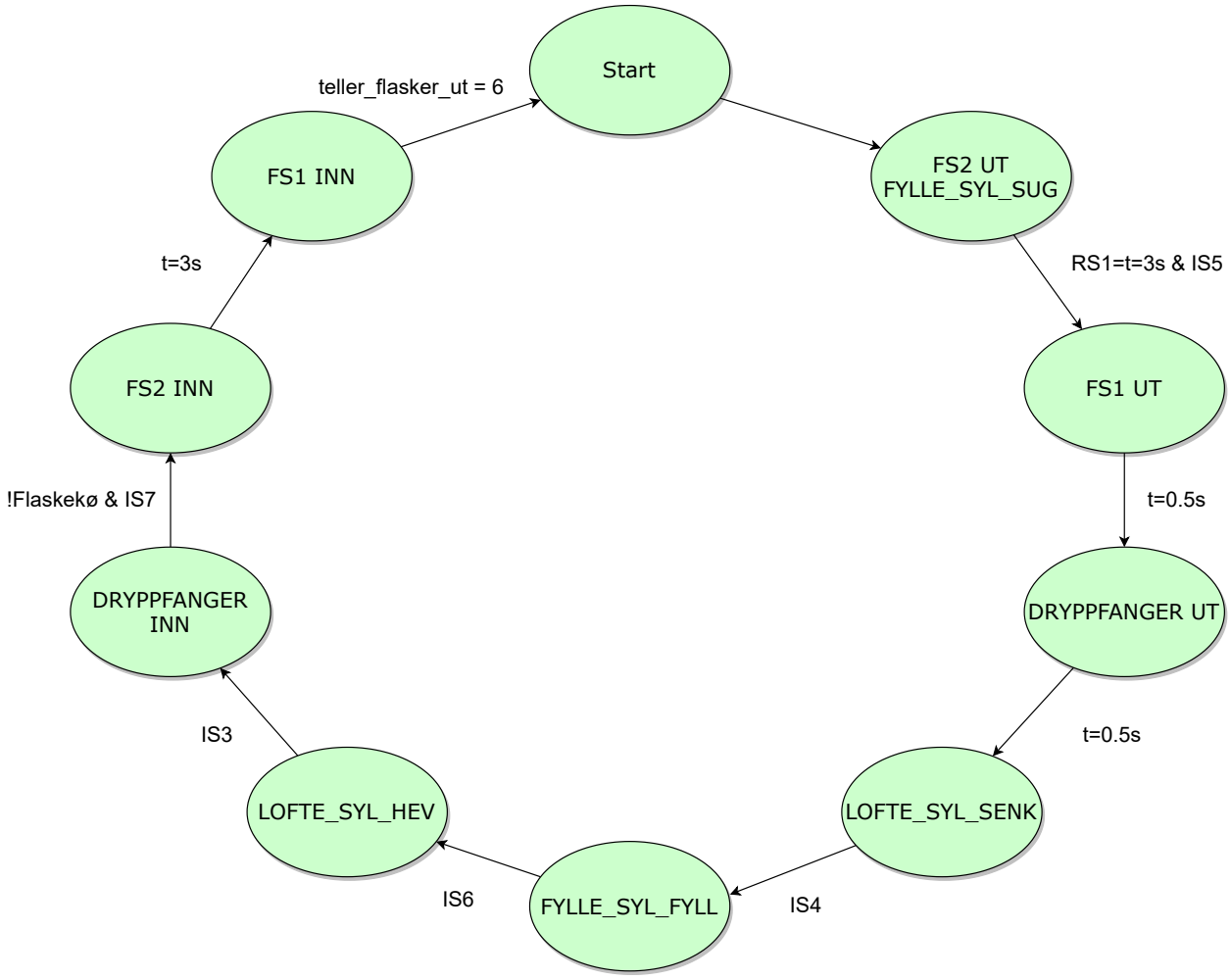
I/O Liste

Variabelkommentarer er tilknyttet relevante variabler for å hjelpe til med forståelsen for tilstandsdiagrammet vist i figur 2.4

Name	Data Type	Constant	Comment
DRYPPFANGER	BOOL	☐	Dryppfanger
FlaskeKø	BOOL	☐	Denne blir aktiv hvis RS3 er høy i 3 sekunder
FS1	BOOL	☐	Flaskestopper 1
FS2	BOOL	☐	Flaskestopper 2
FYLLE_SYL_FYLL	BOOL	☐	Presser ut vesken av doseringskammer
FYLLE_SYL_SUG	BOOL	☐	Fylling av doesringskammer
IS3	BOOL	☐	Løftesylinder hevet
IS4	BOOL	☐	Løftesylinder senket
IS5	BOOL	☐	Doseringskammer fullt
IS6	BOOL	☐	Doseringskammer tomt
IS7	BOOL	☐	Dryppfanger ute
LOFTE_SYL_HEV	BOOL	☐	Hever dysene
LOFTE_SYL_SENK	BOOL	☐	Senker dysene
RS1	BOOL	☐	Sensor for registrering av 6 flasker inne
RS2	BOOL	☐	Sensor for telling av flasker ut
RS3	BOOL	☐	Reflektiv sensor som registrerer flaskekø
Transportband	BOOL	☐	Transportband på

Figur 2.3: Variabler for fyllesekvens

Tilstandsdiagram for fyllesekvens



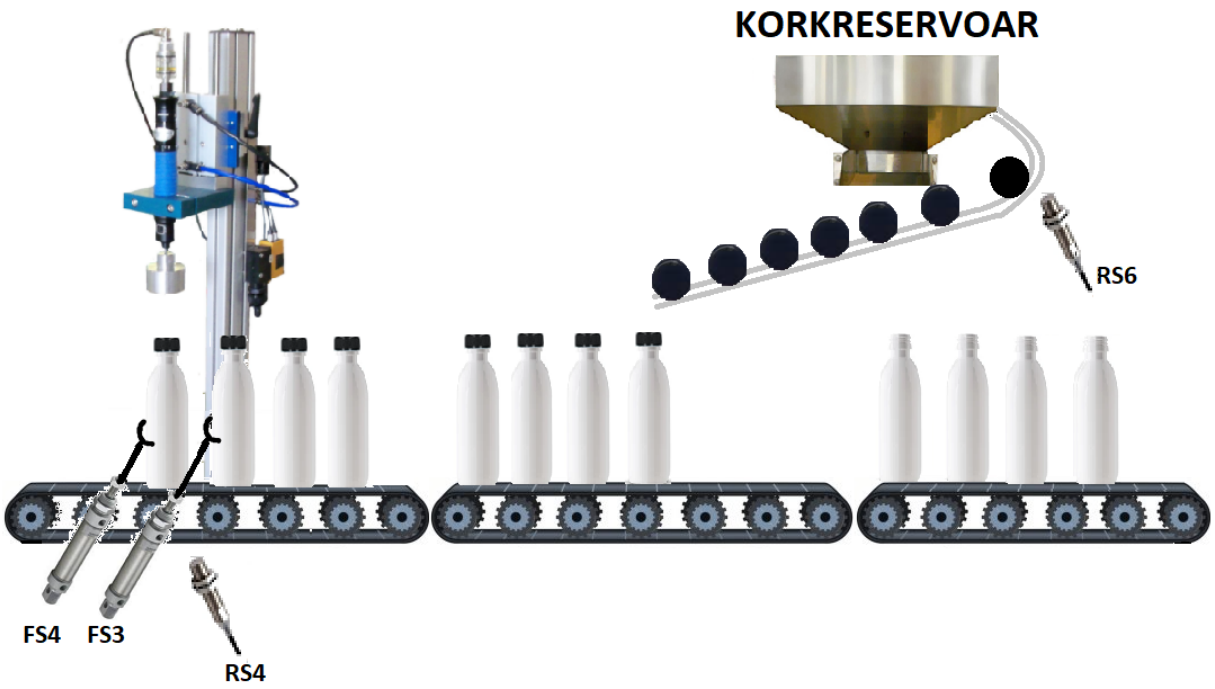
Figur 2.4: RS1 må være høy i 3 sammenhengende sekunder for at det er registrert 6 flasker inne. ”t=0.5s” betingelsene er for å simulere tidsbruken flaskestopperene bruker får å nå endeposisjon. ”!flaskekø” betingelsen er for å forhindre viderkjøring av sekvensen hvis det er flaskekø. Betingelsen ”teller_flasker_ut” er tilknyttet RS2, som teller at alle flaskene er ute av fykkeseksjonen. PLS program for fyllesekvensen kan man finne i vedlegg A.1

2.3 Korkesekvens

2.3.1 Funksjonsbeskrivelse

Også korksekvens foreslås å bli beholdt slik der er nå. Vårt arbeid her er da å dokumentere prosessen, signaler brukt for å styre denne, samt foreslå (nytt) program for å styre prosessen med NX-102 [14].

Figur 2.5 viser hvordan korksekvensen er tenkt.



Figur 2.5: Korksekvens: Før selve korkingen blir korkene plassert på flasketuten via en maskin som sørger for at korkene blir plassert riktig vei. Deretter går flaskene videre til selve korkmaskinen som fester korkene. Også her benyttes to sylindere for å stoppe flaskene og skille ut en og en flaske for korking. FS4 går ut i det sensor etter flaskefylling registrerer at fylling er ferdig. Når flaske blir stoppet av FS4 blir dette registrert av RS4 som gir signal videre til FS3, slik tilhørende sylinder går ut. Nå aktiveres korkesekvensen. Når korksekvens er ferdig går FS4 inn og sender ferdig korket flaske videre.

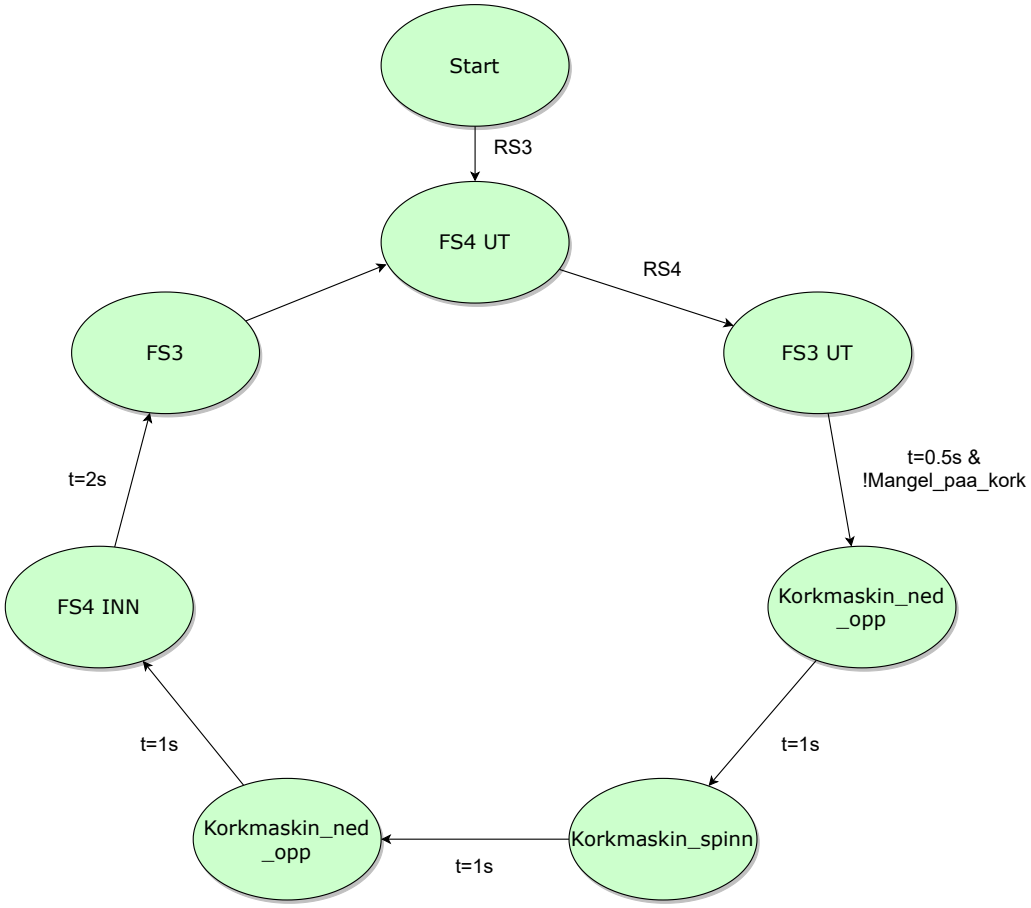
I/O liste og Tilstandsdiagram Korksekvens

Variabelkommentarer er tilknyttet relevante variabler for å hjelpe til med forståelsen for tilstandsdiagrammet vist i figur 2.7

Name	Data Type	Constant	Comment
FS3	BOOL	☐	Flaskestopper 3
FS4	BOOL	☐	Flaskestopper 4
Korkmaskin_ned_opp	BOOL	☐	Korkmomentmaskin, impulsstyrt [1]ned og [2]opp
Korkmaskin_Spinn	BOOL	☐	Egen utgang i PLS for styring av korkmomentmaksin spinn
Mangel_paa_kork	BOOL	☐	Global variabel for mangel på kork feilmelding
RS3	BOOL	☐	Køsensor fra fyllesekvens, brukes for å verifisere at flasker kommer
RS4	BOOL	☐	For å detektere at flaske har ankommet
RS6	BOOL	☐	Sjekker om det er mangel på korker i korkreservoar

Figur 2.6: Globale variabler korksekvens

PLS program for korksekvensen finner man i vedlegg A.2

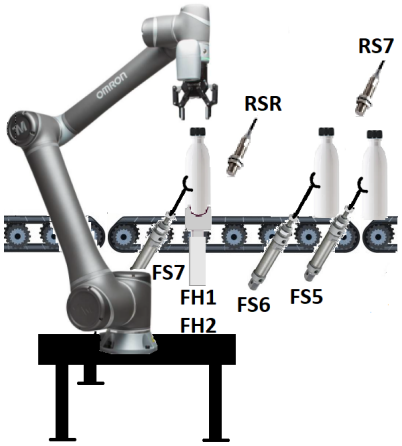


Figur 2.7: Tilstandsdiagram korksekvens: RS3 brukes for å detektere at flasker er på veg fra fyllesekvens. betingelsene "t=1s" er for å simulere tiden korkmaskin bruker på de respektive funksjonene. Betingelsen "t=2" er for gi ferdig korket flaske tid til å forflytte ut av sekvensen før ny flaske kommer inn

2.4 Korkmomenttest med bruk av robotarm

2.4.1 Innledning

Bruken av en kollaborativ robotarm for korkmoment testing er en mulig ekstra del til prosessen for å sjekke om kork er tilstrekkelig påskrudd. Robotarmen kommer med et integrert smartkamera som er montert på siste ledd ved griperverktøyet. I denne oppgaven blir smartkameraet brukt for å detektere at flaske har ankommet posisjonen for selve korkmoment testen. Det har blitt etablert modbus kommunikasjon mellom PLS og robotarm for å kunne lese av eventuelle feilmeldinger på HMI skjerm. Det er også fremstillet eksempel på nødvendige tilleggsverktøy og en sylinderstyrt flaskereguleringsekvens for å etablere et fullstendig system for korkmoment test.

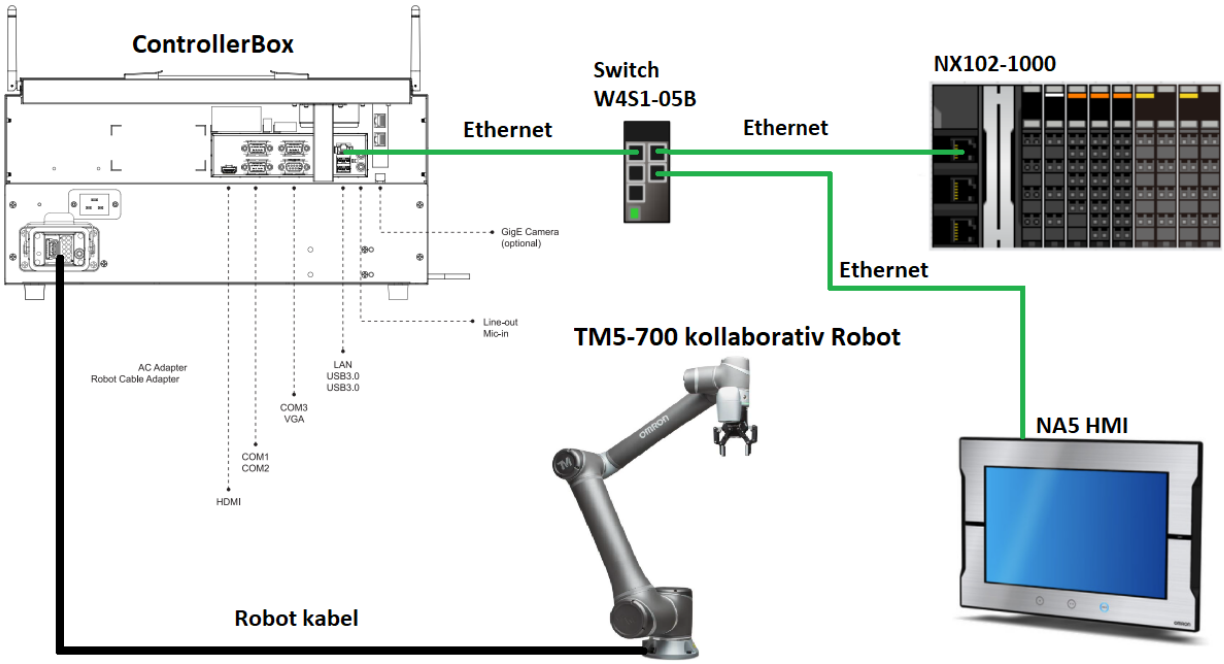


Figur 2.8: Selve sekvensen er tenkt slik at en også her bruker sylinderstyrte flaskestoppere for å skille ut den flasken som skal testes. Flasken stoppes av FS7 slik at flasken er innenfor kameraets søkeområde. FS5 og FS6 regulerer matingen av flaskene slik kun en flaske kan ankomme momenttestingen om gangen. Deretter gjør robotarmen testing av kork. Flaskeholderne FH1 og FH2 brukes for å holde flaske fast under momenttest. Testingen utføres ved at robot lokaliserer flaske ved hjelp av smartkamera for å så bevege seg ned til kork, gripe tak og vri korken med klokken. Oppstår det en gitt motstandmoment i robotleddet under vridning, så er dette klassifisert til godkjent. Fullføres vridningen uten motstand så er dette klassifisert til ikke godkjent, altså kork er ikke tilstrekkelig påskrudd. Etter testing går FS7 inn, slik at flaske transporteres videre i prosessen. Når flaske har passert FS7 så går sylinder ut igjen. FS6 går da inn slik at ny flaske kommer til robotarmen og prosessen repeteres.

Grunnet mangel på utstyr, så har vi ikke hatt muligheten til å simulere tilleggssylindere for flaskeregulering (FS5-7) og Flaskeholderene, men det har blitt gjort fysiske simuleringer for "Godkjent" og "Ikke Godkjent" korkmoment. Resultatene blir demonstrert i prosjektvideoen:

`https://youtu.be/cAYFE2o8z5s?t=70` fra tid 1:10-2:17

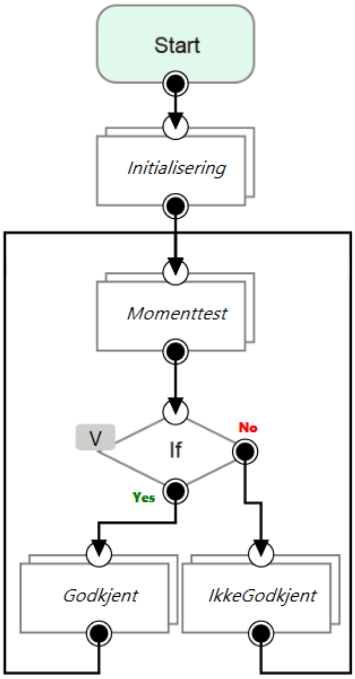
Grensesnitt for Robotarm styring



Figur 2.9: Den praktiske oppkoblingen for styring av robotarmen.

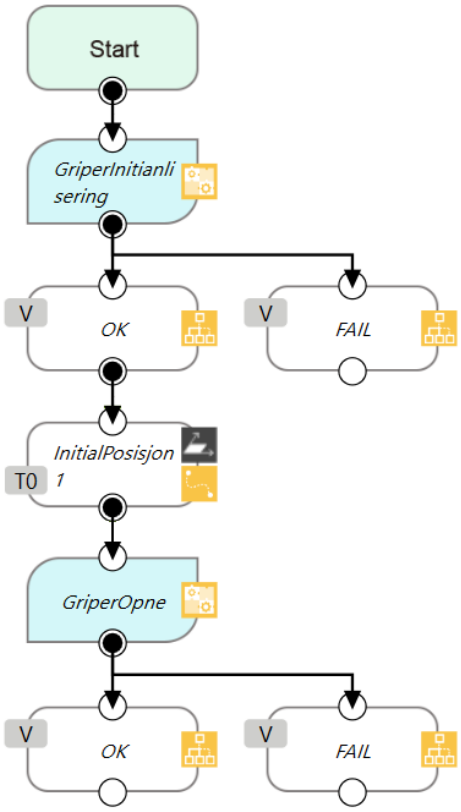
2.4.2 TMFlow Program for å styre robotarm

Programmet er delt opp i flere subflow-noder (”Initialisering”, ”Momenttest”, ”Godkjent” og ”IkkeGodkjent”) som vist i figur 2.10. Hver subflow-node har sin en egen funksjon.



Figur 2.10: Programmet startes ved enten trykk på ”Play” på ”Robot Stick” eller HMI gjennom modbus fra PLS. Noden ”Initialisering” fører robot til initialposisjon og initialiserer griper konfigurasjoner. Subflow-noden ”MomentTest” bruker robotens smartkamera for å detektere at flaske har ankommet, deretter beveger roboten seg ned til flaskekork for testing av korkmoment. IF-Noden sjekker om momentterskelverdien er oversteget (godkjent hvis oversteget). Hvis korken er fastskrudd, så går programmet ned i subflow-noden ”Godkjent” som åpner griperen og går tilbake til subflow noden ”MomentTest”. Hvis korken ikke er fastskrudd så går programmet i subflow-noden ”IkkeGodkjent”, som gjør opp til to ekstra tester der griperen vrir 30° per test. Hvis korken ikke er fastskrudd etter tre tester totalt så går programmet tilbake til subflow-noden ”MomentTest”. En mer detaljert beskrivelse av funksjonaliteten til hver subflow-node er beskrevet i neste seksjon.

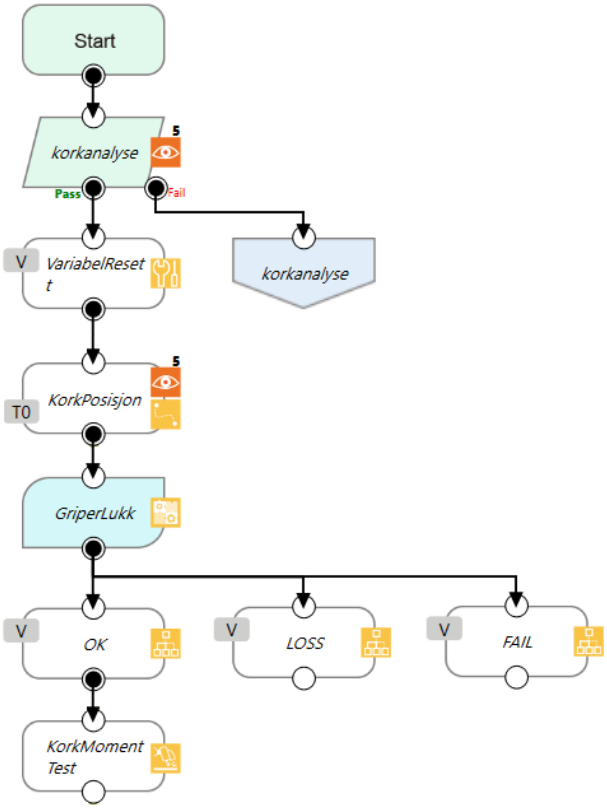
Initialisering



Figur 2.11: TMFlow: subflow-node Initialisering

- **Start** Aktiveres når programmet går inn i subflow-noden
- **GriperInitialisering** setter griperstyrken til 50% og griperfart på 50%
- **OK** hvis griperinitialisering er ferdig, **FAIL** hvis noe forhindrer griperinitialiseringen.
- **InitialPosisjon1** er faste startkoordinater for roboten ved oppstart.
- **GriperOpne** Åpner griperen i tilfellet programmet ble avsluttet i lukket gripertilstand.
- **OK** hvis griperen er åpnet, **FAIL** hvis noe forhindrer åpning.

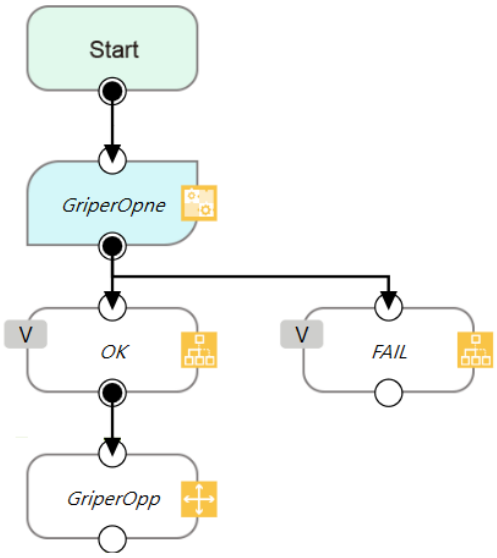
Korktesting



Figur 2.12: TMFlow subflow-node Korktesting

- **Start** Aktiveres når programmet går inn i subflow noden
- **Korkanalyse** Korkanalyse blir gjort av en "vision"-node, denne noden gjør en objektbasert bildeanalyse. Analyserer om flasken står i posisjon til korkmoment testing. Utfyllende forklaring av vision-noden er beskrevet i seksjon .
- **VariabelReset** resetter variabelen brukt for å gi tilbakemelding til PLS hvis korkmomentet ikke er godkjent.
- **KorkPosisjon** robot beveger seg til forhåndsdefinert posisjon til flaskekork.
- **GriperLukk** Lukker griperen
- **OK** hvis objekt er grepet, **LOSS** hvis mangel på objekt og **FAIL** hvis andre komplikasjoner forhindrer lukking av griper.
- **KorkMomentTest** Tester korkmomentet. Utfyllende forklaring til funksjonaliteten til denne noden er beskrevet i seksjon 2.4.2

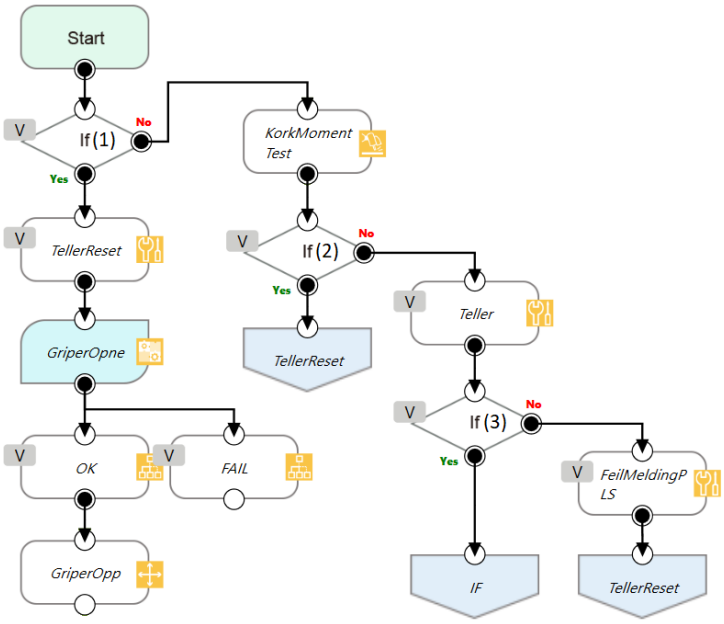
Godkjent



- **Start** aktiveres når programmer går inn i subflow-noden
- **GripeOpne** Åpner griperen
- **OK** hvis griperen er åpnet, **FAIL** hvis noe forhindrer åpning.
- **GriperOpp** Er en "Move"-node som er programmert til å flytte roboten i en rett linje lags Z-aksen (opp). Dette er for å forhindre kollisjon med flasken ved bevegelse til initial posisjon for "Korkanalyse" noden.

Figur 2.13: TMFlow subflow-node Godkjent

IkkeGodkjent

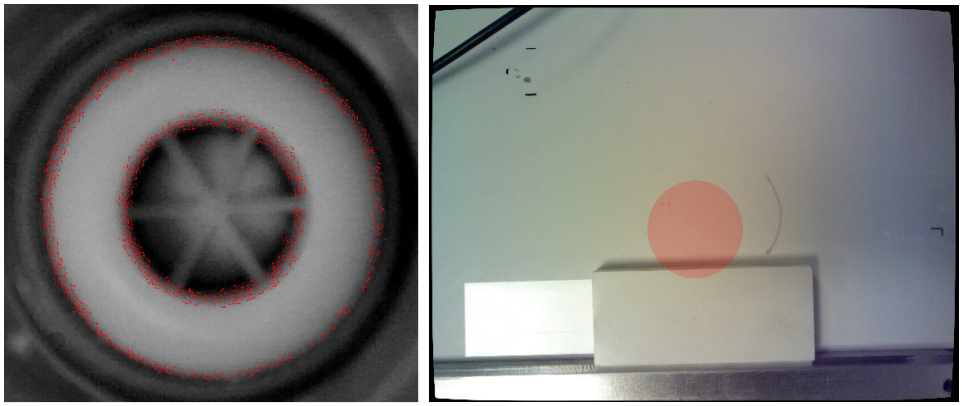


Figur 2.14: TMFlow subflow IkkeGodkjent

- **Start** aktiveres når programmer går inn i subflow-noden
- **IF (1)** Sjekker hvor mange ganger griperen har testet korkmomentet med variabelen "teller". "Yes" hvis "teller=3" og "No" hvis "teller<3"
- **KorkMomentTest** Tester korkmomentet. Utfyllende forklaring av funksjonaliteten til denne noden er beskrevet i seksjon 2.4.2
- **IF (2)** Sjekker om korken er fastskrudd etter ny moment test. "Yes" hvis kork er fastskrudd og "No" hvis ikke fastskrudd.
- **GoTo TellerReset** dette er en "GoTo"-node, som omdirigerer programmet til "TellerReset"
- **Teller** noden teller antall ganger griperen har testet korkmomentet med 1 inkrement intervall
- **IF (3)** Sjekker antall moment-tester. "Yes" hvis "teller<3", "No" hvis "Teller=3".
- **GoTO IF** omdirigerer programmet til "IF(1)" for ny gjennokjøring av subflow noden.
- **FeilmedingPLS** Skriver boolsk feilmelding til PLS (CBoxDO[2]=True). Denne feilmeldingen indikerer at korken ikke er fastskrudd etter tre moment-tester hvor griperen har rotert 30° per test.
- **GoTo TellerReset"** omdirigerer programmet til "TellerReset"
- **TellerReset** resetter teller variabelen.
- **GriperOpne** åpner griperen.
- **GriperOpp** Er en "Move"-node som er programmert til å kun flytte roboten i Z-aksen(opp). dette er for å forhindre kollisjon med flasken ved bevegelse til initial posisjon for "Korkanalyse" noden.

Bildeanalyse av kork: Vision Node

For dette prosjektet så var det mest hensiktsmessig å ta i bruk kamera for å detektere når flaske kommer til robotarmen. Ved å bruke "Vision"-noden så aktiviseres kamera til roboten. I "Vision"-noden har man forskjellige applikasjoner for bildeanalyse. Applikasjonen "AOI" (Automated Optical Inspection) ble brukt for å definere mønstergjenkjenning til flaskekorken som vist i 2.15a og avgrenset søkeområde for mønstergjenkjenningen markert i rødt som vist i figur 2.15b. Hvis kameraet ser det spesifikke mønsteret i figur 2.15a innenfor søkeområdet i figur 2.15b, vil "Vision"-noden godkjenne mønsteret og deretter gå videre til neste node i programmet.



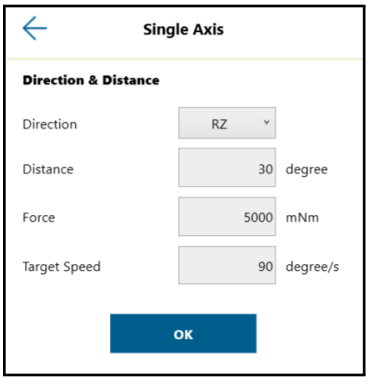
(a) Mønster som kameraet ser etter **(b)** Område kamera søker i

Figur 2.15

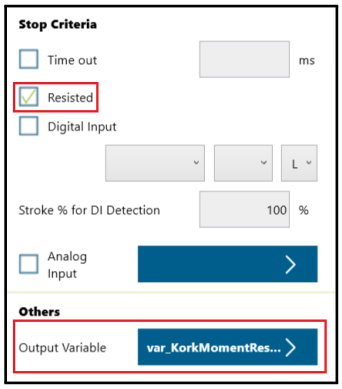
KorkmomentTest: Compliance Node

"Complience"-noden er brukt for selve moment-testingen av korken. Noden har en egen funksjon for styring av enkeltakser. For å kunne vri griperen (verktøyet) så blir enkeltaksen "RZ" brukt i dette tilfellet. Som vist i figur 2.16a kan man justere på rotasjons distanse, moment (Force) og rotasjonshastighet (Target Speed). Noden godtok ikke negative tall for rotasjonsdistanse, som resulterte i at moment-testingen må bli gjort i rotasjon med klokken, heretter omtalt som CW (Clockwise). CW moment-testing går ut på samme prinsipp som testing CCW (Counter Clockwise), bare at du har nå mulighet til å korrigere feilen med å stramme korken under testing. I TMflow er det satt en minimums grense på 5000 mNm for dreiemoment i aksen "RZ".

For å registrere motstandsmomentet til en fastskrudd kork, kan man i "Complience"-noden ta i bruk stop criteria betingelsen "Resisted" som vist i figur 2.16b. Hvis motstanden er lik eller overstiger dreiemomentet 5000mNm i dette tilfellet, vil aksen stoppe å rotere. Da vil roboten registrere at rotasjonen er "Resisted".



(a) Compliance node enkelakse modifikasjoner



(b) Compliance node Stop Criteria modifikasjoner

Figur 2.16

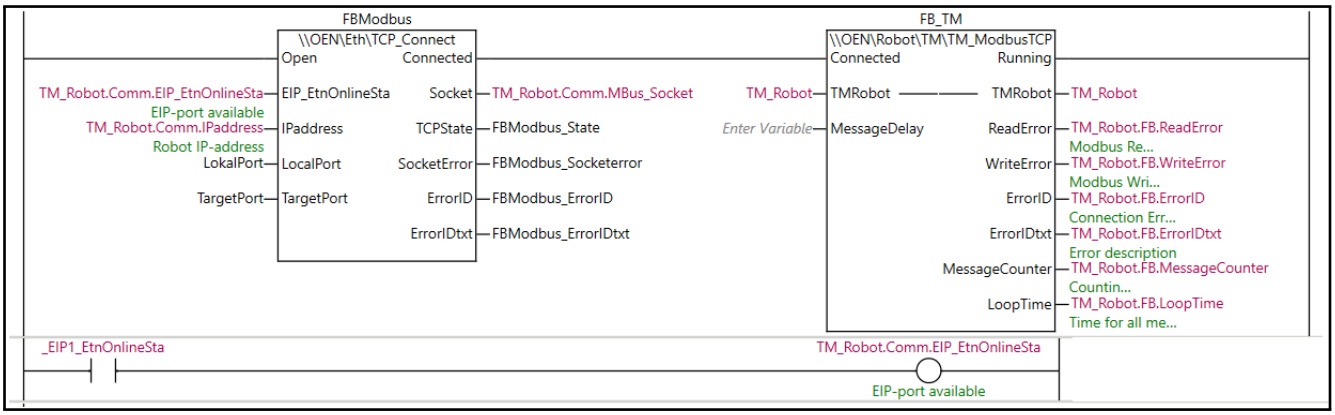
Hver eneste stop criteria betingelse har en egen forhåndsbestemt verdi tilknyttet til dem, hvor "Resisted" har heltalls verdien "5" som vist i figur 2.17. Når betingelsen "Resisted" blir oppnådd, så skriver noden verdien 5 til utgangsvariabelen "var_KorkMomentResisted". Denne variabelen blir brukt som betingelse for IF-løkkene for moment-test i programmet.

Value	Description
2	Timeout
3	Distance Reach
4	Digital Input (or Analog Input) triggered after the Stroke %
5	Resisted
6	ERROR (including TCP speed over limit, incorrect timing of DI triggered, and etc.)

Figur 2.17: Stop Criteria tabellverdier

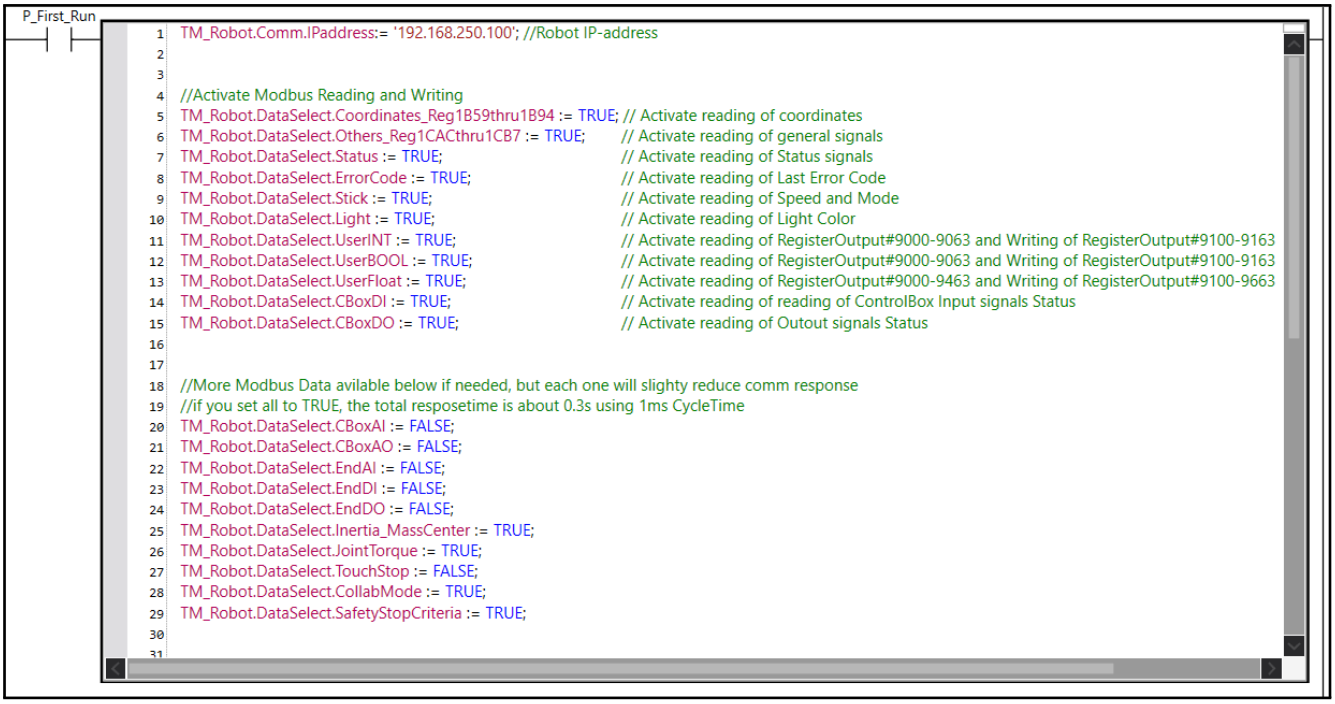
2.4.3 Modbus kommunikasjon mellom PLS og Robot

Modbus TCP kommunikasjon er etablert mellom PLS og Robot for å kunne utføre Robot stick funksjoner og å lese feilmeldinger fra robot på HMI. For å etablere modbus kommunikasjon mellom NX102 og roboten så blir Omron sine funksjonsblokker, "TCP_Connect" og "TM_Modbus TCP" tatt i bruk. "TCP_Connect" etablerer selve kommunikasjonen ved å definere slave IP-adresse, master og slave socket adresse (portnummer) som inputs. "TM_ModbusTCP" funksjonsblokken utfører selve kommunikasjonen for skriving og lesing av adresser til og fra robotens register. Kommunikasjonsprogrammet inkludert funksjonsblokkene er kopiert fra datablad "Star!TmCobot" som man kan finne på Omron sin supportside [18]. Funksjonsblokkene er vist i figur 2.18



Figur 2.18: Funksjonsblokkene brukt for Modbus TCP kommunikasjon fra Omron [18]

I figur 2.19 viser kode for aktivering av robotregister for lesing og skriving via funksjonsblokken "TM_Modbus TCP".



Figur 2.19: Strukturvariabelen "TM_Robot" innholder alle registeradressene til roboten. Man aktiverer de spesifikke adressene som er ønsket for skriving og/eller lesing ved å sette registeradressen til å være "TRUE" som vist i koden. Funksjonsblokken "TM_ModbusTCP" i figur 2.18 vil da utføre selve modbus kommunikasjonen for alle de aktiverte registeradressene i "TM_Robot", gitt at denne strukturvariabelen er en input til denne funksjonsblokken.

For å kunne skrive/lese data fra enkelte robotregister gjennom modbus, kan man bla i selve datastrukturvariabelen "TM_Robot", eksempel på ST kode for å opprette en boolsk variabel for styring av robot stick funksjonen "PlayPause" vist i figur 2.20

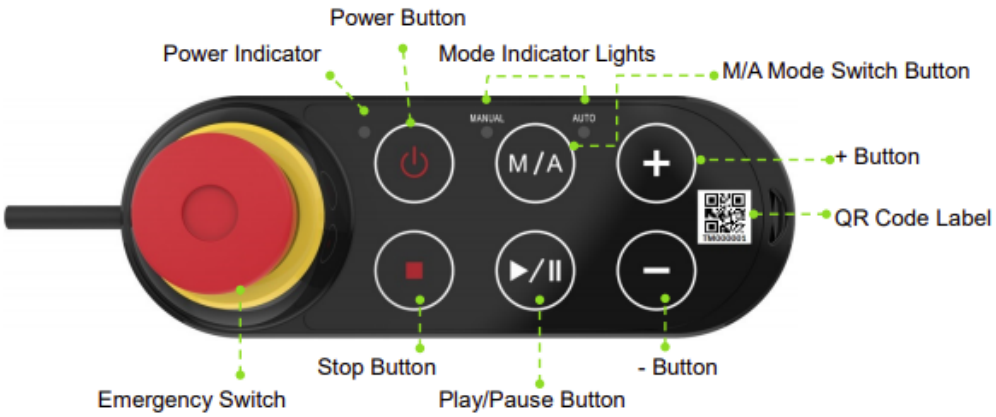
Name	Online value	Modify	Comment	Data type
TM_Robot				OEN\nRobot\nTM\sRobot
Status				OEN\nRobot\nTM\sStatus
IO				OEN\nRobot\nTM\sIO
Button				OEN\nRobot\nTM\sButtons
Light				OEN\nRobot\nTM\eRobotLight
Stick				OEN\nRobot\nTM\sStick
Speed			R (%) Current speed	INT
Mode			R (Man-Auto-Pause) Status	OEN\nRobot\nTM\eMode
Play_Pause			W Same signal as pressing Play/Pause but	BOOL
Stop			W Same signal as pressing Stop button on	BOOL
Stick_Plus			W Same signal as pressing + button on th	BOOL
Stick_Neg			W Same signal as pressing - button on the	BOOL

```
Play_Pause := TM_Robot.Stick.Play_Pause;
```

Figur 2.20: ST kode eksempel for skriving av funksjonen PlayPause, fra Sysmac Studio

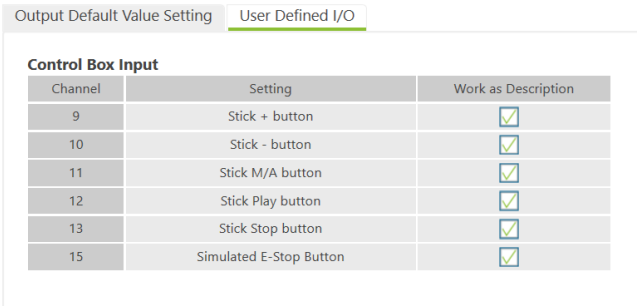
Robot Stick

Figur 2.4 fremstiller robot stick og dens funksjonaliteter



Figur 2.21: Robot stick har funksjonene Av/på, Stop, Manuell/Auto, Play/Pause, programfart opp og programfart ned. Bilde av robotsticken vist i figuren, hentet fra datablad [10]

Grunnet problem med registeradressene til robot stick, har en alternativ løsning blitt utført for å anskaffe robot stickfunksjoner i PLS. Deler av den alternative løsningen har har blitt gjort slik som vist på side 40 i dokumentet "Start!TmCobot" funnet i [18]. Controllerboksen (datamaskinen til selve Roboten) har et digitalt I/O kort "CBoxDI" (Controller Box Digital Input) og "CBoxDO" (Controller Box Digital Output), hvor robot stick funksjonene er forhåndsdefinert til CBoxDI i TMflow som vist i figur 2.22

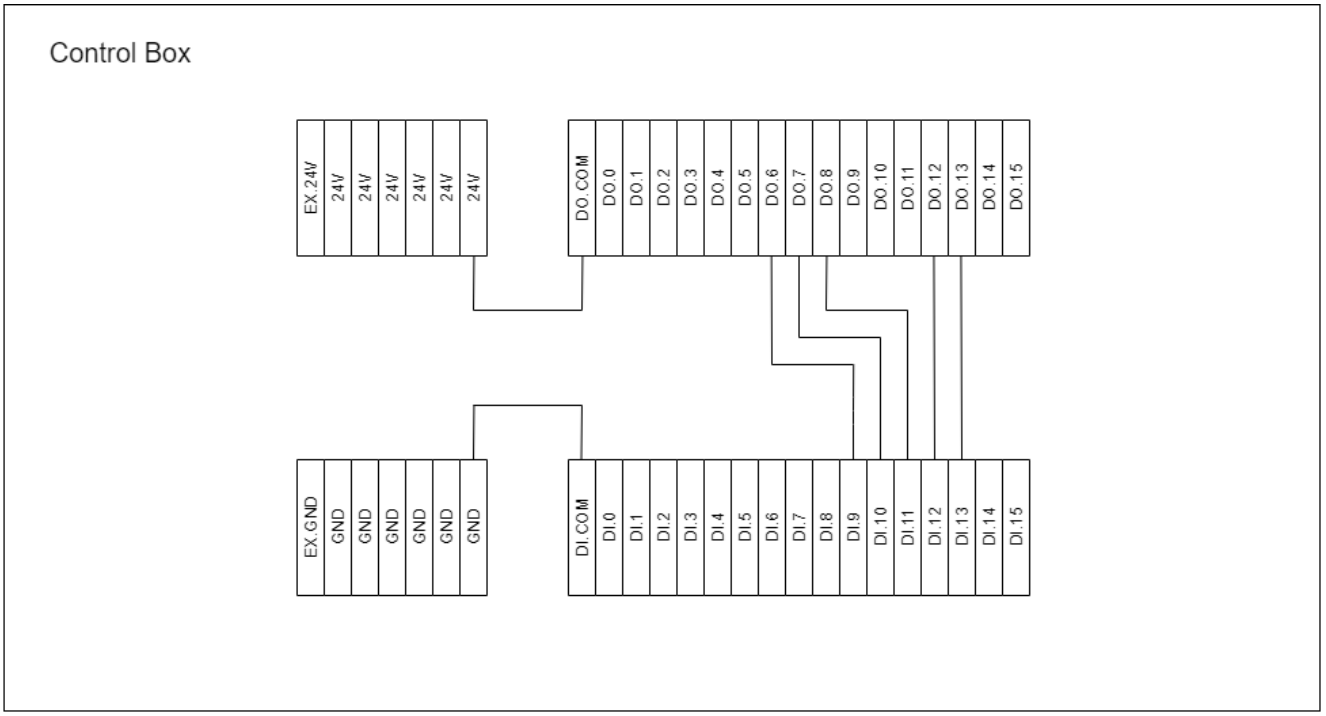

Output Default Value Setting | User Defined I/O

Control Box Input

Channel	Setting	Work as Description
9	Stick + button	☑
10	Stick - button	☑
11	Stick M/A button	☑
12	Stick Play button	☑
13	Stick Stop button	☑
15	Simulated E-Stop Button	☑

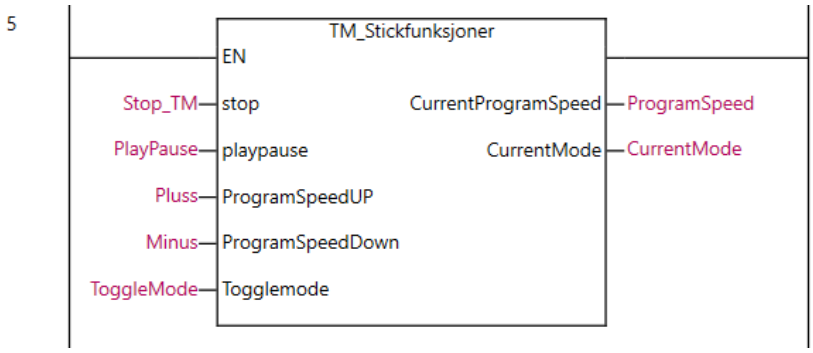
Figur 2.22: Forhandsbestemt Controller Box Inputs

Ved å koble digitale utganger til de digitale inngangene som har forhåndsdefinerte robot stickfunksjoner, får man tilgang til robot stickfunksjonene ved å kommunisere med modbus adressene til I/O kortet. Se figur 2.23



Figur 2.23: Koblingsskjema for Cbox I/O. Grunnet ingen respons fra utgangene CboxDO 9, 10 og 11 brukes 6, 7 og 8 i stedet

Omron har en funksjonsblokk, "TM_Control" [18], for å bruke robot stick funksjoner med å ta i bruk CBoxDO 9, 10 og 11. Grunnet ingen respons fra disse utgangene, så har det blitt programmert en ny funksjonsblokk.



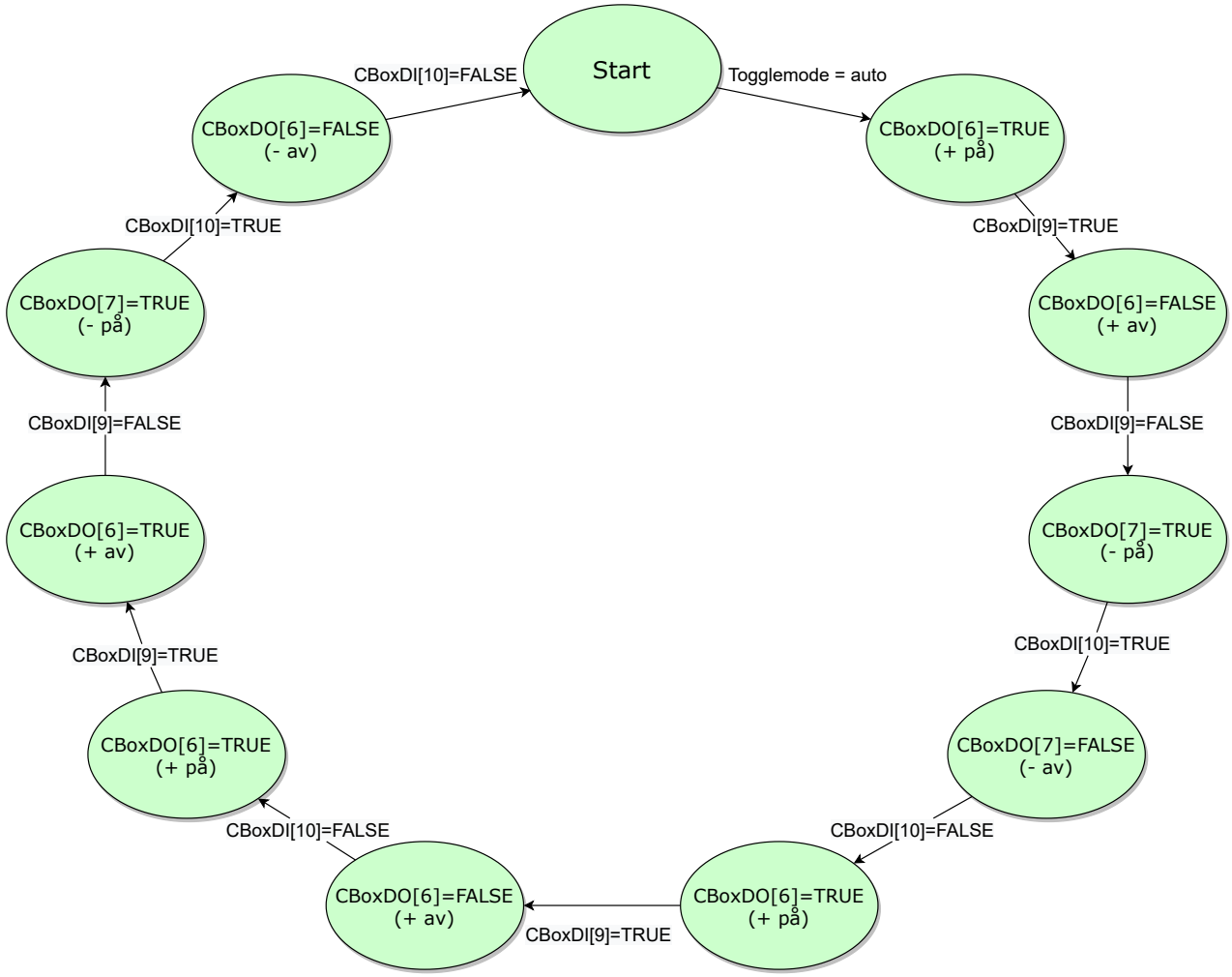
Figur 2.24: Med denne funksjonsblokken så styrer man CBoxDO som vist i 2.23. Med funksjonsblokken kan man bruke funksjonene til robot stick, vist i figur 2.21, utenom "Power Button"

Figur 2.25 viser deler av koden til funksjonsblokken "TM_Stickfunksjoner".

```
4   //INPUTS
5   TM_Robot.IO.CBoxDO[13] := stop;                 // Aktiverer Cbox digital utgang 13 for å stoppe programmet (flanke styrt)
6   TM_Robot.IO.CBoxDO[12] := playpause;            // Aktiverer Cbox digital utgang 12 for å Pause eller Starte programmet (flanke styrt)
7   TM_Robot.IO.CBoxDO[8] := Togglemode;            // Aktiverer Cbox digital utgang 8 for å toggle veksle mellom manuell og auto (flanke styrt)
8
9   IF Autosekvens=FALSE THEN                       // for å forhindre bruker å trykke på + og - under overgang til Auto mode
10     TM_Robot.IO.CBoxDO[7] := ProgramSpeedDOWN;   // Aktiverer Cbox digital utgang 7 for å senke program fart (flanke styrt)
11     TM_Robot.IO.CBoxDO[6] := ProgramSpeedUP;     // Aktiverer Cbox digital utgang 6 for å øke program fart (flanke styrt)
12  END_IF;
13
14  //OUTPUTS
15  CurrentProgramSpeed := TM_Robot.Stick.Speed;
16  CurrentMode := TM_Robot.Stick.Mode;
```

Figur 2.25: Inngangene til funksjonsblokken er knyttet til de Digitale CBox utgangene vist i figur 2.23 og Utgangene er knyttet til adressene for lesing av programfart og nåværende programmodus

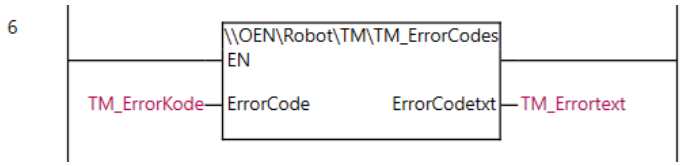
I "TM_Stickfunksjoner" har man mulighet til å sette programmet i "Auto" modus . For å sette TMflow programmet i auto, må man utføre robot stick sekvensen → hold inne knappen M/A til knappene begynner å lyse, deretter trykke $+ - + + -$ på robot stick, som beskrevet på side 8 i "Start!TMCobot" [18]. Koden i funksjonen er en case struktur som styrer CBox utgangene tilknyttet "+" og "−" funksjonene til robot stick, se 2.23 og 2.22. Tilstands diagrammet i figur 2.26 viser hvordan koden automatisk utfører sekvensen for å sette programmet i auto modus.



Figur 2.26: Betingelsene i tilstandsdiagrammet er kun for å verifisere at signalet til de digitale innganene tilknyttet til "+" og "−" er registrert før viderkjøring av koden. Se vedlegg A.9.3 for fullstendig funksjonsblokk kode.

Feilmeldinger fra robot

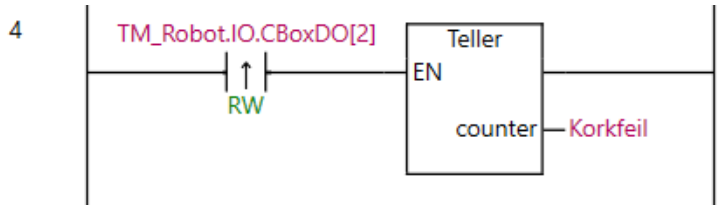
Egen funksjonsblokk fra OMRON blir brukt for lesing av feilmeldinger fra robot gjennom modbus. se figur 2.27 og datablad [12]



Figur 2.27: TM_ErrorCode

Lagring av antall korkmomentfeil i PLS

Hvis korkmomentet ikke blir godkjent i TMflow programmet, settes CBoxDO[2]=True i noden "FeilmeldingPLS" som vist i subflow-noden "IkkeGodkjent" 2.4.2. Dette vil da kunne leses gjennom modbus i Sysmac Studio. Som vist i figur 2.28 vil en funksjonsblokk telle antall korkmoment tester om ikke blir godkjent, og lagre dette inn i en egen global variabel "Korkfeil"



Figur 2.28: Teller for Korkfeil

Resetting av Korkfeil variabel

ST kode for å resette korkfeil variabelen, vist i figur 2.29

```
6  IF Reset THEN          //Resetter tellervariabel for korkfeil
7      Korkfeil := 0;
8  END_IF;
```

Figur 2.29: ST kode for resetting av variabelen korkfeil

Fullstending Korkmomenttest PLS program finner man i vedlegg A.3

2.4.4 Tillegg til korkmoment test sekvensen

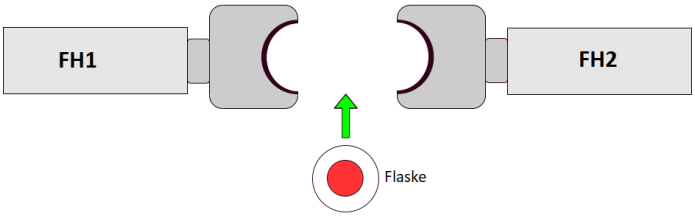
Verktøy

Med nåværende verktøy på roboten vil ikke roboten være i stand til å teste korkmomentet grunnet for lite friksjon mellom griper og kork. Dette medførte at griperen ville skli på korken under moment testing. Et nytt verktøy egnet for denne type jobb bør erstatte nåværende verktøy. Et bra alternativ er "maskinert chucks", vist i figur 2.30 hentet fra [1]. Disse kan tilpasses til korkens ytre støypte mønster og dimensjoner, som resulterer i ekstra bra grep.

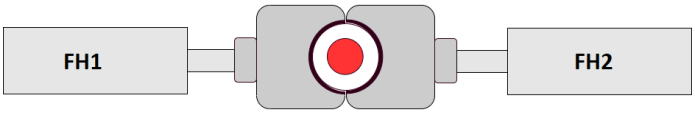
Figur 2.30: Verktøy for korkmoment test

Sylinderstyrte flaskeholdere for fastspenning av flaske

For at selve flasken ikke skal rotere under korkmoment testingen, må det installeres et system for fastspenning av flaske. Et alternativ kan være to sylinderstyrte flaskeholdere. Ett slikt eksempel er presentert i figur 2.31a og 2.31b. Figur 2.31a illustrer startposisjon med innkommende flaske og 2.31b illustrerer sluttposisjon med fastspenning av flaske.



(a) Startposisjon Hlaskeholdere

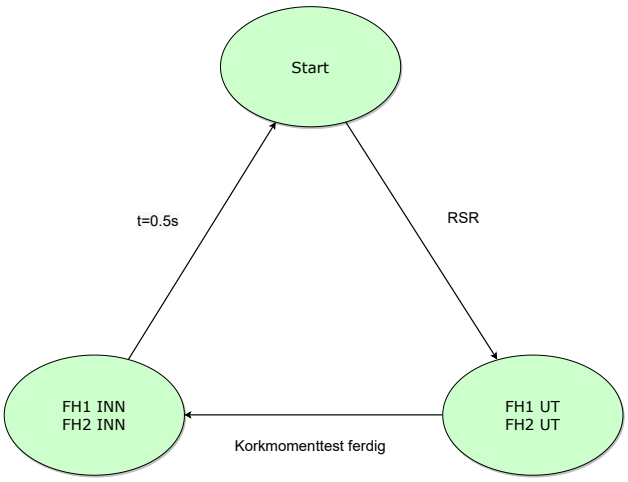


(b) Sluttposisjons Flaskeholdere

Figur 2.31

Tilstandsdiagram for flaskeholdersekvensen

Dette programmet bør programmeres i TMflow, men grunnet begrenset tid med roboten så har programmet for flaskeholdersekvensen ikke blitt implementert i selve TMflow programmet. Figur 2.32 viser tilstandsdiagrammet for tenkt sekvens.

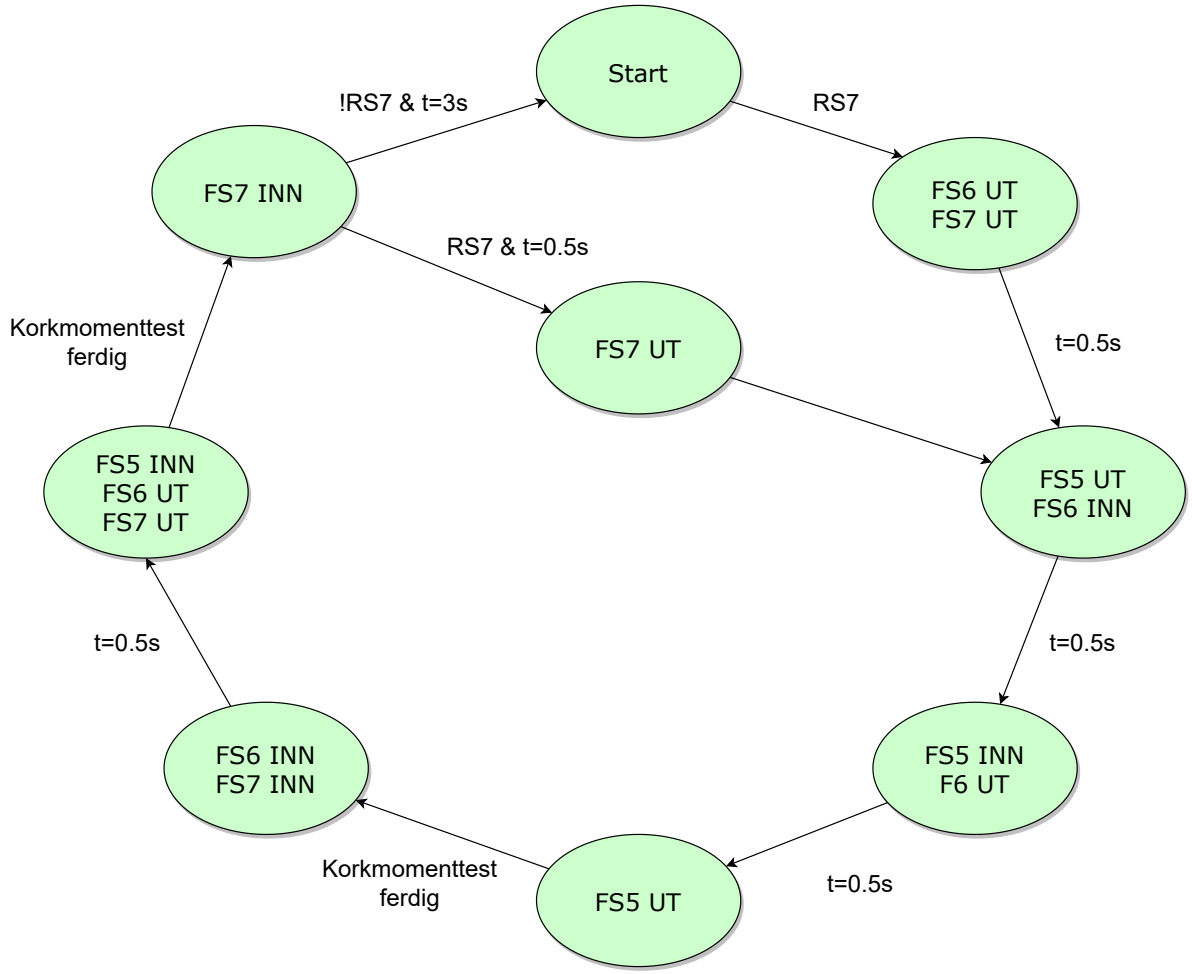


Figur 2.32: RSR er en reflektiv sensor 2.8 som sjekker om flaske er i posisjon, betingelsen "Korkmomenttest ferdig" sjekker om korkmomentesten er ferdig og t=0.5s er for å simulere tiden holdesylinderene bruker inn til startposisjon.

Flaskeregulerings sekvens

For å kunne regulere mating av flasker til selve korkmomenttest sekvensen har det blitt implementert en egen separert flaskeregulerings sekvens. Denne sekvensen er styrt av tre sylindere og en reflektiv sensor(RS7) som er programmert til å kun slippe en og en flaske til korkmoment testing seksjonen. Den tenkte plasseringen til sylinderene samt sensoren er fremstillet i innledende prosesstegning 2.5 til denne seksjonen.

Tenkt tilstandsdiagram for flaskeregulerings sekvensen er presentert i figur 2.33



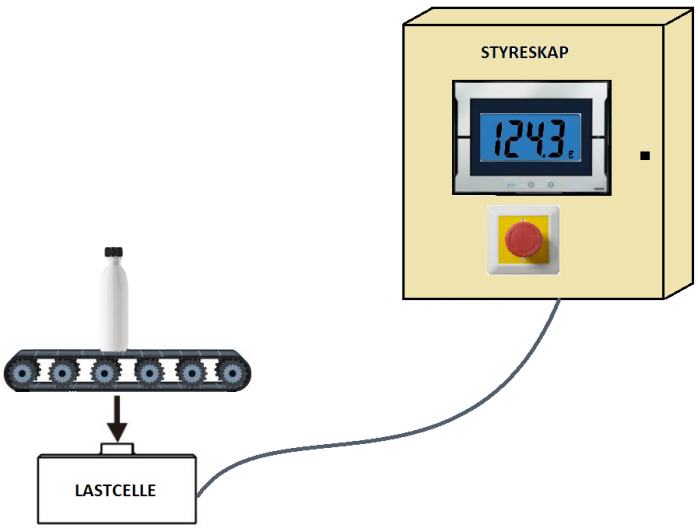
Figur 2.33: RS7 betingelsen mellom "Start" og neste tilstand, er for å detektere om flaske har ankommet. t=0.5s betingelsene er simuleringstid for sylinderene. Når programmet er i siste tilstand "FS7 INN" sjekker programmet om det er ny flaske som har ankommet. Betingelsen "!RS7 & t=3s" hvis ingen flaske er detektert ved RS7 i tre sammenhengende sekunder, da vil programmet gå tilbake til start og "RS7 & t=0.5s" hvis flaske er detektert ved RS7, da vil programmet fortsette.

2.5 Verifisering av innholdsmengde i flaske med bruk av lastcelle (Veiesekvens)

2.5.1 Innledning

Lastcelle er et mulig tillegg til prosessen for å verifisere innholdsmengden av flasker. Her er det tenkt at flaskene blir veiet uten stans på transportsbåndet, altså med bruk av dynamisk veiing. For å oppnå et brukbart målesignal for selve veiesekvensen må man behandle det analoge spenningssignalet fra lastcellen. Det vil bli presentert eksempel på digital filtrering for å redusere støy og deretter kalibrering (skalering) for å oppnå et presist måleresultat. Det vil bli brukt noen sider på fremgangsmåten for å måle innholdsmengden ved dynamisk veiing og deretter presentert eksempel på PLS program for verifisering av innholdsmengde samt lagring av antall innholdsmengdefeil i både antall og antall gitt i %.

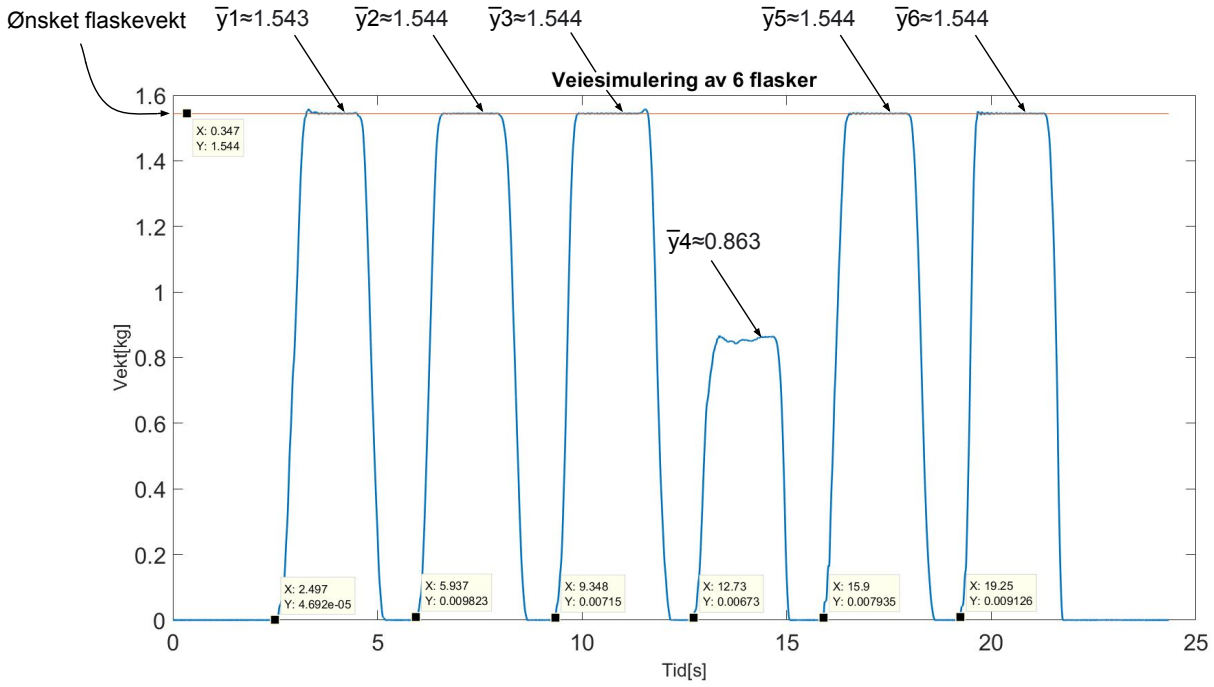
Figur 2.34 viser den tenkte oppsettet av veiesekvensen.



Figur 2.34: Lastcellen veiier passerende flasker og PLS gjør vurderinger av innholdmengden til flaskene basert på tilført målesignal.

Resultat fra labsimuleringer

Labsimulering av passerende flasker over lastcelle er gjort for hånd med gradvis pålegging og gradvis fjerning av metallklosser på lastcelle. Resultat for simulering av seks passerende flasker hvor en flaske har innholdsmengde utenfor toleranseverdiene er vist i figur 2.35. Det er antatt at toleranseverdien for innholdsmengden er $[+2,-2]\%$ av ønsket flaskenivå.



Figur 2.35: y[1-6] indikerer den loggførte gjennomsnittsvekten for hver respektive flaske under simuleringen. Resultatet viser at y4 avviker fra ønsket innholdsmengde som betyr at tilhørende dyse har tilført feil væskemengde. I dette tilfellet avviker væskemengden fra dyse nummer tre, dette vil da bli lagret i variabler som kan bli monitorert på HMI.

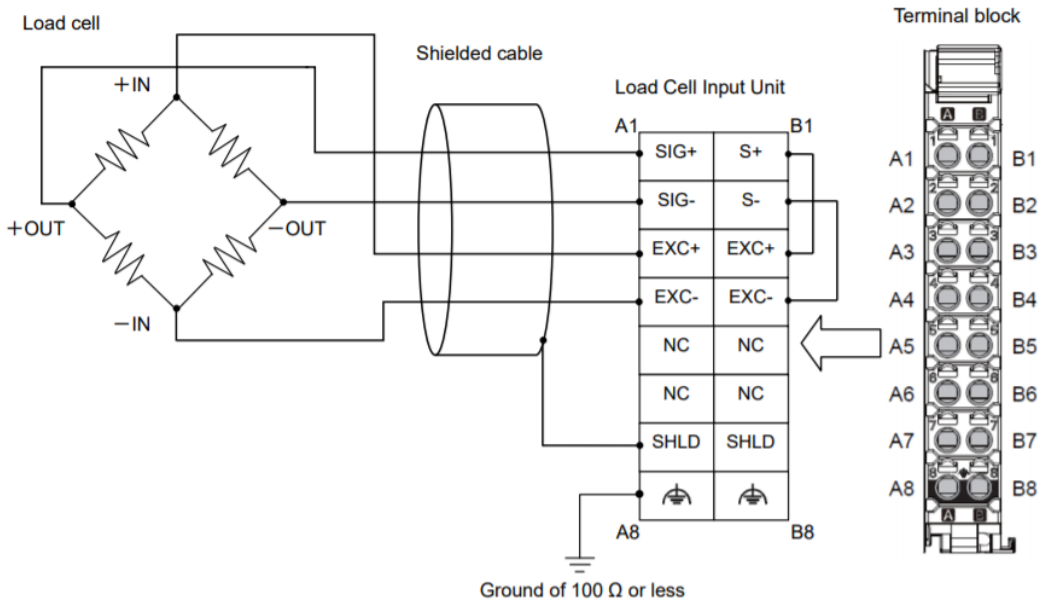
Som vist i figur 2.35 kan man se at flaske nummer tre avviker fra ønsket nivå. Dette vil bli lagret i feilvariablene som vist i figur 2.36, dette blir også blir fremstillet på HMI-en vist i 2.69.

```
60  Flaskenivåfeil := Flaskenivåfeil + 1;
61  flaskenivåfeil_prosent := (Flaskenivåfeil /antall_flasker_veiet )*100;
62
63  IF dysenummer = 1 THEN (*konfigurerer flaskenivåfeilen til respektiv fylledyse*)
64    nivåfeil_dysnummer6 := nivåfeil_dysnummer6 + 1;
65    nivåfeil_dysenummer6_prosent := (nivåfeil_dysnummer6 /dysenummer6_antall_fyllinger )*100;
66  ELSIF dysenummer = 2 THEN
67    nivåfeil_dysnummer5 := nivåfeil_dysnummer5 + 1;
68    nivåfeil_dysenummer5_prosent := (nivåfeil_dysnummer5 /dysenummer5_antall_fyllinger )*100;
69  ELSIF dysenummer = 3 THEN
70    nivåfeil_dysnummer4 := nivåfeil_dysnummer4 + 1 ;
71    nivåfeil_dysenummer4_prosent := (nivåfeil_dysnummer4 /dysenummer4_antall_fyllinger )*100;
72  ELSIF dysenummer = 4 THEN
73    nivåfeil_dysnummer3 := nivåfeil_dysnummer3 + 1;
74    nivåfeil_dysenummer3_prosent := (nivåfeil_dysnummer3 /dysenummer3_antall_fyllinger )*100;
75  ELSIF dysenummer = 5 THEN
76    nivåfeil_dysnummer2 := nivåfeil_dysnummer2 + 1;
77    nivåfeil_dysenummer2_prosent := (nivåfeil_dysnummer2 /dysenummer2_antall_fyllinger )*100;
78  ELSIF dysenummer = 6 THEN
79    nivåfeil_dysnummer1 := nivåfeil_dysnummer1 + 1;
80    nivåfeil_dysenummer1_prosent := (nivåfeil_dysnummer1 /dysenummer1_antall_fyllinger )*100;
81  END_IF;
```

Figur 2.36: Øverste røde pil viser at koden har registrert en innholdsmengdefeil samt hvor mange prosent total innholdsmengdefeil for de seks flaskene under simulering. Nederste røde pil viser hvilken dyse som skaper avvik samt antall og antall prosent feil per flaske denne har fyllet.

RS1201

RS1201 [7] er en terminalblokk tilkoblet NX102, som brukes for å konvertere det analoge spenningssignalet fra lastcellen om til et digitalt signal som kan prosesseres i Sysmac Studio. RS1201 er designet for små målesignal typisk gitt fra lastceller som operer i millivolt området. Ved bruk av RS1201 så slipper man å koble inn forsterkere for å skalere opp spenningssignalet fra lastcellen for innlesning av normale analoge inngangskort. Oppkobling av lastcellen til RS1201 er vist i figur 2.37 hentet fra datablad [6]



Figur 2.37: Eksempel på tilkobling mellom RS1201 og lastcelle. Hentet fra datablad [6]

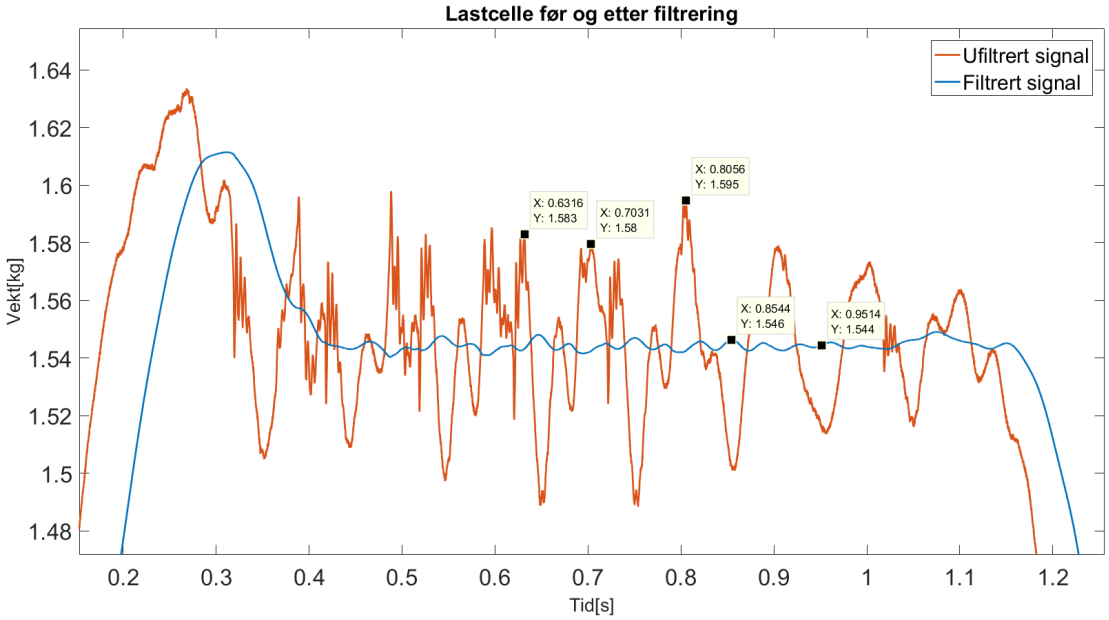
2.5.2 Filter dimensjonering og kalibrering

Filter dimensjonering

RS1201 har et innebygget lavpass FIR filter som bruker "Moving average" prinsippet [39] for filtrering av støy. Fordelen med å bruke det innebygde FIR filteret er at RS1201 har en oppdateringsfrekvens på 8kHz ([7], s.8-9) i motseting til NX102-1000 som har en oppdateringsfrekvens på 1kHz ([14], s.1-8), dette gir muligheten for å filtrere målesignalet bedre. Datablad [9] viser til en egen filtreringsprosedyre for å dimensjonering av "Moving average" filteret i RS1201, som blir tatt i bruk under labtesting.

Resultat av filtrering

Figur 2.38 viser signalresponsen før og etter filtrering ved simulering av en flaske som beveger seg over lastcellen.



Figur 2.38: Matlabplot av resultat for digital filtrering. Dette plottet viser hvordan signalet ser ut ved pålegging av vekt på lastcellen før filtrering og etter filtrering. Som vist i figuren, så vil innsvingninger og generelt støy ved pålegging av vekt skape et upresist måleresultat.

Kalibrering

Sysmac Studio har en innebygget kalibreringsfunksjon for RS1201 hvor fremgangsmetoden for kalibreringen blir fremstilt i datablad [6]. Kalibrering med bruk av den innebygde funksjonen i Sysmac Studio krever at man må koble seg direkte opp mot PLS via ethernet med PC, noe som kan være slitsomt hvis rekalibrering ofte er nødvendig. Derfor har det blitt implementert en egen kalibreringsprosess med bruk av funksjonsblokker. Dette gir mulighet for rask og brukervennlig kalibrering gjennom HMI. Skaleringsblokken A.9.2 brukt i kalibreringsprogrammet A.8 inneholder den lineære matematiske funksjonen vist i likning 2.1.

$$y(t) = a(x(t) - b) \tag{2.1}$$

y - Skalert vekt
a - Skaleringsfaktor
x - Uskalert vekt inn
b - Forskyvning fra nullvekt

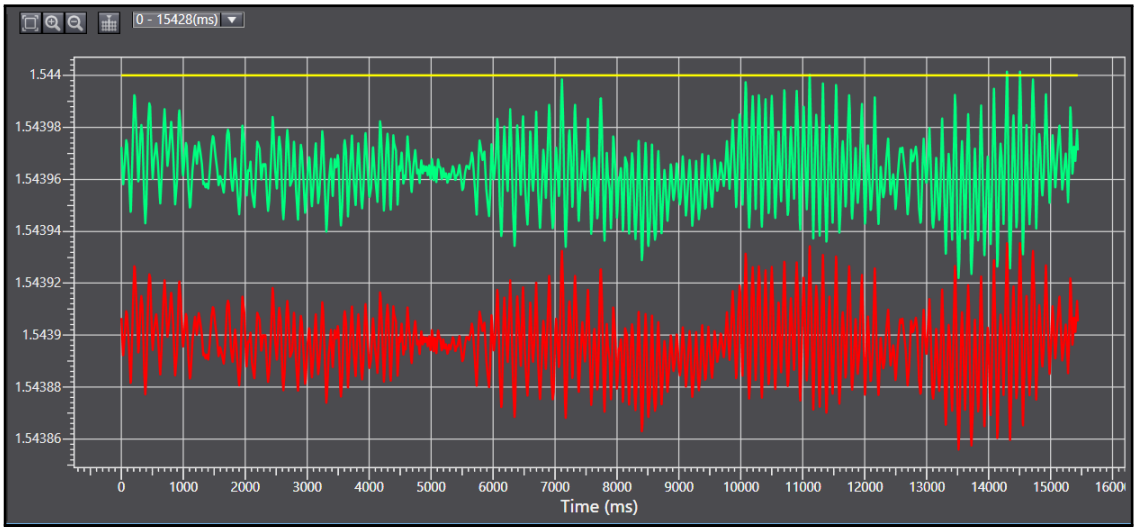
$$a = \frac{y_1 - y_0}{x_1 - x_0} \tag{2.2}$$

y_1 - Kalibreringsobjektets maksimumsvekt
y_0 - Kalibreringsobjektets minimumsvekt (0kg)
x_1 - Målt uskalert kalibreringsobjekt maksimumsvekt
x_0 - Målt uskalert kalibreringsobjekt minumumsvekt

Det nye kalibreringsprogrammet er tilknyttet HMI-skjermene som blir fremstillet i seksjon 2.8.5 hvor man blir tatt stegvis gjennom den nye kalibreringsprosessen.

Resultat av ny kalibreringsmetode

Figur 2.39 viser at kalibrert signal fra innebygget kalibreringsfunksjon i RS1201 og selvlaget kalibreringsfunksjon.



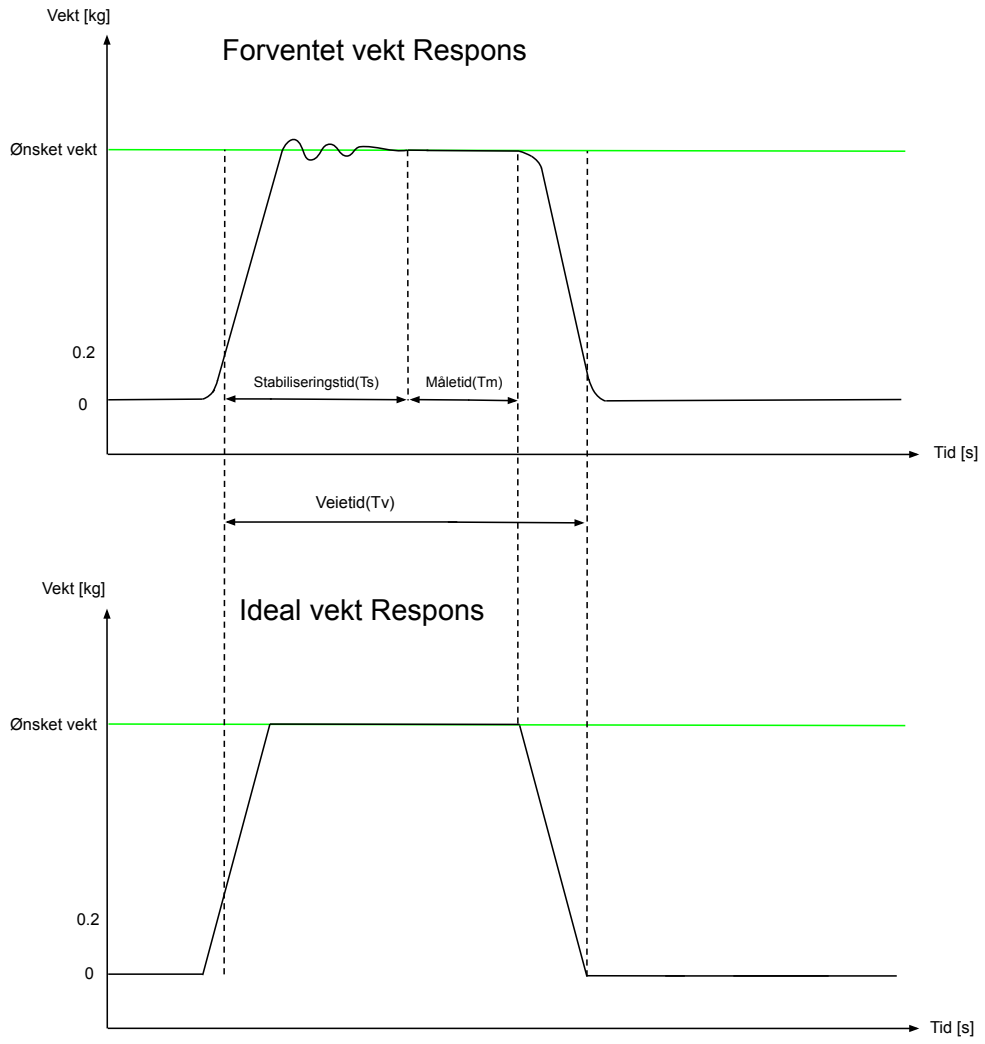
Figur 2.39: Resultatet for den selvlaget kalibreringsprosessen (grønn), Sysmac Studio sin kalibreringsprosess (rød) og ønsket verdi (gul). Grafen fremstiller y-aksen i Kg

Som vist i figur 2.39 avviker begge målesignalene ønsket verdi med mindre en 0.1 gram. Dette er innafor alle toleranser med stor margin. Med at det nye kalibreringsprogrammet gir et litt mer nøyaktig måleresultat en den innebyggde kalibreringsmetoden, kan man konkludere at egen kalibreringsprosess kan brukes.

2.5.3 Dynamisk vektmålingsprinsipp

Teoretisk prinsipp for dynamisk vektmåling

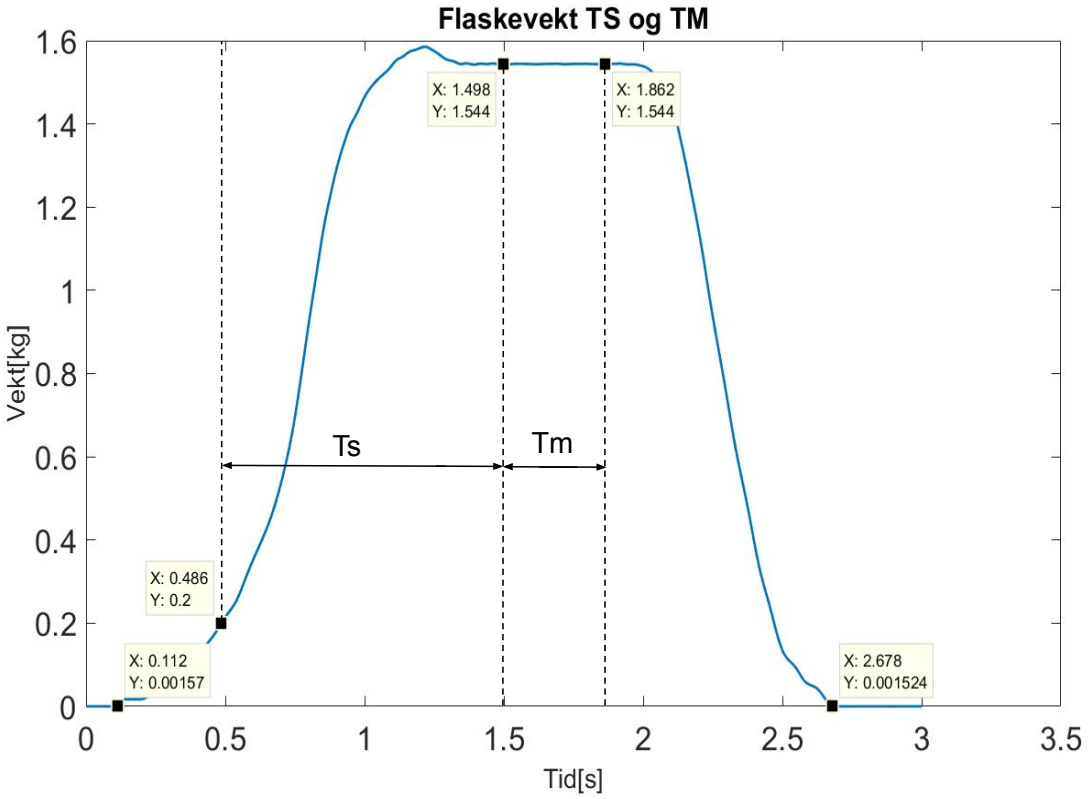
Gitt at veiesekvensen er tenkt plassert etter korkmoment-test sekvensen, blir det antatt at det vil oppstå et mellomrom på noen sekunder mellom hver flaske. For prinsippets skyld brukes det her tre sekunders mellomrom. Da blir PLS programmet programert med hensyn på at veieperioden er 2.5 sekunder per flaske slik det vil være 0.5 sekunders feilmargin mellom flaskene. For å kunne hente ut relativt presise verdier i henhold til flaskens vekt under selve målingen må man ta hensyn til påstigningstid, innsvingningstid og avstigningstid. Figur 2.40 viser en teoretisk fremstilling av forventet vekt respons.



Figur 2.40: Innsamling av vektverdier gjort i intervallet mellom stabiliseringstiden og avstigningen (Tm). Påstigningstid og innsvingningstid blir samlet og referert til som stabiliseringstid (Ts). Selve veiesekvensen vil ikke starte før lastcellen detekterer en vekt på 0.2kg, så da vil stabiliseringstiden basere seg i henhold til den betingelsen.

Labtesting for å finne Ts og Tm

Labtesting ble gjort for å finne stabiliseringstid og måletid ved å gradvis legge vekten på lastcelle for å simulere flaskens stabiliseringstid samt gradvis fjerning av vekten for simulering av avstigningstid. Resultat er fremstillet i figur 2.41



Figur 2.41: Matlab plot flaskevekt analyse for Ts og Tm

Som vist i figur 2.41, har man:
Ts $\approx 1.00s$
Tm=357ms $\approx 0.36s$

For mest mulig nøyaktig måleresultat, tar man gjennomsnittsverdien av alle målingene i perioden Tm. Matematisk fremstilling av gjennomsnittsmålingen vist i likning 2.3

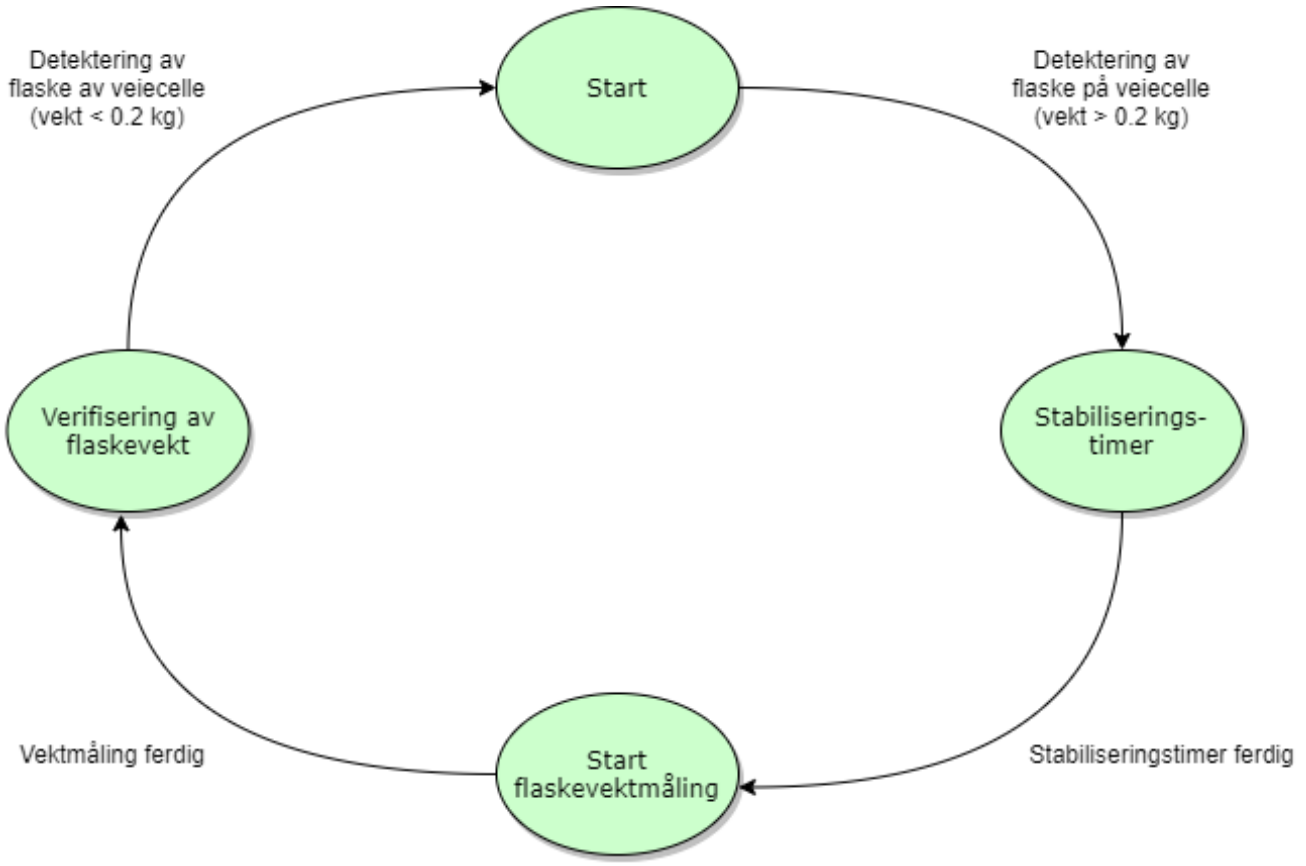
$$\overline{y} = \frac{1}{N} \sum_{n=1}^{N} x_i \tag{2.3}$$

hvor:
$N = Tm[s] \cdot 1000$(Oppdateringsfrekvens for NX102)
$\overline{y}$ er gjennomsnittet av vektmålingene
x_i er diskrete input målinger

2.5.4 PLS program

Tilstandsdiagram

Veiesekvensen er tenkt som vist i tilstandsdiagrammet i figur 2.42.



Figur 2.42: Betingelsene for å starte og stoppe en gjennomkjøring av sekvensen baserer seg på deteksjon av vekt på lastcellen. Hvis lastcellen detektereren vekt på over 0.2kg vil veiesekvensen starte, og hvis lastcellen detekterer at vekten deretter understiger 0.2kg vil sekvensen avsluttes hvis programmet er i siste tilstand for veiesekvensen. Se vedlegg A.4 for PLS-koden til denne sekvensen

I tilstanden "verifisering av flaskevekt" vurderes resultatet fra målingen opp mot vektintervallet [-2, +2]% av ønsket flaskevekt. Hvis flaskevekten ikke befinner seg i dette intervallet, vil programmet lagre vektfeil-sinformasjon i to forskjellige variabler:

1. Totalt antall flaskenivåfeil:
Som vist i figur 2.43 så vil det lagres både antall og antall i prosent av nivåfeil for totalt antall flasker veiet.

```
47          Flaskenivåfeil := Flaskenivåfeil + 1;
48          flaskenivåfeil_prosent := (Flaskenivåfeil/antall_flasker_veiet)*100;
```

Figur 2.43: Kodeutsnitt fra PLS-program som viser variabler for totalt flaskenivåfeil i antall og prosent

2. Hvilken fylledyse som har fyllet feil nivå:
Fyllesekvensen har totalt seks dyser som brukes for flaskefylling. Koden for veiesekvens har en variabel som holder styr på hvilken dyse som tilhører hvilken flaske under fylling som vist i figur 2.44.

```
14      IF dysenummer = 1 THEN (*Holder styr på hvilken fylledyse som tilhører hvilken flaske*)
15          dysenummer := 6;
16      ELSE
17          dysenummer := dysenummer - 1;
18      END_IF;
```

Figur 2.44: Initialverdien til variabelen "dysenummer" er seks, og vil deretter dekrementere for hver gang koden gjennomkjøres. Utgangspunktet for nummerering av dysene er at man teller dysene fra høyre til venstre på prosesstegningen 2.1, der nummer 1 vil være dysen lengst til høyre.

Med desse variabelene lagrer man informasjon om hvilken dyse som skaper avvik når koden detekterer flaskenivåfeil som vist i figur 2.45. Dette gjør det lettere å feilsøke og skille ut hvilken dyse som skaper problemer med flaskenivå under produksjon.

```
48      IF dysenummer = 1 THEN (*konfigurerer flaskenivåfeilen til respektiv fylledyse*)
49          nivåfeil_dysnummer6 := nivåfeil_dysnummer6 + 1;
50          nivåfeil_dysenummer6_prosent := (nivåfeil_dysnummer6/antall_flasker_veiet)*100;
51      ELSIF dysenummer = 2 THEN
52          nivåfeil_dysnummer5 := nivåfeil_dysnummer5 + 1;
53          nivåfeil_dysenummer5_prosent := (nivåfeil_dysnummer5/antall_flasker_veiet)*100;
54      ELSIF dysenummer = 3 THEN
55          nivåfeil_dysnummer4 := nivåfeil_dysnummer4 + 1 ;
56          nivåfeil_dysenummer4_prosent := (nivåfeil_dysnummer4/antall_flasker_veiet)*100;
57      ELSIF dysenummer = 4 THEN
58          nivåfeil_dysnummer3 := nivåfeil_dysnummer3 + 1;
59          nivåfeil_dysenummer3_prosent := (nivåfeil_dysnummer3/antall_flasker_veiet)*100;
60      ELSIF dysenummer = 5 THEN
61          nivåfeil_dysnummer2 := nivåfeil_dysnummer2 + 1;
62          nivåfeil_dysenummer2_prosent := (nivåfeil_dysnummer2/antall_flasker_veiet)*100;
63      ELSIF dysenummer = 6 THEN
64          nivåfeil_dysnummer1 := nivåfeil_dysnummer1 + 1;
65          nivåfeil_dysenummer1_prosent := (nivåfeil_dysnummer1/antall_flasker_veiet)*100;
66      END_IF;
```

Figur 2.45: Kode for dysefeil

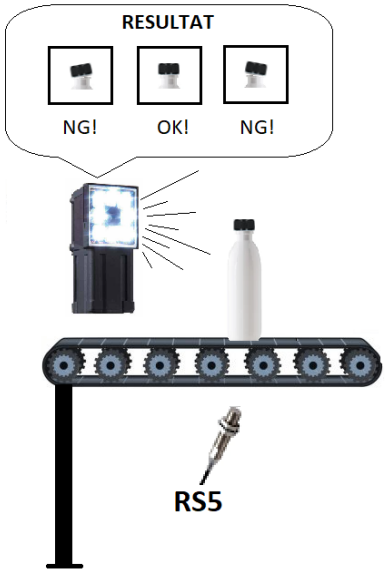
2.6 Bildeanalyse av kork med bruk av smartkamera

2.6.1 Innledning

I Prosessen ved NOS er det allerede to smartkameraer av typen FQ2-S installert, men vi har fått utdelt et smartkamera for labtesting. Dette kapittelet vil da fremstille hvilken inspeksjonfunksjon man kan bruke for detektering av skjev eller mangel på flaskekork gjennom bildeanalyse med FQ2-S smartkamera. Det vil også bli presentert hvordan man kan etablere kommunikasjon mellom PLS og FQ2-S for tilbakemeldinger av flaskekorkens tilstand.

Smartkameraet har to hoved oppgaver, analysere om flasken mangler kork eller om korken er skjevt påskrudd. Kameraet skal gi tilbakemedling til PLSen basert på bildeanalysen for videre vurdering. For å utføre denne oppgaven trengs det også en reflektiv sensor(FS5) for å styre triggersignalet til kameraet. Når flasken passerer sensoren så vil smartkameraet ta et bilde, deretter analysere bilde og sender bildevurderingen over ethernet til PLSen.

Figur 2.46 viser tenkt oppsett for bildeanalyse med FQ2 smartkamera.



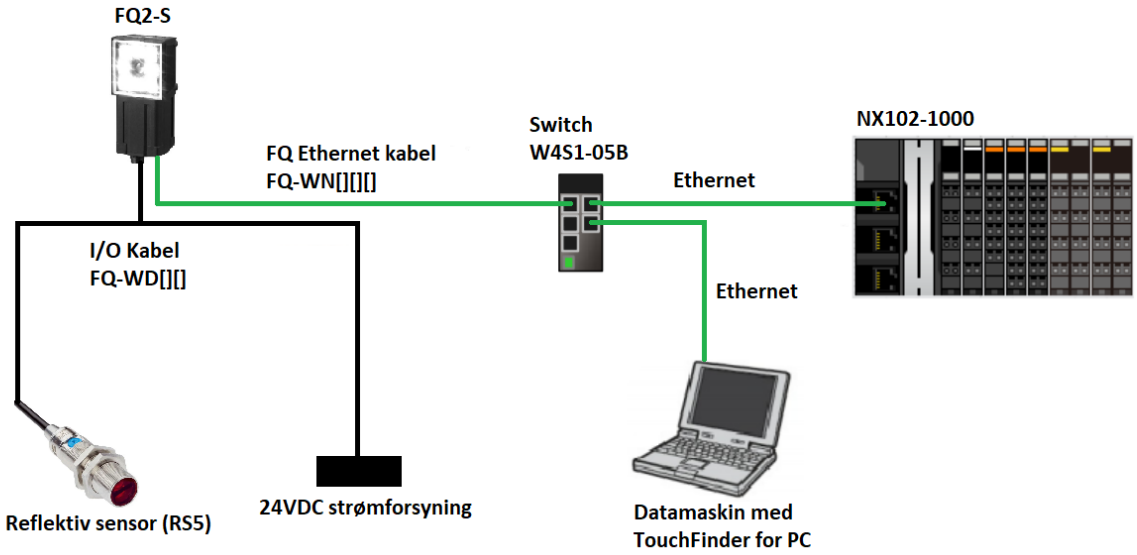
Figur 2.46: Når RS5 registrerer flaske så tar smartkamera bilde av flaskekork

Et eget oppsett for simulering av bildeanalyse av flasker i bevegelse ble brukt for verifisering av kodens funksjonalitet. Fastspenning av smartkameraet i ei stikke, fastspent endbryter for å utløse triggersignal og en "vegg" som referanseavstand for flaskene ved passering. Resultatet av simulering for vanlig, skjev og mangel på kork finner man på videoformat i denne linken:

`https://youtu.be/cAYFE2o8z5s` fra 0:00-1:09

2.6.2 Grensesnitt

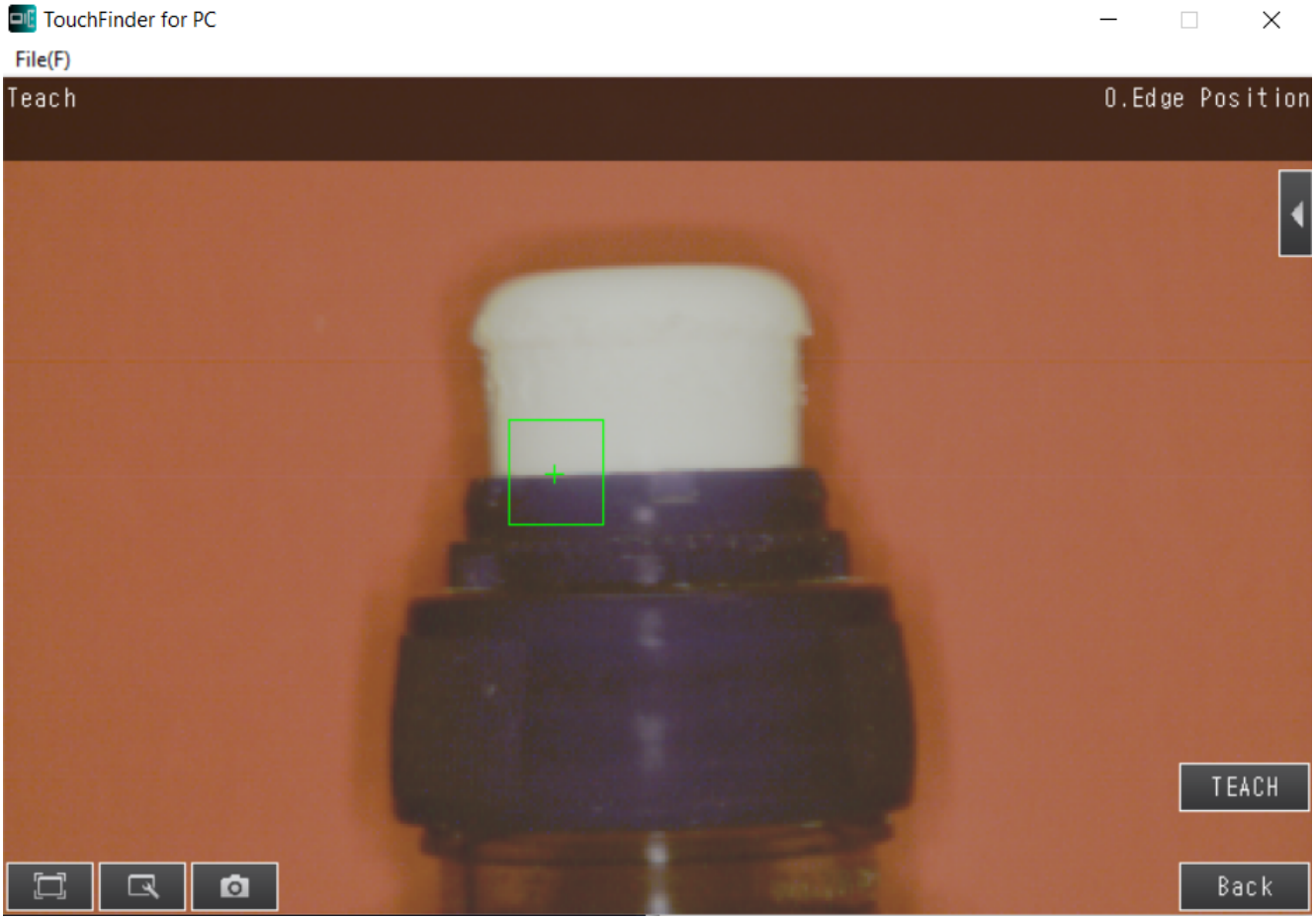
Figur 2.47 viser relevant grensesnitt med FQ2.



Figur 2.47: Den tenkte oppkoblingen for bildeanalyse sekvensen. Under labtesting så ble den reflektive sensoren substituert med en mekanisk endebryter grunnet mangel på sensoren. Ethernet blir brukt for kommunikasjon mellom Smartkamera og PLS, I/O kabel for strømtilførsel og trigger.

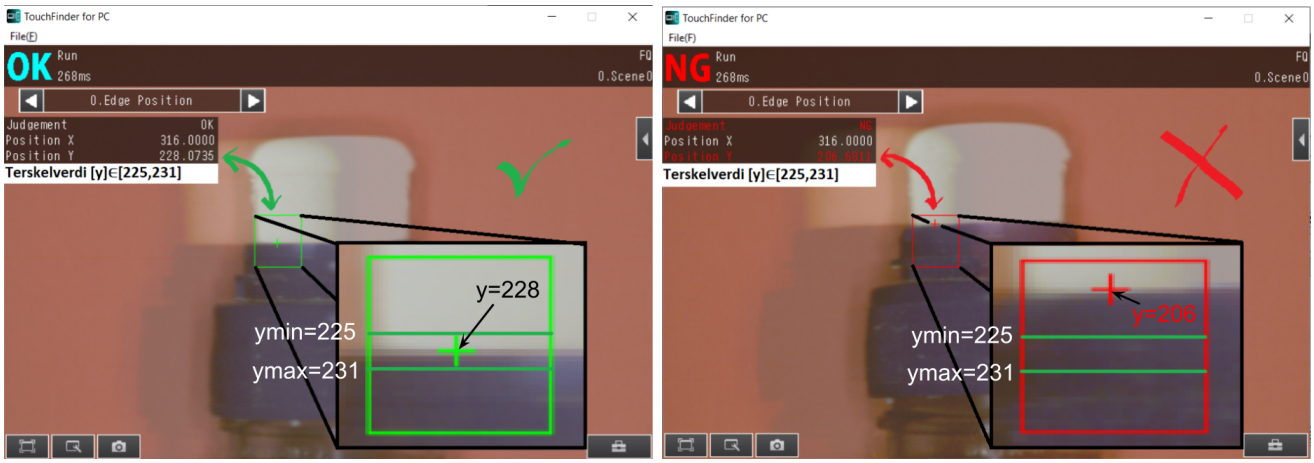
2.6.3 Bildeanalyse

For å verifisere korkplasseringen blir "Touchfinder for PC" brukt for å lære kameraet til å bruke inspeksjonsfunksjonen "Edge position". Denne funksjonen baserer seg på detektering av posisjonen til kanter(fargeoverganger). Eksempel på en kant i dette tilfellet er vist i figur 2.48 hvor smartkameraet har blitt lært til å se etter fargeovergangen blått til hvitt (grønt kryss) innafor egendefinert søkeområdet (grønn boks).



Figur 2.48: Kantposisjon kork

Man definerer terskelverdigrensene for godkjenning av kantposisjonen innenfor søkeområdet. I dette tilfellet blir kun terskelverdien for kantposisjonen i y-aksen begrenset. Dette er for at feilposisjonert kork har vist seg lettest å detektere i loddrett retning. Eksempel på godkjent (OK) og ikke godkjent (NG) kant innenfor terskelområdet (grønne strekene) $[y_{min}, y_{max}]$ vist i figur 2.49a og 2.49b. Verdiene y, ymin og ymax i figurene er fremstillet i pixelkoordinaten for y-aksen, hvor y-aksen teller fra top til bunn på bildet. Grunnet at korkbilder blir tatt i bevegelse, blir bildene litt uklare. Dette er for å simulere hvordan det ville sett ut i praksis. Kryss indikerer detektert kant.

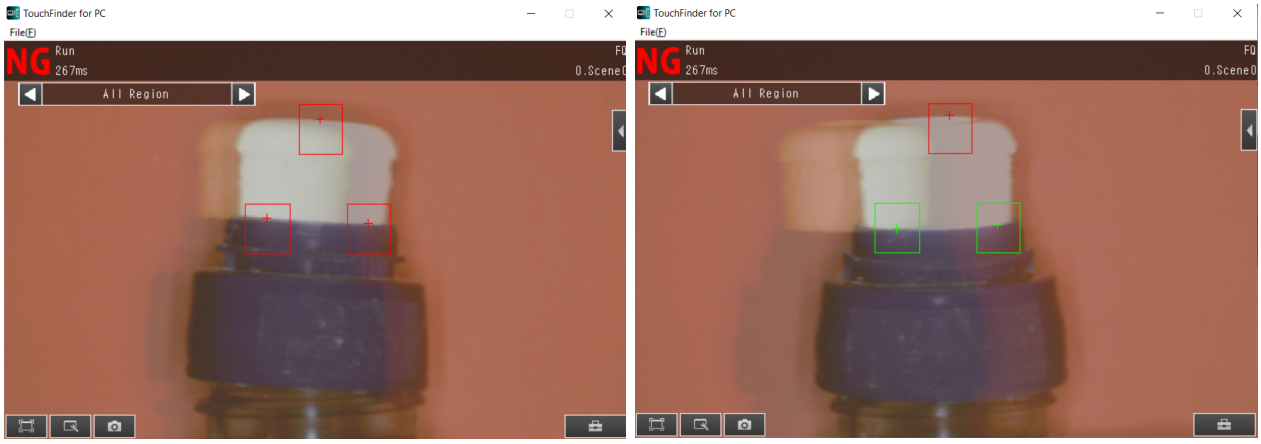


(a) Eksempel på kant innenfor terskelverdi

(b) Eksempel på kant utenfor terskelverdi

Figur 2.49

Grunnet at flasker kan ende opp med skjev kork i alle retninger, er det nødvendig med flere inspeksjonsfunksjoner for mer presis vurdering av korkposisjonen. En skeiv kork kan godkjennes fra en bestemt vinkel hvis man kun analyserer en kantposisjon, derfor blir det implementert tre inspektsjonsfunksjoner av typen "Edge Position" for å forhindre alle falske positive utfall. Se figur 2.50a for sidebilde av skeiv kork og figur 2.50b for frontbilde av skeiv kork.



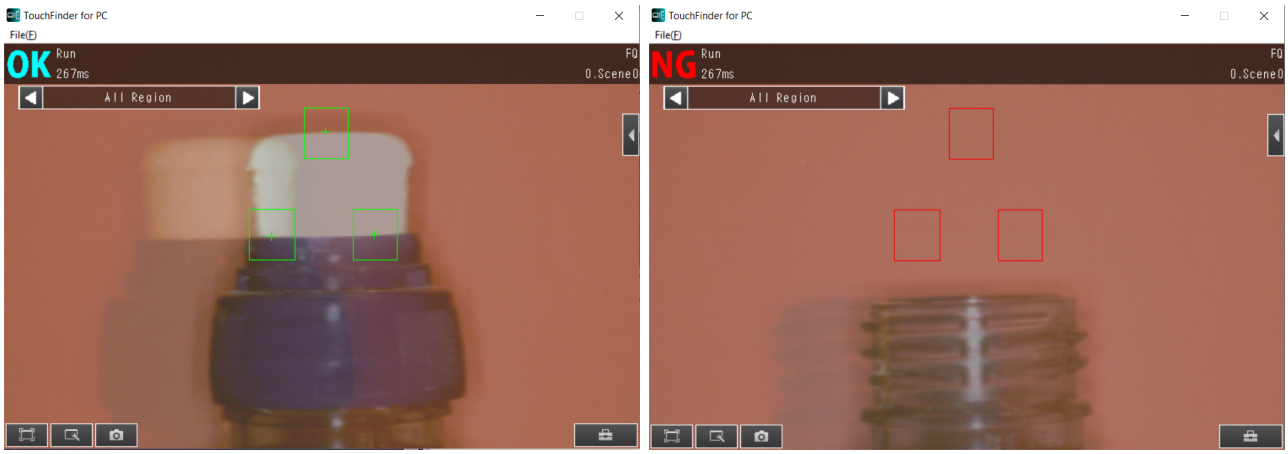
(a) Sidebilde skeiv kork

(b) Frontbilde av skeiv kork

Figur 2.50

Som vist i figur 2.50b vil inspeksjonsfunksjone nederst detektere kantposisjoner innafor terskelverdiene til en godkjent flaske, men øverste inspeksjonsfunksjon detekterer en kantposisjon som overstiger definert høyde til en godkjent flaske som indikerer skjev flaskekork.

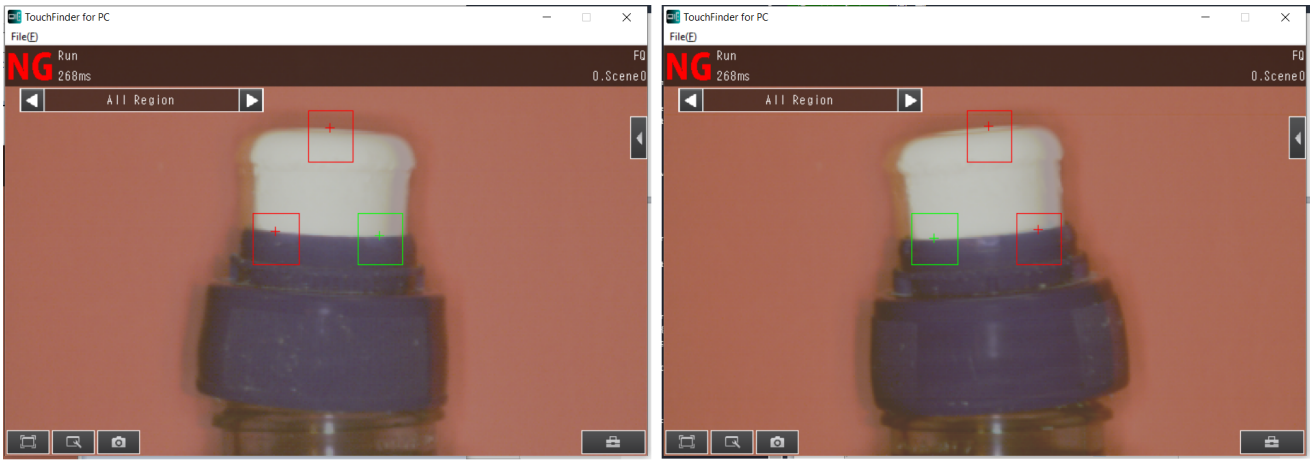
Når inspeksjonsfunksjonene ikke detekterer noen kanter innenfor søkeområdet, vil det registreres at korken ikke er godkjent. Se figur 2.51a for godkjent flaske i figur 2.51b og flaske uten kork i figur 2.51b.



(a) Godkjent kork **(b)** Flaske uten kork

Figur 2.51

Hvis det oppstår en situasjon der den øverste inskepsjonsfunksjonen godkjennes på en skeiv kork, vil inspeksjonsfunksjon nummer to eliminere falsk positiv godkjenning av korkplasseringen vist i figur 2.52a og 2.52b.

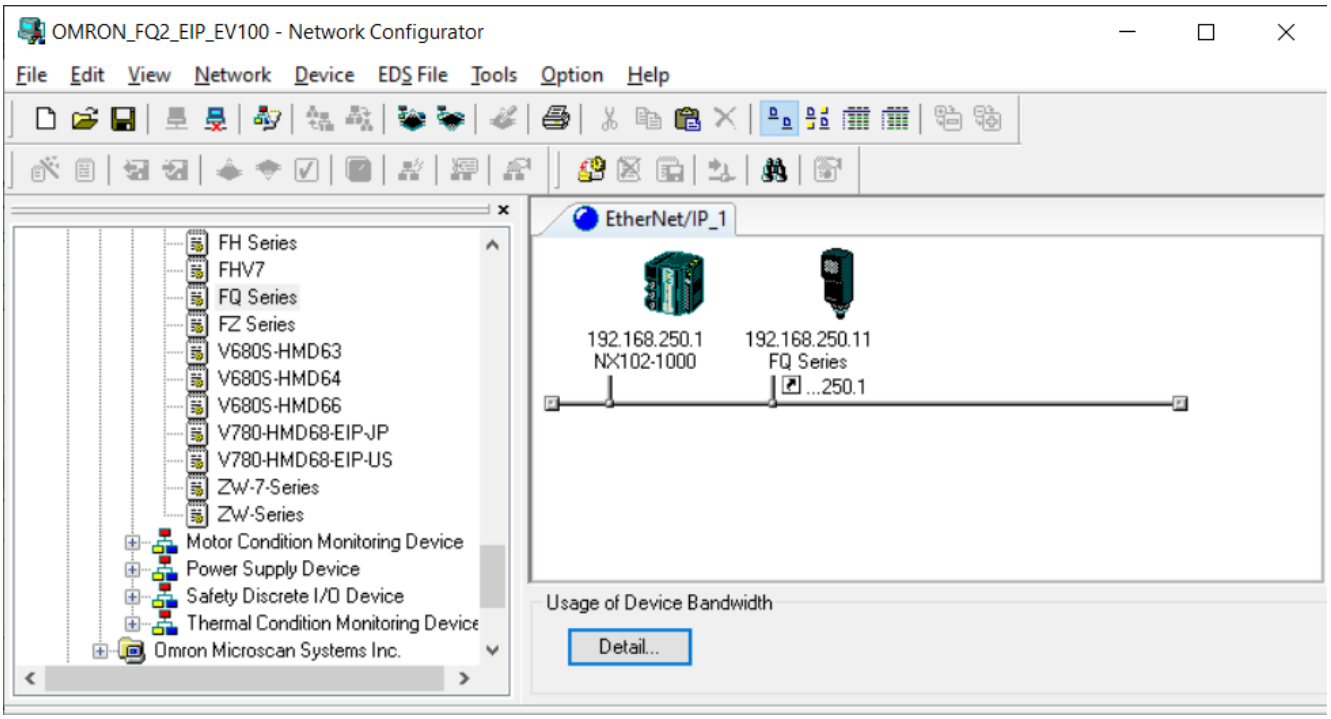


(a) Sidebilde av skeiv kork **(b)** Motsatt sidebilde av skeiv kork

Figur 2.52

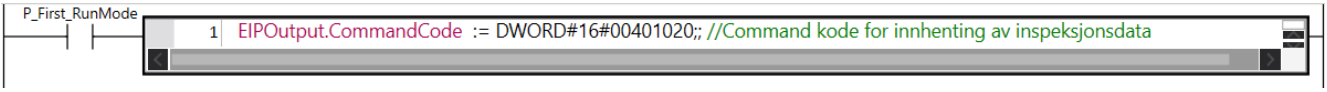
2.6.4 PLS Program og Kommunikasjon

Ethernet/IP blir tatt i bruk for kommunikasjon mellom PLS og Smartkamera. Det blir brukt strukturvariabler levert av Omron [26] for uthenting av relevant informasjon fra utgangsregisterert til smartkamerat. For å etablere "Data link" mellom strukturvariablene i Sysmac Studio og utgangsregisterert til kameraet, blir strukturvariablene konfigurer i programmet "Network Configurator" som følger med Omrons pakke ved installering av Sysmac Studio. Dette programmet etablerer data link mellom variabler i Sysmac Studio og andre Omrom komponenter. Strukturvariablene blir lastet opp og konfigurert mellom NX102-1000 og FQ2-S i programmet som vist i figur 2.53.



Figur 2.53: Nettverkskonfigurasjon mellom PLS og Smartkamera

For å spesifisere hvilket utgangsregister man vil hente data fra hver gang man sender dataetterspørsel, må man definere variabelen "CommandCode" i strukturvaiabelen "EIPOutput" med en "Word" verdi. Fra datablad [25] så vil hexadesimalverdien lagt inn i CommandCode vist i figur 2.54 kommandere anskaffelsen av resultatet fra inspeksjonsfunksjonene fra seksjon 2.6.3 ved dataetterspørsel.

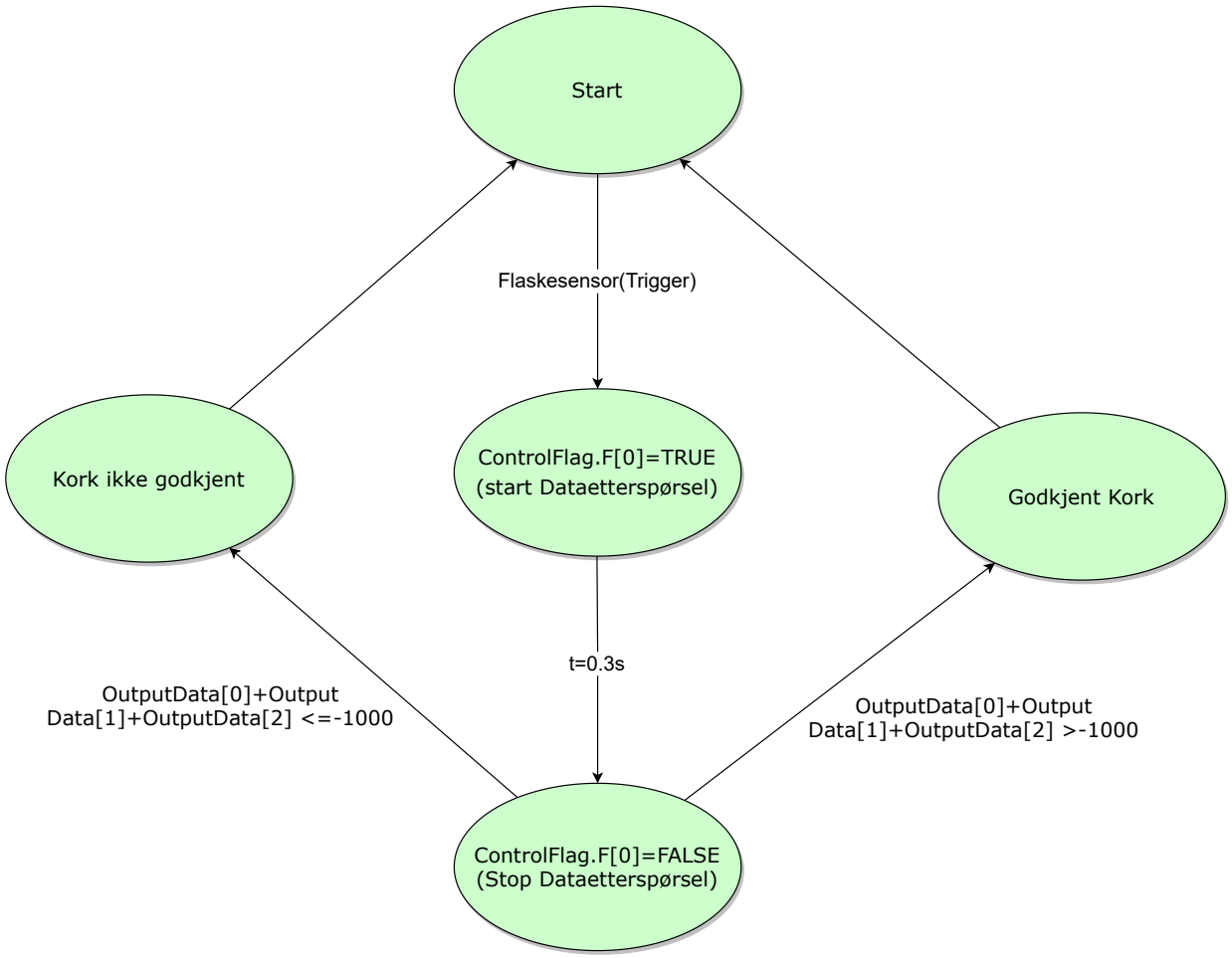


Figur 2.54: Definering av CommandCode variabelen i strukturvariabelen

Dataetterspørsel basert på innholdet i CommandCode blir gjort hver gang variabelen "ControlFlag.[0]" i strukturvariabelen EIPOutput blir aktivert.

Innhentet data fra inspeksjonsfunksjonene blir lagret som element i strukturvariabelen EIPOutput.OutputData[ ]. Resultatet fra inspeksjonsfunksjonene fra seksjon 2.6.3 har blitt spesifisert til å gi ut boolske vurderinger av bildeanalysen til utgangsregisteret. Gitt at det er tre inksepsjonsfunksjoner, bruker man EIPOutput.OutputData[0-2] for innhenting av inspeksjonsresultatene. Verdien 0 indikerer "godkjent" (OK) og verdien -1000 indikerer "ikke godkjent" (NG) per inspeksjonsfunksjon i Sysmac Studio.

Tilstandsdiagrammet i figur 2.55 beskriver den sekvensielle funksjonen til vedlagt kode A.5 for vurdering av korkposisjonen på flaskene.



Figur 2.55: Betingelsen fra tilstanden (start dataetterspørsel) til (stop dataetterspørsel) blir det brukt en timer på 0.3 sekunder for å forhindre koden i å viderkjøre før Sysmac Studio har fått tid til å faktisk hente data fra smartkameraet.

For hver gjennomkjøring hvor korken ikke er godkjent, vil variabelen "Korkfeil_teller" som vist i figur 2.56 aktiveres. Dette medfører at den globale variabelen for "Bildeanalyse_korkfeil" og "Bildeanalyse_Korkfeil_prosent" inkrementere.

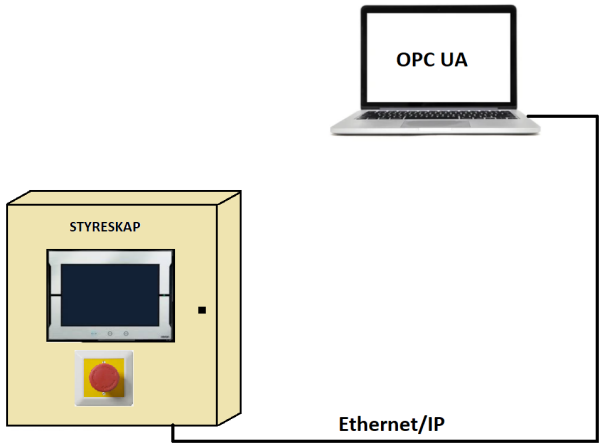
Figur 2.56: Global variabel for lagring av antall bildeanalyse korkfeil

2.7 OPC UA

En ønsket del av oppgaven var å etablere en OPC UA kommunikasjonsprotokoll mellom datamaskin og PLS. Dette er for å kunne få statusrapport av prosessen samt loggføre prosessens forløp og eventuelle feilmeldinger. John ønsket at det ble presentert to ulike løsninger for OPC UA kommunikasjon: En der den eksisterende PLS CP1L blir brukt, og en der NX102 blir brukt.

Figur 2.57 viser skjematisk OPC UA oppkobling mellom PLS og datamaskin over Ethernet/IP tilkobling.

2.7.1 Innledning



Figur 2.57: OPC UA kommunikasjon mellom PLS i styreskap og datamaskin over Etnernet/IP

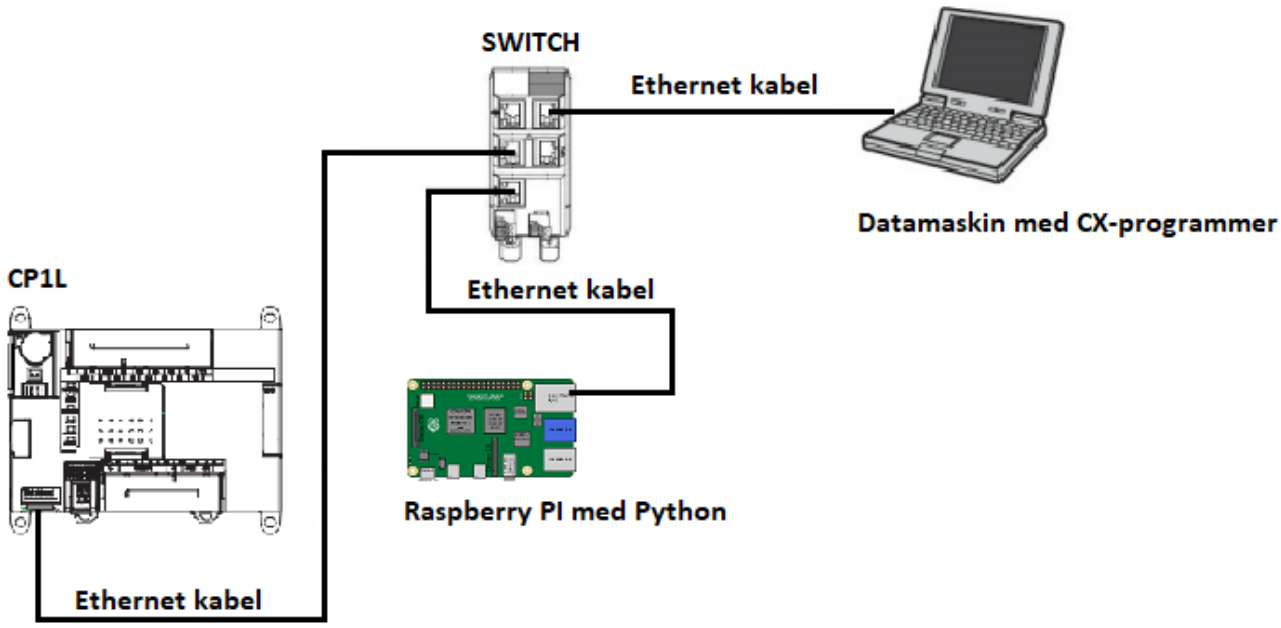
For å løse kommunikasjonen med CP1L så bruktes det en Raspberry Pi som mellomvert som både settes opp med modbus kommunikasjon og OPC UA kommunikasjon. Med bruk av NX102 viste det seg å være en relativt triviell metode for å etablere OPC UA kommunikasjon, da denne type PLS har en innebygget modul for nettopp denne kommunikasjonsprotokollen. For begge metodene er klientprogrammet UaExpert brukt på datamaskin til å vise status til PLS.

2.7.2 OPC UA PLS: CP1L

Som nevnt innledningsvis til kapittelet så ble det brukt en Raspberry Pi som mellomvert for å danne kommunikasjonen mellom CP1L og datamaskin. Kort forklart så er Raspberry Pi en liten datamaskin der en blant annet kan installere og ta i bruk ulike programmer. Python ble i dette tilfellet installert og tatt brukt for å programmere Raspberryen til å fungere som både en klient for modbus kommunikasjon og som en server for OPC UA kommunikasjon.

Grensesnitt

Bildet 2.58 viser en skisse av oppkoblingen.

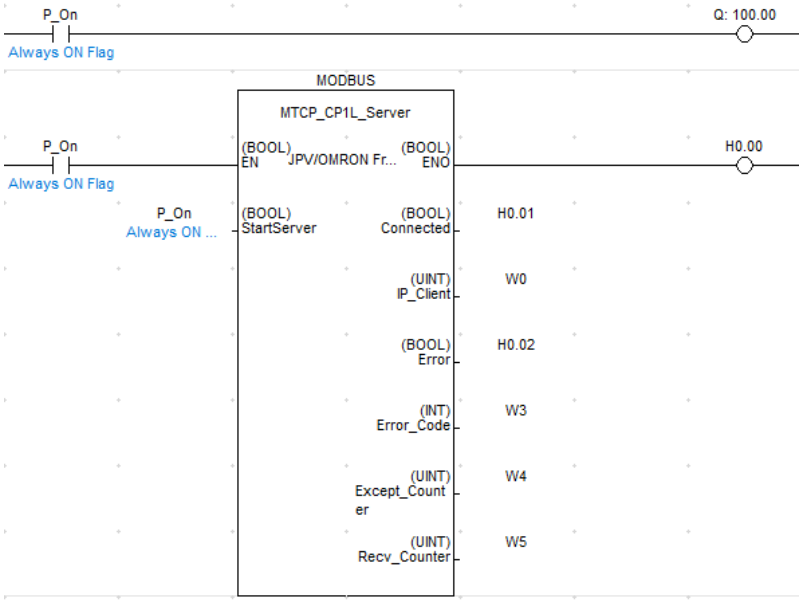


Figur 2.58: PLS, Raspberry Pi og datamaskin kommuniserer via ethernet/IP og er koblet sammen med bruk av en switch.

Modbus kommunikasjon

Modbus kommunikasjon mellom PLS og Raspberry må dannes for å kunne lese verdier og status fra CP1L i klientprogrammet UaExpert. Teori rundt modbus protokoll blir ikke omtalt i detalj her, men modbus baserer seg også på klient/server (evt. master/slave) kommunikasjon [38]. Med bruk av CX-programmer så ble PLS her konfigurert til å være server og med bruk av Python ble Raspberry satt til å være klient.

I CX-programmer så benyttes funksjonsblokken "MTCP_CP1L_Server" for å konfigurere CP1L til å være modbus server. I bildet 2.59 så vises funksjonsblokken som er brukt. Konfigurasjon av funksjonblokken er gjort ved hjelp av youtube filmen funnet i [31] og databladet til funksjonsblokken [21]



Figur 2.59: Funksjonsblokk modbus server fra datablad [21]

For å få til modbus protokollen med Python så ble biblioteket "pymodbus" installert. Pakken "ModbusTcpClient" ble brukt for å danne kommunikasjon og for å kunne bruke modbus funksjonaliteter. Siden kommunikasjonen går over ethernet/IP så brukes modbus varianten TCP (Transmission Controll Protocol).

Når biblioteket er installert og "ModbusTcpClient" pakken er importert inn i scriptet, så kan man opprette kommunikasjon med PLS med å oppgi dens IP adresse. I kodeutsnittet under vises det hvordan det er løst i denne oppgaven.

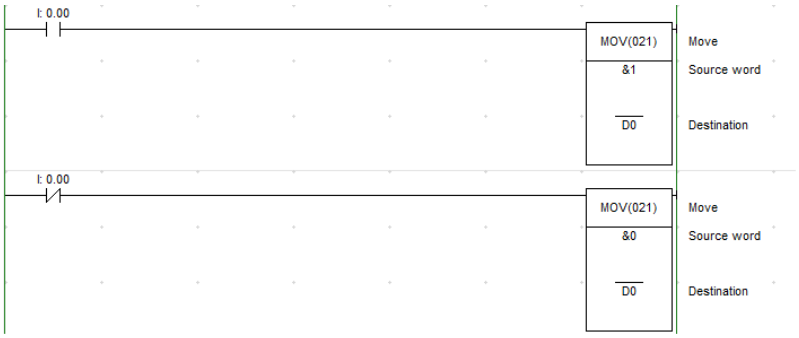
```
1 from pymodbus.client.sync import ModbusTcpClient as modbus
2 pls_IP = '192.168.250.1'
3 client = modbus(pls_IP)
4 PLS = client.connect()
```

Med bruk av modbus kommunikasjon er det mulig å både skrive og lese statusverdier til PLS. I denne oppgaven har det blitt løst lesing av status på inngangssignaler for CP1L. Dette kan løses på ulike måter, her ble det gjort ved å bruke funksjonen "read_holding_registers". Funksjonen "read_discrete_inputs" benyttes ikke på grunn av funksjonsblokken ikke støtter denne funksjonen [21]. I kodeutsnittet under så vises det hvordan det har blitt løst for å lese inngang 0 på CP1L.

```
1 def input_0():
2     result = client.read_holding_registers(0, 12)
3     input_0 = bool(result.registers[0])
4     return input_0
```

For hver inngang så har det blitt laget en Python-funksjon som leser de 12 PLS-inngangene 0 til 11 og returnerer et boolsk uttrykk for den gitte inngangen som vist i kodeutsnitt overnfor. Fullstending Python kode alle inngangene finnes i vedlegg A.11.

I CX-programmer er det blitt programert slik at når en inngang går høy så skal tallet "1" sendes til D minnet(holding registeret) i PLS'en. Når inngangen går lav så sendes tallet "0". Hvordan dette er løst for inngang 0 vises i figur 2.60. Dette er gjort for alle PLS-inngangene, se CX kode i vedlegg A.10.



Figur 2.60: Sending av 1 og 0 til D minnet i CP1L

OPCUA server

Raspberry ble som sagt også programmert til å være server for OPCUA kommunikasjonen. For å få til dette, ble biblioteket "freeopcua" i Python installert og pakken "Server" importert. Hele koden er å finne i vedlegg A.12 I kodeutsnittet under vises oppkobling av serverfunksjon og tilknytting til Raspberry sin IP adresse.

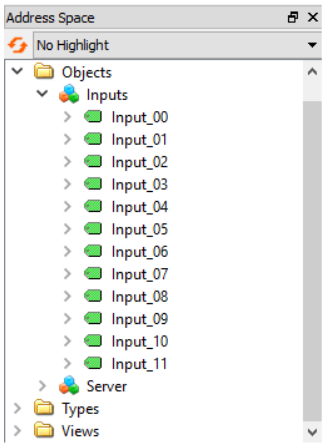
```
1 # Importerer OPCUA serverfunksjoner fra opcua bibliotek samt time funksjon og ...
      scriptet modbus.py
2 from opcua import Server
3 import time
4 import modbus
5
6 server = Server()
7 IP_raspberry = "opc.tcp://192.168.250.2:4840"
8 server.set_endpoint(IP_raspberry)
```

Importerer modbus.py scriptet, for å lese inputstatus på CP1L i OPCUA klienten. For å få tilgang til å lese ønsket verdier i UaExpert, må det også opprettes en nodeklasse som man videre kan knytte noder til som inneholder ønsket variabler. Denne nodenklassen vil da fungere som en mappe som man kan utforske i og å hente ut variabler fra noden som man vil lese i visningviduet i UaExpert, se figur 2.61. I Python er dette løst slik som i kodeutsnittet under.

```
1 # Oppretter nodeklassen objects
2 node = server.get_objects_node()
3 # Knytter noden Inputs til nodeklassen objects
4 Inputs = node.add_object(addspace, "Inputs")
5 #Legger inn variabler som skal leses
6 input0 = Inputs.add_variable(addspace, "Input_00", 0)
```

Som kodeutsnittet viser så er input0 lagt inn som en variabel med navn "Input_00" til Inputs for nodeklassen. Her blir status til inngang 0 lagt inn. Det samme har blitt gjort for alle de 12 inngangene.

Bilde 2.61 viser hvordan dette blir seende ut i "Address Space" vinduet i UaExpert.



Figur 2.61: Viser Objects mappen, noden Inputs og inngangsvariblene knyttet til CP1L

Selve verdien til inngangene for CP1L er ikke enda knyttet til "Input_00" til "Input_11". En må først sette variablene til å kunne endres/skrives. Dette løses slikt

```
input0.set_writable()
```

Deretter hentes verdiene fra modbus.py

```
while True:

    # Henter statusverdier fra PLS via modbus.py

    Input0 = modbus.input_0()
```

For å så knytte verdiene hentet fra modbus.py scriptet til variablene laget i UaExpert

```
    # Setter verdiene
    input0.set_value(Input0)
```

Når UaExpert er koblet til serveren så kan verdiene leses av i "Data Access View" vinduet.

Data Access View

Server	Display Name	Value	Datatype	Source Timestamp	Statuscode
OPCUA server	Input_00	false	Boolean	13:35:17.461	Good
OPCUA server	Input_01	false	Boolean	13:35:17.461	Good
OPCUA server	Input_02	false	Boolean	13:35:17.461	Good
OPCUA server	Input_03	false	Boolean	13:35:17.461	Good
OPCUA server	Input_04	false	Boolean	13:35:17.461	Good
OPCUA server	Input_05	false	Boolean	13:35:17.461	Good
OPCUA server	Input_06	false	Boolean	13:35:17.461	Good
OPCUA server	Input_07	false	Boolean	13:36:31.520	Good
OPCUA server	Input_08	false	Boolean	13:35:17.461	Good
OPCUA server	Input_09	false	Boolean	13:35:17.461	Good
OPCUA server	Input_10	false	Boolean	13:35:17.461	Good
OPCUA server	Input_11	false	Boolean	13:35:17.461	Good

Figur 2.62: Viser statusverdi til inngangene for CP1L

I figur 2.62 vises alle verdiene i kollonnen "Value" som "false", dette på grunn av alle inngangene til CP1L er lave her. For å verifisere det at en inngang går høy også registreres i UaExpert så "tvinges" vilkårlige innganger til å få høy status ved å laske inngangene til PLSens spenning. Dette er vist i figur 2.63

Figur 2.63: Lask på inngang 0 for å gi høy status på inngangen

Fra bilde 2.63 kan man se at inngang 0 lyser når inngangen er høy. Denne statusendringen vises også i UaExpert. Figur 2.64 verifiserer dette.

Data Access View

Server	Display Name	Value	Datatype	Source Timestamp	Statuscode
OPCUA	Input 0	true	Boolean	15:34:20.952	Good
OPCUA	Input 1	false	Boolean	15:34:20.953	Good
OPCUA	Input 2	false	Boolean	15:32:08.019	Good
OPCUA	Input 3	false	Boolean	15:32:08.019	Good
OPCUA	Input 4	false	Boolean	15:32:53.779	Good
OPCUA	Input 5	false	Boolean	15:32:08.019	Good
OPCUA	Input 6	false	Boolean	15:32:08.019	Good
OPCUA	Input 7	false	Boolean	15:32:08.019	Good
OPCUA	Input 8	false	Boolean	15:32:08.019	Good
OPCUA	Input 9	false	Boolean	15:32:08.019	Good
OPCUA	Input 10	false	Boolean	15:32:08.019	Good
OPCUA	Input 11	false	Boolean	15:32:08.019	Good

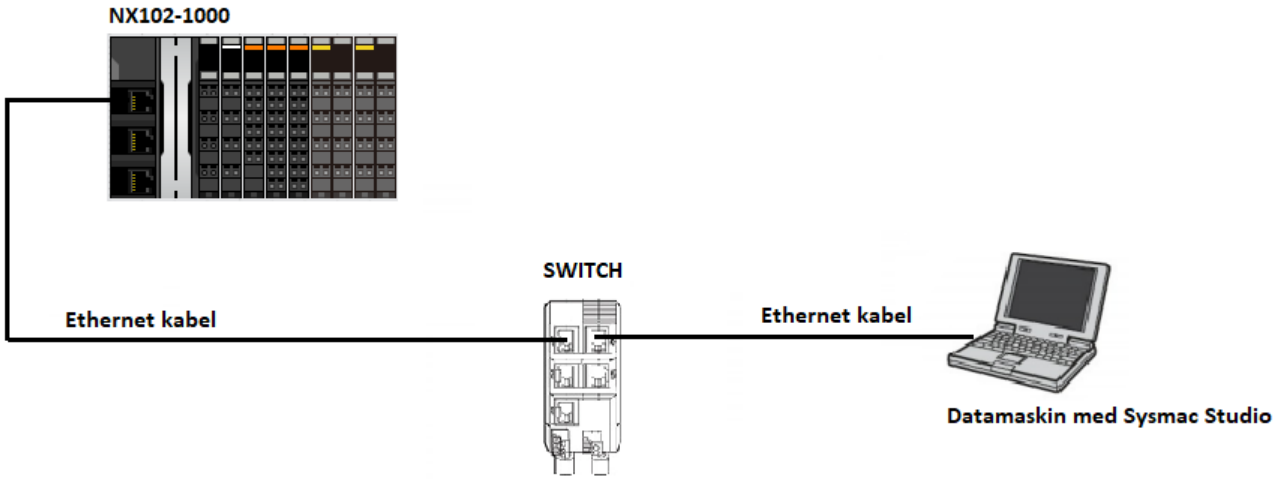
Figur 2.64: Verifisering av statusendring i UaExpert

2.7.3 OPC UA PLS: NX102-1000

For å etablere OPC UA kommunikasjonen mellom datamaskin og NX102 så gjøres dette via Sysmac Studio. Kommunikasjonen går over ethernet/IP. I Sysmac Studio kan man sette opp en OPC UA server i PLS og bruke et klientprogram på datamaskin, her brukt UaExpert. Løsningsmetoden i denne oppgaven har følgt fremgangsmåten utført i youtube filmen til Omron [27].

Grensesnitt

Selve tilkoblingen mellom datamaskin og PLS er som i figur 2.65

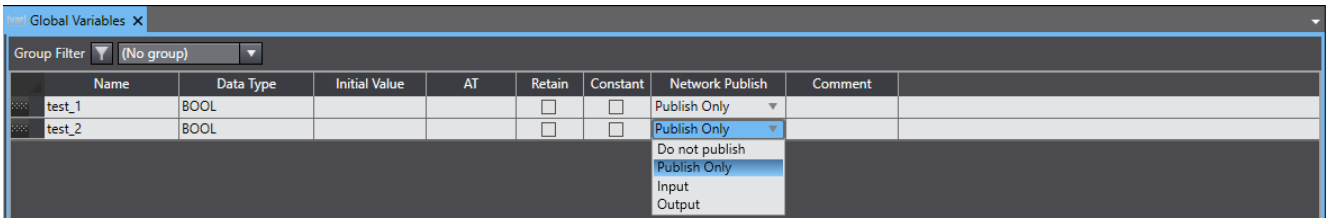


Figur 2.65: OPC UA oppsett mellom datamaskin og NX102

Fremgangsmåte og resultat

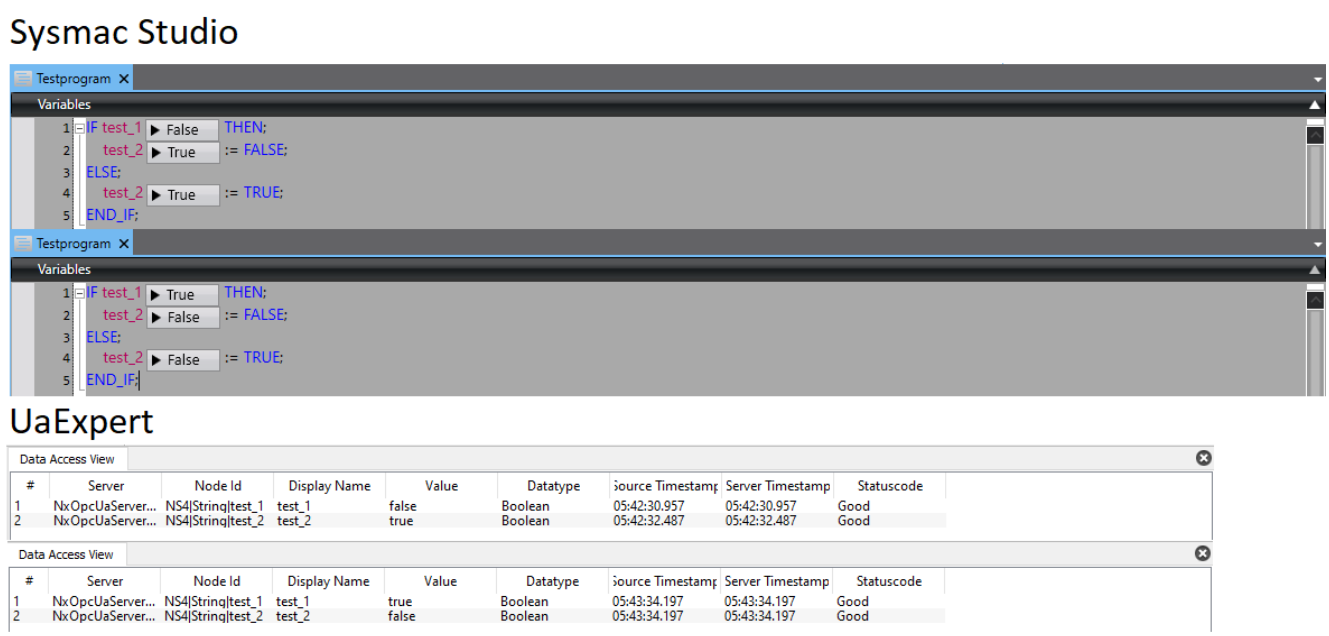
Prinsipiell presentasjon av løsning blir fremstillet med ett enkelt program laget i Sysmac Studio. Dette er fordi den prinsipielle løsningen også fungerer fint til hovedprogrammene laget til de ulike sekvensene og at det blir mer oversiktlig å vise funksjonaliteten.

For å kunne se status på en variabel i UaExpert, må variabelen være satt som global variabel og gjøre den synlig ved å velge "Publish Only" under "Network Publish" innstillinger som vist i figur 2.66



Figur 2.66: Konfigurering av globale variabler

Figur 2.67 viser enkelt program laget i Sysmac Studio med to boolske testvariabler og verdien vist i UaExpert når verdi blir endret.



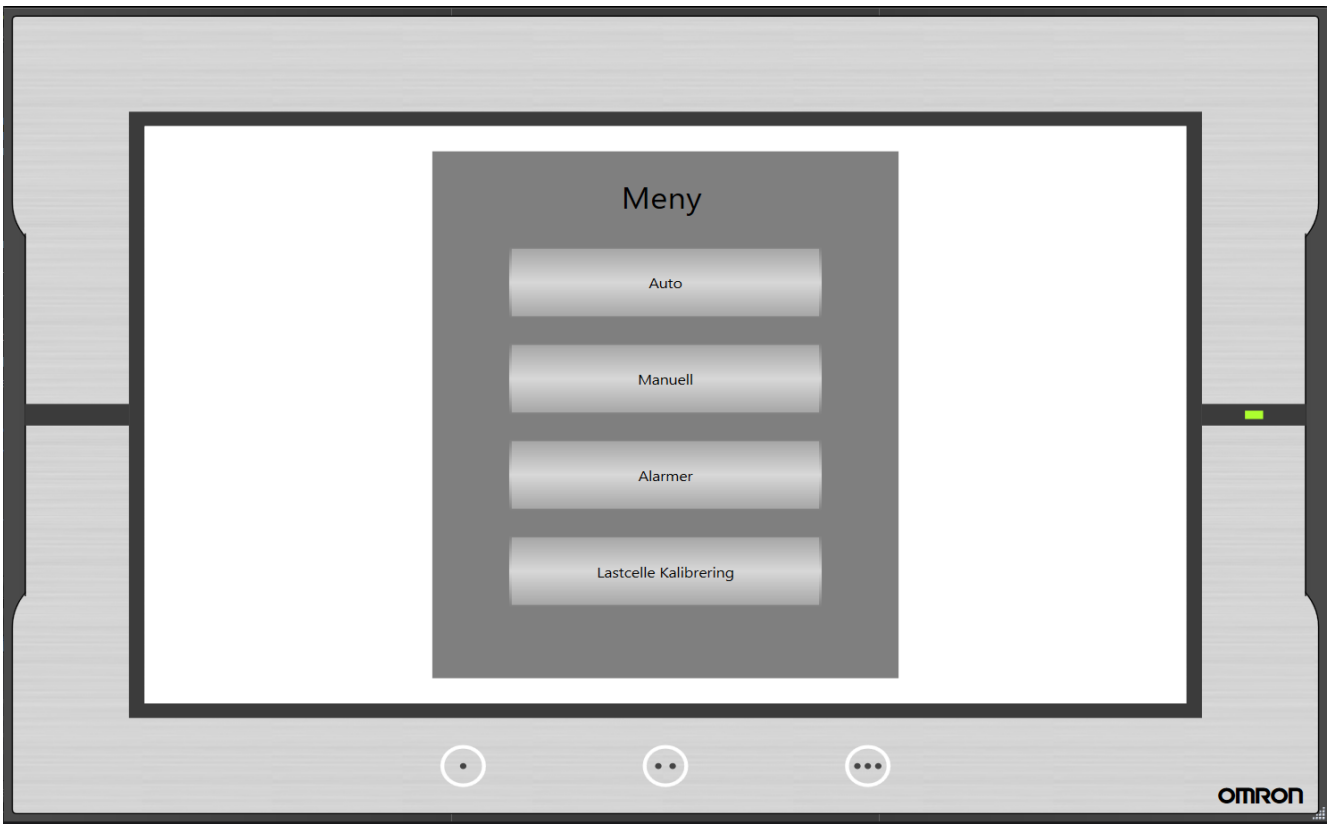
Figur 2.67: Variabelstatus i Sysmac Studio og UaExpert

2.8 HMI

2.8.1 Innledning

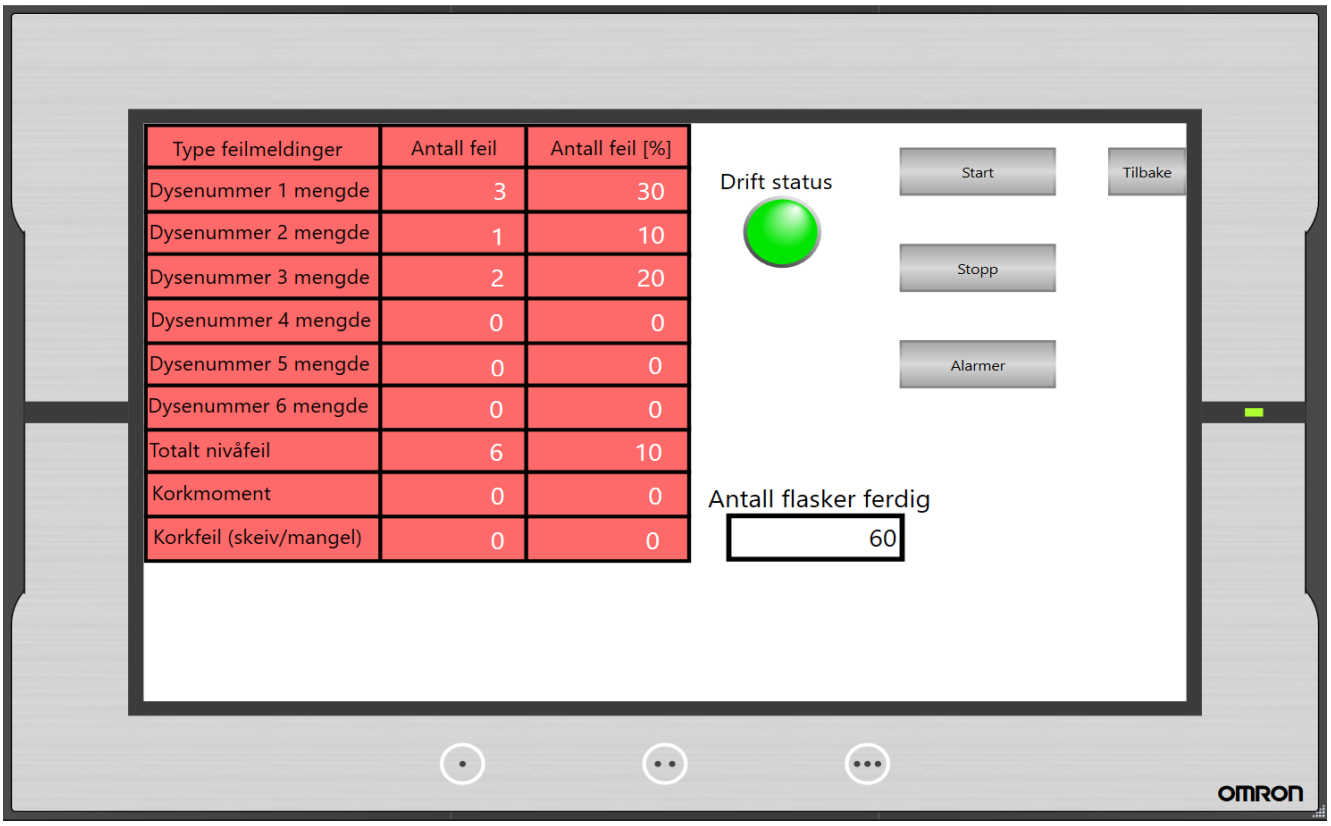
I dette kapittelet vil det blir presentert en brukervennlig HMI for forskjellige styringer samt monitorering av prosessen. Hver HMI-skjerm vil bli fremstillet med en forklarende figurtekst.

Meny



Figur 2.68: Dette er første skjermen man ser ved oppstart av PLS. Fra her så har man muligheten til å velge de fire alternativene som er fremstillet. "Auto" for automatisk styring og monitorering av feilmeldinger i prosessen under drift. "Manuell" for manuell styring av aktuatorer og monitorering av sensorer ved manuell kjøring, denne HMI-skjermen er hovedsaklig for feilsøking. "Alarmer" for monitorering av alarmer under drift og historikkalarmer. "Lasecelle kalibrering" for kalibrering av Lastcellen

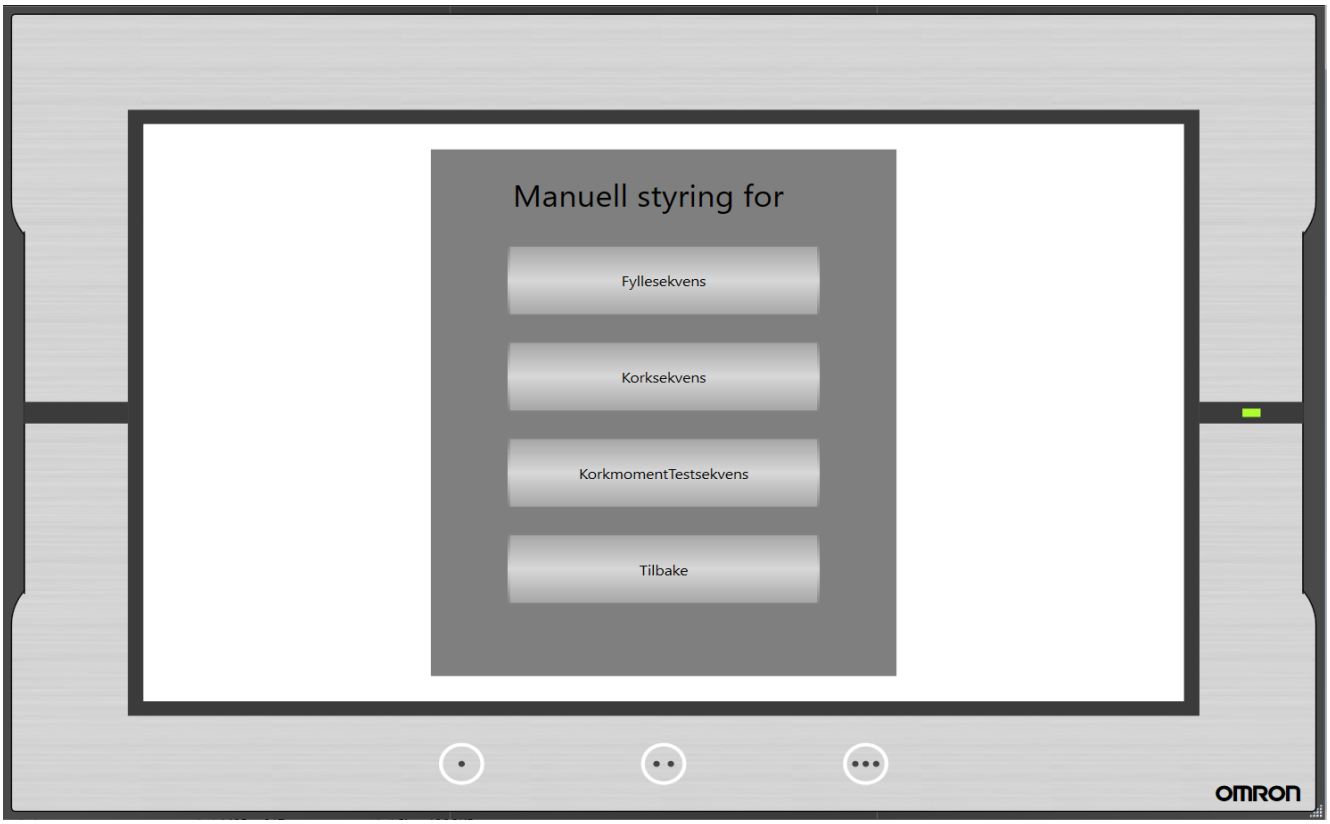
2.8.2 Auto



Figur 2.69: Denne skjermen blir brukt for automatisk drift av hele prosessen. Her kan man monitorere alle feilmeldingene som kan oppstå i prosessen. Disse feilmeldingen blir fremstillet i ”Antall feil” og ”Antall feil” i prosent. Knappen ”Alarmer” gir direkte tilgang til Alarm-skjermen.

Et eget ladderprogram er implementert for dirkete styring av alle sekvensprogrammene fremstillet i vedlegg A.6

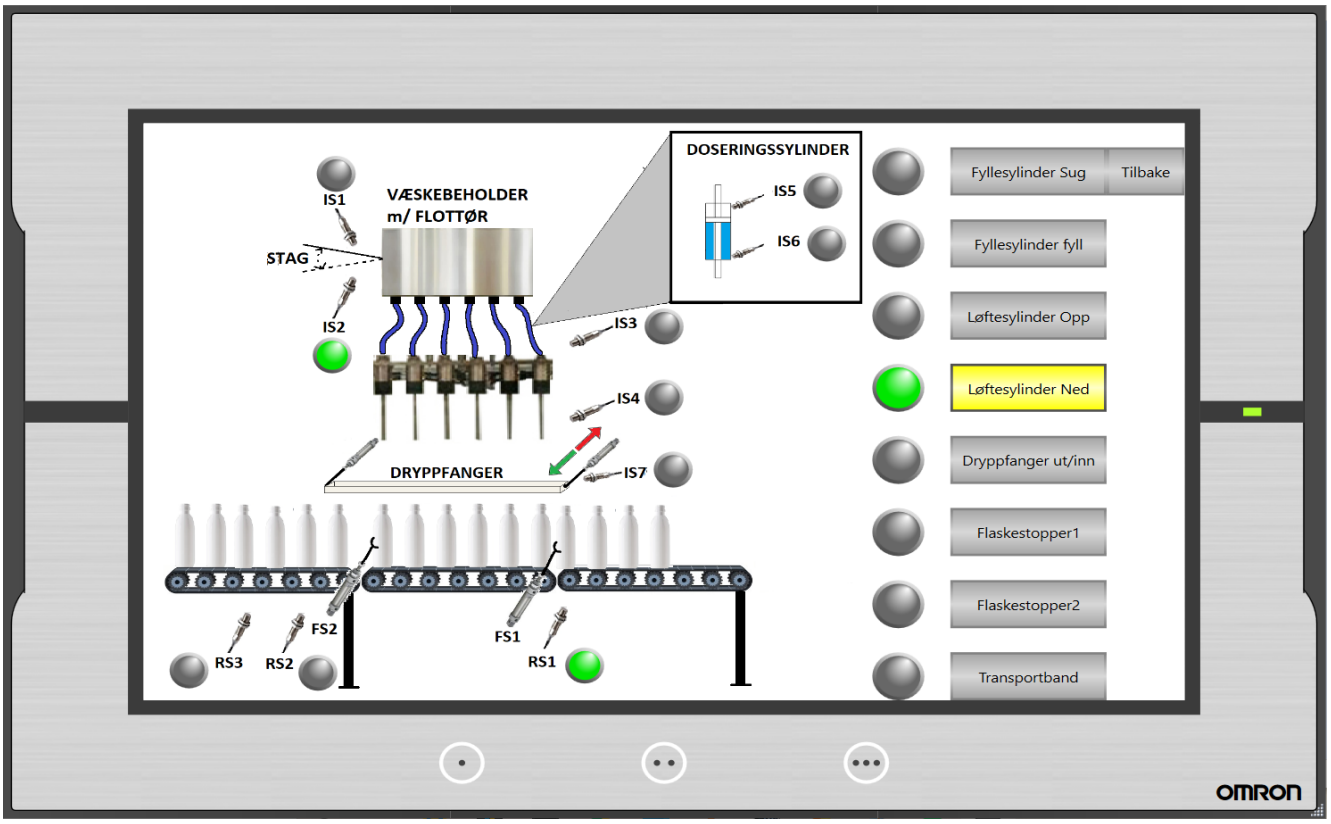
2.8.3 Manuell



Figur 2.70: Etter man har trykket på "Manuell" på menyskjermen som vist i seksjon 2.8.1, kommer man til denne HMI-skjermen. HMI-skjermen inneholder valgmulighetene for manuell styring av alle aktuatorer i fyllesekvens, korksekvens og korkmomenttest sekvens. Styring av transportband vil være inkludert i alle alternativene.

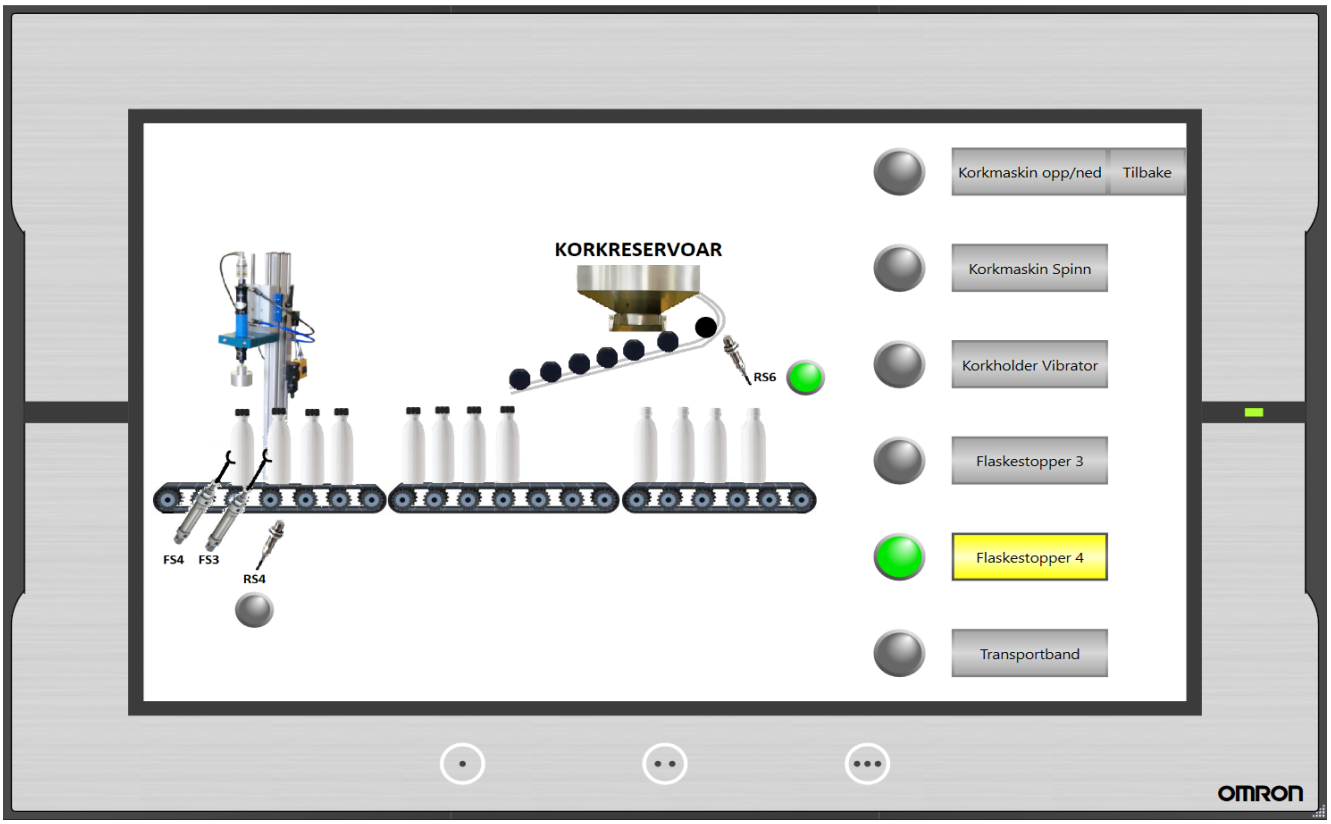
PLS program for manuell styring finner man i vedlegg A.7

Manuell styring fyllesekvens



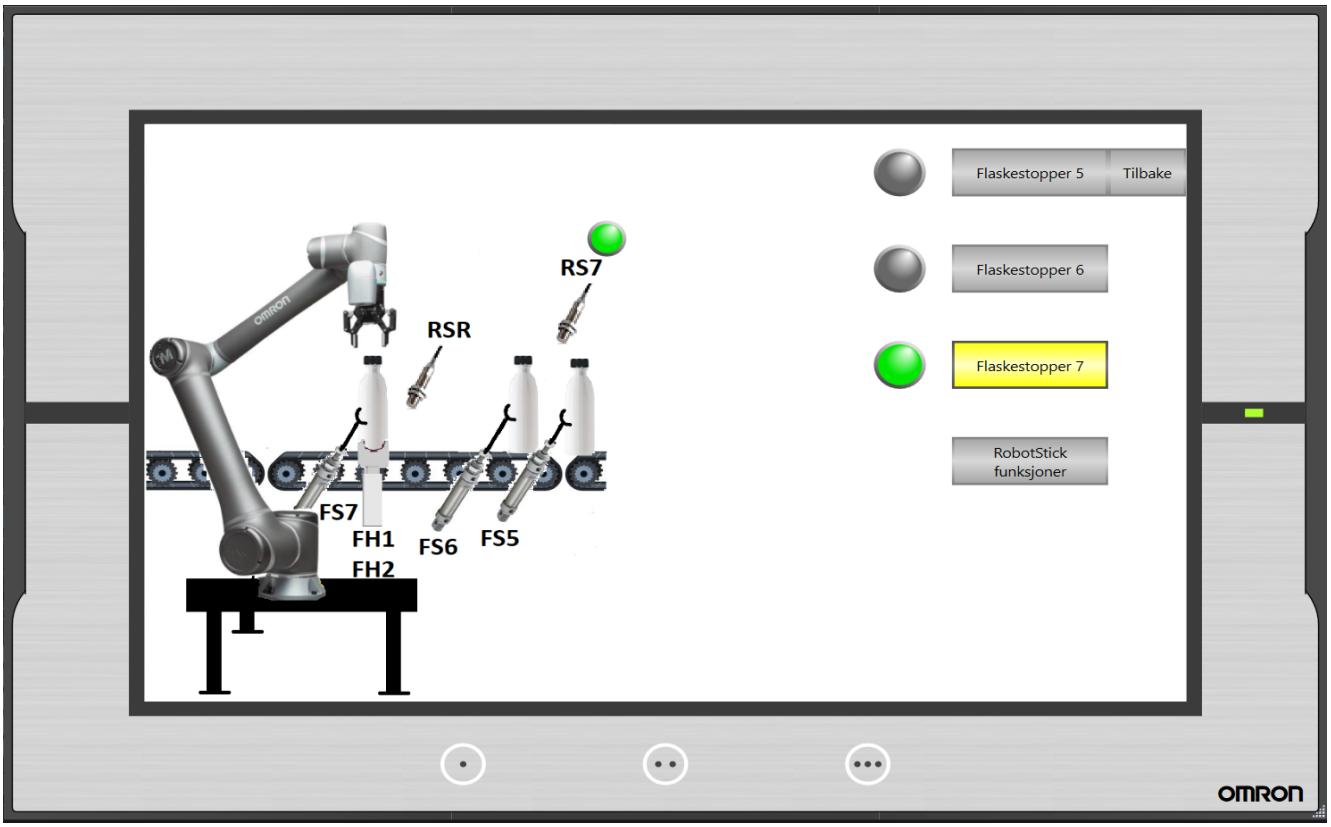
Figur 2.71: Denne HMI-skjermen har knapper på høyre siden som styrer alle aktuatorene i sammenheng med fyllesekvensen. Lamper ved siden av knappene er tilknyttet de respektive aktuatorene for å indikere av (grå) eller på (grønn) tilstand. Knappene er av typen ”toggle” som betyr et trykk for ”på” tilstand og et trykk til for ”av” tilstand. Dette er for å slippe programmering av holdebrytere i ladderprogrammet for manuell styring. Bilde av selve fyllesekvensen er fremstillet på venstresiden av skjermen, hvor man har lamper som indikerer av (grå) eller på (grønn) tilstanden til sensorene som de er plassert ved siden av tilhørende lampe.

Manuell styring korksekvens



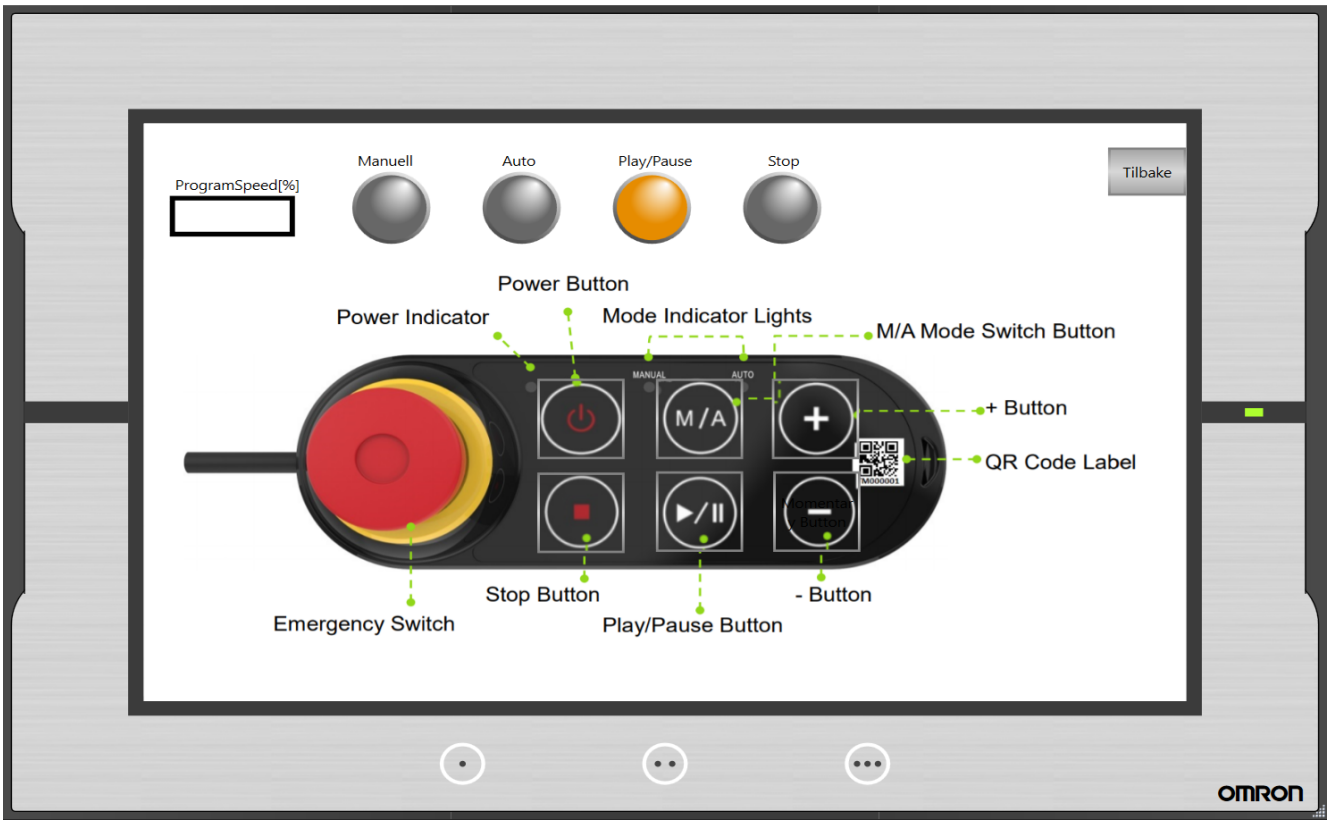
Figur 2.72: Denne HMI-skjermen har knapper på høyre siden som styrer alle aktuatorene i sammenheng med korksekvensen. Lamper ved siden av knappene er tilknyttet de respektive aktuatorene for å indikere av eller på tilstand. Alle knappene er av typen ”Toggle” utenom ”Korkmaskin opp/ned”, denne er av typen ”Momentary”. ”Momentary” knappen sender ut "høytsignal ved trykk og lavt"ved slepp. Korkmaskinen styres av impulser, 1 impuls for ned og deretter 1 for opp. Lamper er plassert ved sensorene for å indikere av eller på tilstand.

Manuell styring korkmoment test sekvens



Figur 2.73: Denne HMI-skjermen har knapper på høyre siden som styrer alle stoppesylinderene til korkmomenttest sekvensen. FH1, FH2 og RSR er direktet koblet opp med kontrollerboksen til roboten, derfor har man ikke tilgang til styring og tilbakemeldinger fra disse komponentene. Man har tilgang til tilbakemelding fra RS7, som er koblet opp mot PLSen. Knappen "Robotstick funksjoner" omdirigerer brukeren til en egen skjerm for robotstick fuksjoner.

Robotstick funksjoner



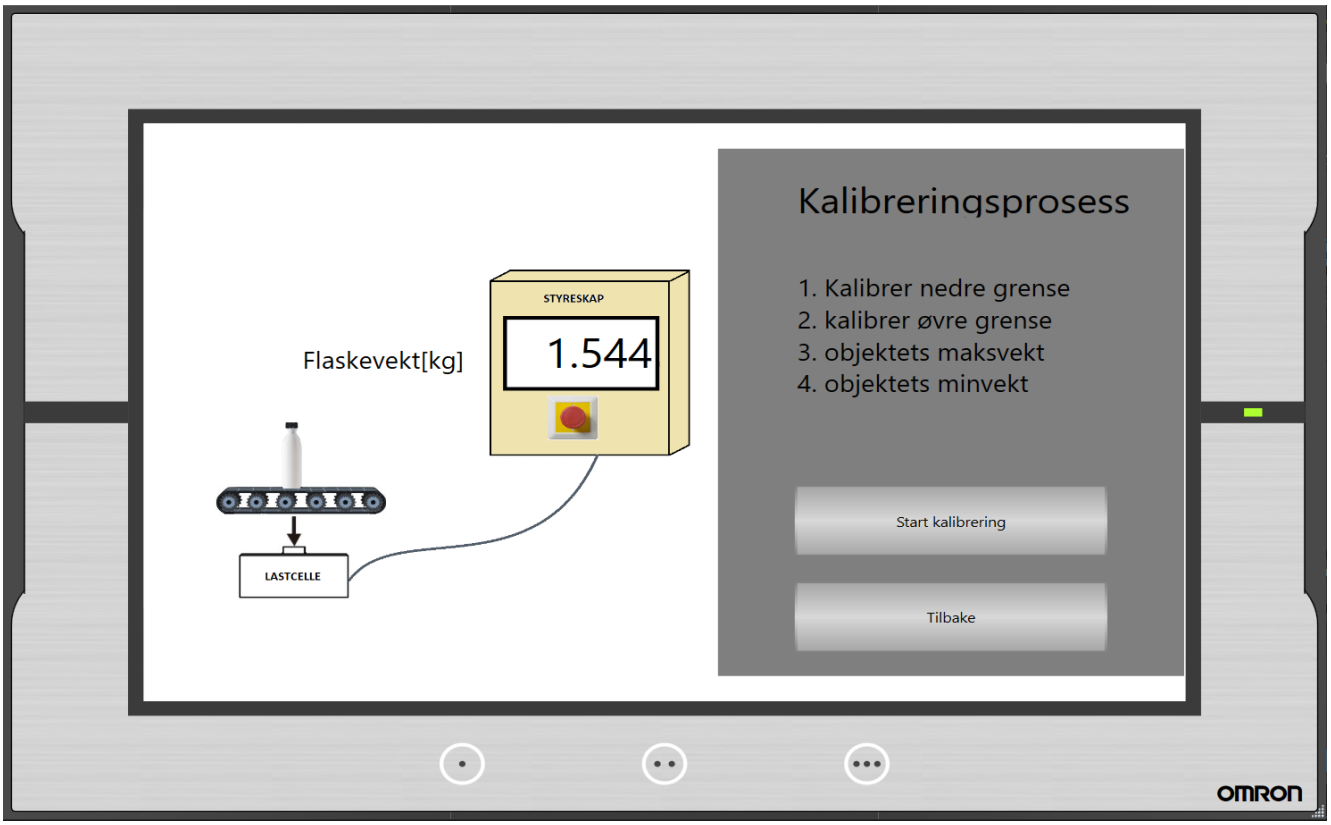
Figur 2.74: Denne skjermen inneholder knappene til robotstick kontrolleren til roboten brukt i denne oppgaven. Det er fremstillet et bilde av selve robotsticken på denne HMI-skjermen, hvor knappene på selve bilde kan man fysisk trykke på. Teksten som er tilknyttet knappene beskriver funksjonaliteten. Lampene indikerer tilstanden til de respektive knappene. "Play/pause" lampen lyser grønt under drift og oransje ved pause.

2.8.4 Alarmer

Figur 2.75: Denne HMI-skjermen inneholder en liste over alle alarmer. Det er tre typer feilmeldinger som blir definert som "Alarmer", "Korkmaskin_tom" som indikerer mangel på korker i korkreservoar, denne har prioritetsnivå 1 som der av høyeste nivå. Alarm nummer 2 er "Flaskekø", denne beskriver at det har oppstått kø, noe som indikerer at mating av flasker bør stoppes. Denne alarmen har prioritetsnivå 3, dette er ikke en alarm som behøver umiddelbar oppmerksomhet. Alarm nummer 3 er "Væskebeholder lavt nivå", denne alarmen har prioritetsnivå 1 siden mangel på væske krever umiddelbar oppmerksomhet for å forhindre feilprodusering.

Figur 2.76: Eksempel på alarm notifikasjon under drift, feilmeldingen forsvinner hvis man trykker på "Acknowledge". Det vil bli vist i alarmhistorikken på HMI-skjermen 2.75 etter man har trykket på "Acknowledge"

2.8.5 Lastcelle kalibrering



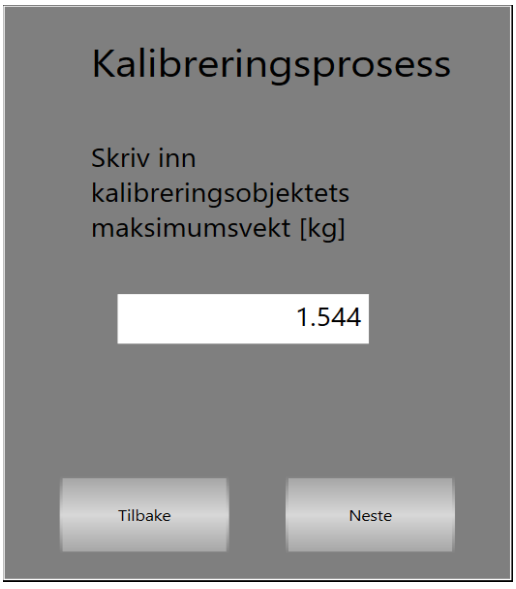
Figur 2.77: Denne HMI-skjermen inneholder et bilde av veiesekvensen med et display som indikerer nåverdi fra lastcellen. Dette displayet er for å verifisere resultatet etter kalibreringen. Kalibreringsprosessen begynner når brukeren trykker på "Start kalibrering" hvor man må gjennom alle trinnene listet i grå boks.

Kalibreringsprosess

Kalibreringsprosessen vil være selvforklarende med teksten fremstillet i hvert trinn i HMI-en. Etter man har utført et trinn, trykker man på "Neste" for å komme til neste kalibreringstrinn.

(a) Første kalibreringstrinn

(b) Andre kalibreringstrinn



(c) Tredje kalibreringstrinn

(d) Fjerde kalibreringstrinn, man kommer tilbake til hoved-kalibrerings skjermen etter dette avsluttende trinnet

Kapittel 3

Diskusjon

3.1 Korkmoment test

Mye av tiden i begynnelsen gikk i selvlæring av TMFlow. Gitt at låneperioden var på to uker så ble foreslått program ikke så ferdig som ønsket. Det er blitt oppdaget småfeil med programmet i etterkant, og redigering vil ikke være mulig grunnet man trenger direkte tilkobling til selve roboten for å åpne TMFlow. Nåværende oppdaget problem er at roboten ikke vil fullføre sekvensen hvis kommende flaske mangler kork. "Vision noden" vil stå fast i en evig løkke hvis ingen kork blir oppdaget, dette medfører at roboten aldri vil slippe forbi flasken. Det som kan gjøres for å forhindre dette problemet er å programmere en ekstra "Vision node" før den nåværende implementerte "Vision noden" for å sjekke om kommende flaske **ikke** har kork. Hvis "Vision noden" registrerer mangel på kork, vil programmet hoppe over selve moment-testingen, kvittere feil til PLS og deretter slippe forbi flasken. Hvis denne noden ser en kork, vil noden registrere feil og deretter fullføre original sekvensrutine for momenttesting.

Verktøy
Roboten ble levert med griperverktøyet vist i innledende bilde til seksjon 2.4 fra Omron. Dette verktøyet påførte komplikasjoner under labtesting. Griperen skapte ikke nokk friksjon mellom gripergummi og flaskekork for utløsning av sikkerhetsfunksjonen til "Compliance" noden for godkjenning av korkmoment i programmet. Dette medførte at griperen ville skli på korken. Som vist i vedlagt video i innledningen 2.4 måtte man bruke handkraft for å simulere motstandsmomentet til en tilstrekkelig påskrudd kork. Hvis NOS ønsker denne roboten implementert for korkmoment testing må det installeres et nytt verktøy, eksempel på anbefalt verktøy er fremstillet i seksjon 2.4 figur 2.30.

Begrensningene for dreieretning og momenttesting
Den eneste løsningen som ble oppdaget for selve momenttestingen, var å bruke compliance noden som beskrevet i seksjon 2.4.2 i seksjonen om korkmoment testing. Ulempene med denne funksjonen var begrensningene på dreieretning og detektering av momentmotstand. Man hadde kun mulighet for dreieretningen CW, som er dreieretningen får å skru fast korker. Ifølge artikkelen [24] så bør man momentteste i retning CCW med 40-60% av applikasjonsmomentet (momentet brukt for å fastskru korken).

Problemet med detektering av momentmotstand med compliance noden var at terskelmomentet for å utløse sikkerhetsfunksjonen "Resisted" var på 5Nm. Det blir beskrevet i artikkel [24] at anbefalt applikasjons momentet (lb/inch) for å fastskru en kork er 50% av diameteren til korken. I dette tilfellet er korken 30mm i diameter, og dette medfører at anbefalt applikasjonsmoment vil være 15 $lb/inch \approx 1.7Nm$. Som nevnt tidligere så bør momenttestingen være 40-60% av applikasjonsmomentet, da vil anbefalt momenttest være på ca 0.85Nm. Hvis NOS AS ønsker å implementere denne roboten for momenttesting, bør en sensor egnet for momenttesting installeres i sammenheng med "maksinert chuck" som vist i seksjon 2.4 figur 2.30

Flaskeregulering
For at roboten skal kunne teste korkmomentet til en flaske, behøves det at flasken fastspennes. For at fastspenningssylinderene(holdesylinder) ikke skal kollidere med flasker som står tett i kø, må det implementeres flaskestoppere for flaskeregulering som slipper inn en og en flaske. Posisjonsplasseringen for disse sylinderene er fremstillet i innledende prosessbilde i seksjon 2.4. Det vil også være nødvendig med en flaskestopper som stopper flaskene for holdesylinderene. Denne flaskestopperen må bli styret av en reflektiv sensor (RSR).

3.2 Veiesekvens

I rapporten har det blitt fremstillet en metode for signalbehandling, kalibrering og selve PLS programmering for detekteringen av feil innholdsmengde i flasker. Alt dette er basert på antagelser og reint teoretiske prinsipp. Hvis man vil oppnå et system som faktisk fungerer på produksjonslinjen til NOS, bør systemet for veiesekvensen bli dimensjonert i henhold til produksjonslinjens forhold. En måte å oppnå et robust system for dynamisk veiing, kan være å implementere ferdig integrerte veiiesystem som fremstillet i denne linken [23]. De intergrerte veiiesystemene kobles bare inn som et ekstra ledd for produksjonslinjer.

3.3 Bildeanalyse

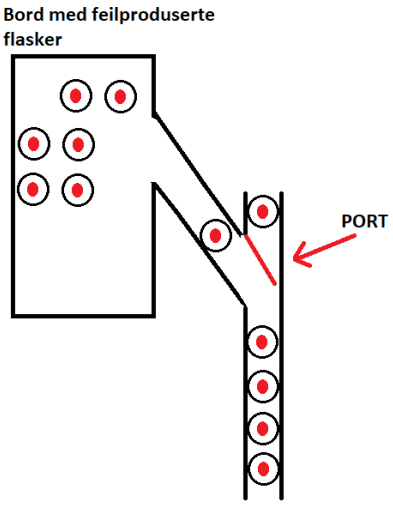
Som vist i vedlagt video i seksjon 2.6 om bildeanalyse, ble det brukt et spesielt oppsett for simulering av passerende flasker som ble bildeanalysert. Dette oppsettet forsikret at flaskene ville passere smartkameraet på nøyaktig samme sted for å forhindre endringer i bildeperspektivet. Hvis man har relativt strenge terskelverdier for inspeksjonfunksjonene i smartkameraet, vil veldig små forandringer i distansen mellom flaske og smartkamera medføre inkonsistente bildeanalysevurderinger. Hvis dette blir et problem for NOS, kan det være hensiktsmessig og innstallere en innsnevring som forsikrer at flasker passerer med konsistent distanse mellom flaske og smartkamera. NOS har nå to smartkameraer av samme type installert i prosessen som står på hver sin side av transportbandet for å dekke 360 graders synsvinkel. Det er antatt at tidligere funksjonalitet for disse smartkameraene var bildeanalyse av flaskekork fra begge sider, men basert på resultat fra labtesting, vil det ikke være nødvendig med mer en et kamera for bildeanalyse. Det kan da være hensiktsmessig og flytte det ene smartkameraet til en annen produksjonslinje.

3.4 OPC UA

Oppgaven har presentert oppkobling og resultater knyttet til OPC UA kommunikasjon på to ulike PLSer. OPC UA modulen som er integrert i NX102-1000 gjør selve oppkoblingen enkel og intuitiv sammenlignet med oppkoblingen for CP1L. Siden det krever mer utstyr for å oppkoble CP1L for OPC UA så vil dette kunne være mer ustabilt. Under oppkobling av CP1L dukket det noen ganger opp feilmeldinger i Python programmet som resulterte i at oppkoblingen måtte gjøres på ny og at PLS måtte restartes for at oppkoblingen skulle fungere. Når oppkoblingen var etablert så virket statusoppdateringen mellom PLS og datamaskin. Endret statusverdi på PLSens innganger ble registrert i UaExpert, uten bemerkelsesverdig etterslep i tid.

3.5 Videreutvikling

For videreutvikling av produksjonslinjen, vil det være anbefalt å installere et automatisk sortertingssystem for feilproduksjon. Med et automatisk sorteringssystem, trenger man ikke å stoppe prosessen for å fjerne feilproduserte flasker. Flasker som ikke er godkjente vil automatisk bli omdirigert ut til siden på et eget bord. Et system som dette kan lett implementeres med å installere en servomotor/penumatisk styrt "Port" som alternerer retningen på flasker ut til sidebordet. Se figur 3.1 for grov skissering over tenkt virkemåte



Figur 3.1: Tenkt skisse av sorteringssystem

Kapittel 4

Konklusjon

I denne rapporten har det blitt presentert løsningsforslag for de ulike tilleggsfunksjonaliteter som NOS ønsker å implementere. Det har vert en krevende prosess å gjennomføre, men resultatene fremvist i rapporten viser mulige oppgraderinger som kan implementeres i prosessen.

Gjennom de simuleringer og labtester som er blitt gjort har vi kommet frem til følgende konklusjoner om prosessen og de ulike verktøy og komponenter som kan bli implementert.

Fyllesekvens og Korksekvens

Basert Sysmac Studio simuleringer så har implementert program tilsvarende samme funksjonalitet som eksisterende program for flaske -og korksekvensen som produksjonslinjen hos NOS. Det eneste som avviker er sylinder timerene for flaskereguleringen i sekvensene. Disse timerene må finjusteres i praksis for å oppnå ønsket timerverdier.

Korkmomenttest

Som vist i vedlagt video av reslutatet for korkmomenttest i seksjon 2.4, kan det konkluderes at roboten har potensialet for utførsel av ønsket oppgaver. For å realisere dette i praksis bør man implementere holde -og stoppesylindere, bytte verktøy og installere en momentsensor.

Veiesekvens

Basert på labtesting har det vist seg at dynamisk veiing med bruk av lastcelle vil kunne være en robust metode for verifisering av flaskens innholdsmengde i produksjon. Lastcellen gir rask respons og nøyaktig måleresultat hvis det analoge signalet blir riktig behandlet. Signalfiltreringsmetoder for måling av flaskevekten kan videre optimaliseres for en raskere respons i praksis hvis en raskere prosess er ønsket/-nødvendig.

Bildebehandling

Bildeanalyse av flaskens kork under bevegelse ga konsistente tilbakemeldinger til PLSen. Det var ingen mulige korkposisjoner utenfor ønsket posisjon som ga falsk positiv tilbakemelding. Man kan basert på labtester utført i dette prosjektet konkludere at kun et smartkamera vil være nødvendig for bildeanalyse gitt at det er nå installert to smartkameraer for å dekke begge sider av flaskekorken i prossessen hos NOS.

OPC UA

OPC UA kommunikasjon har vist seg å fungere fint for å dele statusinformasjon av prosessen ut til eksterne rom. For lettest å etablere kommunikasjonen og mest robust oppkobling anbefales alternativet med bruk av NX102 fremfor eksisterende PLS. Det er likevell vist gjennom rapporten at oppkoblingen fungerer på begge PLSer.

Bibliografi

[1] Cap tightener. URL https://www.kinexcappers.com/ls-cap-tightener/accessories.htm.

[2] *Software Manual TMflow*. URL https://assets.omron.com/m/66721cde51512bbc/original/TM-Software-Manual-TM-Flow.pdf.

[3] Lastcelle wiki. URL https://en.wikipedia.org/wiki/Load_cell.

[4] *CP1L-EL/EM CPU Unit*, 2014. URL https://assets.omron.eu/downloads/manual/en/v59/w516_cp1l-el,_cp1l-em_cpu_unit_operation_manual_en.pdf.

[5] *Programmable Terminal NA-series Software*, 2014. URL https://assets.omron.eu/downloads/manual/en/v118_na_series_programmable_terminal_software_users_manual_en.pdf.

[6] *Load Cell Input Unit Startup Guide for Weight Measurement*, 2016. URL https://assets.omron.eu/downloads/manual/en/v1/p104_nx_series_load_cell_input_unit_setup_manual_en.pdf.

[7] *Machine Automation Controller NX-series Load Cell Input Unit*, 2017. URL https://assets.omron.eu/downloads/manual/en/v3/w565_nx-series_load_cell_input_unit_users_manual_en.pdf.

[8] *Smart Camera FQ2-S/CH Series*, 2017. URL https://assets.omron.eu/downloads/manual/en/v4/z337_fq2-s_ch_smart_camera_users_manual_en.pdf.

[9] *Load Cell Input Unit*, 2017. URL https://assets.omron.eu/downloads/manual/en/v3/w565_nx-series_load_cell_input_unit_users_manual_en.pdf.

[10] *Collaborative Robot Hardsware Installation Manual*, 2018. URL https://assets.omron.eu/downloads/manual/en/v2/i623_collaborative_robots_hardware_installation_manual_en.pdf.

[11] *ROBOTIQ*, 2018. URL https://assets.robotiq.com/website-assets/support_documents/document/2F-85_2F-140_Instruction_Manual_e-Series_PDF_20190206.pdf.

[12] *Omron support, Application Library*, 2018. URL https://omrongroup.sharepoint.com/sites/ext_Mtm_OMCE_IAB_0002/Shared%20Documents/1%20Produkter/CPU%20og%20HMI/NY5%20og%20NJ%20og%20NA%20og%20NX/Funksjonsblokker/OEN_Robot/OEN_Robot%20Manual.pdf.

[13] *CX-Programmer Ver. 9*, 2019. URL https://assets.omron.eu/downloads/manual/en/v6/w446_cx-programmer_operation_manual_en.pdf.

[14] *Machine Automation Controller NJ/NX-series CPU Unit Built-in EtherNet/IPTM Port*, 2019. URL https://assets.omron.eu/downloads/manual/en/v3/w506_nx_nj-series_cpu_unit_built-in_ethernet_ip_port_users_manual_en.pdf.

[15] *Raspberry Pi 4 Model B*, 2019. URL https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi_DATA_2711_1p0_preliminary.pdf.

[16] *Sysmac Studio Version 1*, 2020. URL https://assets.omron.eu/downloads/manual/en/v4/w504_sysmac_studio_operation_manual_en.pdf.

[17] *Collaborative Robots TM5*, 2020. URL https://assets.omron.eu/downloads/datasheet/en/v5/i837_collaborative_robots_datasheet_en.pdf.

[18] *Omron support, Start!TMCobot*, 2021. URL https://omrongroup.sharepoint.com/sites/ext_Mtm_OMCE_IAB_0002/_layouts/15/search.aspx/siteall?q=Start!tmcobot&scope=site. Hentet 29.04.2021.

[19] Elteco. Induktive givere, 2021. URL https://www.elteco.no/induktive. Hentet 21.04.2021.

[20] Gyldendal. Tekniske tjenester, 2021. URL http://web2.gyldendal.no/undervisning/felles/pixdir20/?archive=tip_tt&menuitem=menu_3_5&resultsource=menu_3_5&detailsource=image_294. Hentet 23.04.2021.

[21] JP Viskovic. Modbus TCP Server for CP1L series, 2019. URL https://www.support-omron.fr/telechargements/documentations/2017-03-09%20-%2011-57-06%20-%201394359222/MTCP_CP1L_Server_E.pdf. Hentet 13.04.2021.

[22] MathWorks. Matlab, 2021. URL https://se.mathworks.com/help/matlab/. Hentet 10.05.2021.

[23] Mettler toledo. Dynamic Parcel Weighing. URL https://www.mt.com/int/en/home/products/Transport_and_Logistics_Solutions/Dynamic_Parcel_Weighing.html. Hentet 29.04.2021.

[24] Oberk.com. How To Make Sure Your Cap Is Tight Enough?, 2015. URL https://www.oberk.com/packaging-crash-course/quick-question-monday-how-to-make-sure-your-cap-is-tight-enough. Hentet 29.04.2021.

[25] Omron. User's Manual for communication settings, 2020. URL https://assets.omron.eu/downloads/manual/en/z338_fq2-s_ch_smart_camera_for_communication_settings_users_manual_en.pdf. Hentet 15.04.2021.

[26] Omron. Omron strukturvariabler for ethernet kommunikasjon, 2020. URL https://omrongroup.sharepoint.com/sites/ext_Mtm_OMCE_IAB_0002/SitePages/Home.aspx. Hentet 16.04.2021.

[27] Omron Industrial Automation EMEA. How to configure opc-ua on sysmac nx-102 controllers, 2020. URL https://www.youtube.com/watch?v=JiJK1pU9XMU. Hentet 13.04.2021.

[28] Omron.eu. FQ2-S4, 2018. URL https://industrial.omron.eu/en/products/fq2-s4. Hentet 12.05.2021.

[29] OPC Foundation. Unified Architecture, 2019. URL https://opcfoundation.org/about/opc-technologies/opc-ua/. Hentet 14.04.2021.

[30] OPC Foundation. Unified Architecture, 2019. URL https://opcfoundation.org/about/opc-technologies/opc-ua/. Hentet 13.05.2021.

[31] Project Plc. Cp1l e modbus tcp server, 2019. URL https://www.youtube.com/watch?v=Ek_M8IWMsI0. Hentet 13.04.2021.

[32] Sofi Marffy. OPC and OPC UA explained, 2020. URL https://www.novotek.com/uk/solutions/kepware-communication-platform/opc-and-opc-ua-explained/. Hentet 14.04.2021.

[33] Tektron.ie. Diffuse Mode Sensing, 2021. URL http://www.tektron.ie/diffuse.htm. Hentet 22.04.2021.

[34] Unified Automation. UaExpert Documentation, 2021. URL http://documentation.unified-automation.com/uaexpert/1.5.1/html/index.html. Hentet 10.05.2021.

[35] Unified Automation. Session, 2021. URL https://documentation.unified-automation.com/uasdkdotnet/3.0.2/html/L2ClientSdkSession.html. Hentet 14.04.2021.

[36] Variohm. *Load Cell - AL6B Series*, 2021. URL https://www.distrelec.biz/Web/Downloads/_t/ds/variohm_AL6B_eng_tds.pdf. Funnet 16.01.2021.

[37] Wikipedia. OPC Unified Architecture, 2021. URL https://en.wikipedia.org/wiki/OPC_Unified_Architecture. Hentet 14.04.2021.

[38] Wikipedia Contributors. Modbus, 2021. URL https://en.wikipedia.org/wiki/Modbus. Hentet 11.05.2021.

[39] Wikipedia Contributors. Finite impulse response, 2021. URL https://en.wikipedia.org/wiki/Finite_impulse_response#Moving_average_example. Hentet 14.05.2021.

Vedlegg A

Program

A.1 Program Fyllesekvens

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 4/25/2021 10:30:43 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

1-9-1.Programs

1-9-1-1.fyllesekvens

1-9-1-1-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
tilstand	INT			False	False	
timer_flaskestopper	ton			False	False	timer for flaskestopper
ikke_i_bruk	TIME			False	False	bare en "filler" variabel
timer_flasker_inn	ton			False	False	timer for 6 flasker inn
timer_flasker_ute	BOOL			False	False	
Start_timer_flasker_inn	BOOL			False	False	
timer_flasker_inne	BOOL			False	False	
timer_flaskekø	ton			False	False	timer for flaskekø sjekk
timer_dryppfanger_ute	BOOL			False	False	
timer_dryppfanger	ton			False	False	
start_timer_dryppfanger	BOOL			False	False	
start_timer_flaskestopper	BOOL			False	False	
timer_flaskestopper_ferdig	BOOL			False	False	
teller_flaske_ut	CTU			False	False	
teller_flasker_ute	BOOL			False	False	
ikke_i_bruk2	INT			False	False	
Start_timer_flasker_ut	BOOL			False	False	
timer_flasker_ut	ton			False	False	
VAR_EXTERNAL						
DRYPPFANGER	BOOL				False	Dryppfanger
RS1	BOOL				False	Reflektiv sensor 1
LOFTE_SYL_SENK	BOOL				False	Senker dysene

	FYLLE_SYL_FYLL	BOOL				False	presser ut vesken av doseringskammer
	LOFTE_SYL_HEV	BOOL				False	hever dysene
	FYLLE_SYL_SUG	BOOL				False	Fylling av doesringskammer
	FS1	BOOL				False	Flaskestopper 1
	IS5	BOOL				False	Induktiv sensor 5
	FS2	BOOL				False	Flaskestopper 2
	IS4	BOOL				False	Induktiv sensor 4
	IS6	BOOL				False	Induktiv sensor 6
	IS3	BOOL				False	Induktiv sensor 3
	IS7	BOOL				False	Induktiv sensor 7
	RS3	BOOL				False	Reflektiv sensor 3
	FlaskeKø	BOOL				False	Flaskekø
	Transportband	BOOL				False	Transportband på
	RS2	BOOL				False	Reflektiv sensor 2

1-9-1-1-2.Program Body

```
1       Transportband:=TRUE;
2
3       timer_flaskestopper(In:=start_timer_flaskestopper, PT:=T#0.5s, Q=>timer_flaskestopper_ferdig,
ET=>ikke_i_bruk);
4       timer_dryppfanger(In:=start_timer_dryppfanger, PT:=T#0.5s, Q=>timer_dryppfanger_ute, ET=>ikke_i_bruk);
5       timer_flasker_ut(In:=Start_timer_flasker_ut, PT:=T#3s, Q=>timer_flasker_ute, ET=>ikke_i_bruk);
6       timer_flasker_inn(In:=RS1, PT:=T#3s, Q=>timer_flasker_inne, ET=>ikke_i_bruk);
7       timer_flaskekø(In:=RS3, PT:=T#5s, Q=>FlaskeKø, ET=>ikke_i_bruk);
8       teller_flaske_ut(CU:=RS2, Reset:=teller_flasker_ute, PV:=6, Q=>teller_flasker_ute, CV=>ikke_i_bruk2);
9
10
11
12      CASE tilstand OF
13
14          0: (*Start tilstand, trenger ingen betingelser for å gå inn i neste tilstand*)
15                  tilstand := 1;
16
17          1: (*flaskestopper 2 ut for å stoppe flaskene inn og doseringskammeret fylles opp*)
18                  FS2:=TRUE;
19                  FYLLE_SYL_SUG := TRUE;
20                  IF timer_flasker_inne AND IS5 THEN
21                          tilstand := 2;
22                  END_IF;
23
24          2: (*flaskestopper 1 stopper flaskene etter 6 stk har kommet inn*)
25                  FS1 := TRUE;
26                  start_timer_flaskestopper := TRUE;
27                  IF timer_flaskestopper_ferdig           THEN
28                          start_timer_flaskestopper := FALSE;
29                          tilstand := 3;
30                  END_IF;
31
32          3: (*dryppfanger ut for å gi plass til fylledysene*)
33                  DRYPPFANGER := TRUE;
34                  start_timer_dryppfanger := TRUE;
35                  IF timer_dryppfanger_ute THEN
36                          start_timer_dryppfanger := FALSE;
37                          tilstand := 4;
38                  END_IF;
39
40          4:(*senker fylledysene inn i flaskene*)
41                  LOFTE_SYL_SENK :=TRUE;
42                  IF IS4 THEN
43                          LOFTE_SYL_SENK := FALSE;
44                          tilstand := 5;
45                  END_IF;
46
47          5: (*fylling*)
48                  FYLLE_SYL_FYLL := TRUE;
49                  IF IS6 THEN
50                          FYLLE_SYL_FYLL := FALSE;
51                          tilstand := 6;
52                  END_IF;
```

```
53
54      6: (*løfter fylledysene ut av flaskene*)
55              LOFTE_SYL_HEV := TRUE;
56              IF IS3 THEN
57                      LOFTE_SYL_HEV := FALSE;
58                      tilstand := 7;
59              END_IF;
60
61      7: (*dryppfanger inn*)
62              DRYPPFANGER := FALSE;
63              IF FlaskeKø=FALSE THEN
64                      IF IS7 THEN
65                              tilstand := 8;
66                      END_IF;
67              END_IF;
68
69      8:(*flaskestopper nr 2 inn for å sleppe flaskene videre*)
70              FS2 := FALSE;
71              Start_timer_flasker_ut := TRUE;
72              IF timer_flasker_ute THEN
73                      Start_timer_flasker_ut := FALSE;
74                      tilstand := 9;
75              END_IF;
76
77      9:(*flaskestopper nr 1 slepper inn 6 nye flasker for fylling*)
78              FS1:=FALSE;
79              IF teller_flasker_ute THEN
80                      tilstand := 0;
81              END_IF;
82  END_CASE;
```

A.2 Program Korksekvens

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 4/26/2021 12:04:44 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

1-9-1.Programs

1-9-1-2.Korksekvens

1-9-1-2-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
Flaskestopper_timer	TON			False	False	Flaskestopper timer som blir fremstillet som "t=0.5" i diagram
Flaskestopper_timer_start	BOOL			False	False	
Flaskestopper_timer_ferdig	BOOL			False	False	
ikkeibruk	TIME			False	False	En "Filler" variabel
korkmaskin_ned_timer_start	BOOL			False	False	
korkmaskin_ned_timer_ferdig	BOOL			False	False	
korkmaskin_timer_ned	TON			False	False	Timer for korkmaskin ned, fremstillet som "t=1s" i diagram
korkmaskin_timer_spinn	TON			False	False	Timer for korkmaskin spinn, femstillet som "t=1s" i diagram
korkmaskin_spinn_ferdig	BOOL			False	False	
korkmaskin_spinn_timer_start	BOOL			False	False	
korkmaskin_opp_timer_start	BOOL			False	False	
korkmaskin_opp_timer_ferdig	BOOL			False	False	
korkmaskin_timer_opp	TON			False	False	Timer for korkmaskin opp, fremstillet som "t=1s" i diagram
Timer_flaske_ut	TON			False	False	Timer for flaske ut, fremstillet som "t=2s" i diagram
Flaske_ut_timer	BOOL			False	False	

	Flaske_ute	BOOL			False	False	
	korksekvenstilstand	INT			False	False	
	Timer_mangel_på_kork	TON			False	False	Timer for mangel på kork sjekk
VAR_EXTERNAL							
	FS4	BOOL				False	Flaskestopper 4
	RS4	BOOL				False	Reflektiv sensor 4
	FS3	BOOL				False	Flaskestopper 3
	Korkmaskin_ned_opp	BOOL				False	Korkmomentmaskin, impulsstyrt [1]ned og [2] opp
	Korkmaskin_Spinn	BOOL				False	Egent utgang i PLS for styring av korkmomentmaksin spinn
	RS3	BOOL				False	Reflektiv sensor 3
	Mangel_paa_kork	BOOL				False	Global variabel for mangel på kork feilmelding
	RS6	BOOL				False	Reflektiv sensor 6

1-9-1-2-2.Program Body

```
1
2       Flaskestopper_timer(In:=Flaskestopper_timer_start, PT:=T#0.5s, Q=>Flaskestopper_timer_ferdig, ET=>ikkeibruk);
3       korkmaskin_timer_ned(In:=korkmaskin_ned_timer_start, PT:=T#1s, Q=>korkmaskin_ned_timer_ferdig,
ET=>ikkeibruk);
4       korkmaskin_timer_spinn(In:=korkmaskin_spinn_timer_start, PT:=T#1s, Q=>korkmaskin_spinn_ferdig,
ET=>ikkeibruk);
5       korkmaskin_timer_opp(In:=korkmaskin_opp_timer_start, PT:=T#1s, Q=>korkmaskin_opp_timer_ferdig,
ET=>ikkeibruk);
6       Timer_flaske_ut(In:=Flaske_ut_timer, PT:=T#2s, Q=>Flaske_ute, ET=>ikkeibruk);
7       Timer_mangel_på_kork(In:=RS6, PT:=T#10s, Q=>Mangel_paa_kork, ET=>ikkeibruk);
8
9       CASE korksekvenstilstand OF
10
11          0: (*Registrerer at flasker er på vei*)
12                  IF RS3 THEN
13                          korksekvenstilstand :=1;
14                  END_IF;
15
16          1:(*Flaskestopper 4 ut og venter til RS4 registrerer flaske*)
17                  FS4 := TRUE;
18                  IF RS4 THEN
19                          korksekvenstilstand := 2;
20                  END_IF;
21
22          2: (*Flaskestopper 3 ut for å skille ut flaskene*)
23                  FS3 := TRUE;
24                  Flaskestopper_timer_start := TRUE;
25                  IF Flaskestopper_timer_ferdig THEN
26                          Flaskestopper_timer_start := FALSE;
27                          korksekvenstilstand := 3;
28                  END_IF;
29
30          3: (*Sjekker om det er mangel på korker. Hvis ikke mangel på korker, korkmomentmaskin ned*)
31                  IF Mangel_paa_kork=FALSE THEN
32                          Korkmaskin_ned_opp :=TRUE;
33                          korkmaskin_ned_timer_start := TRUE;
34                          IF korkmaskin_ned_timer_ferdig THEN
35                                  korkmaskin_ned_timer_start := FALSE;
36                                  Korkmaskin_ned_opp :=FALSE;
37                                  korksekvenstilstand :=4;
38                          END_IF;
39                  END_IF;
40
41          4: (*Spinner verktøyet for stramming av kork*)
42                  Korkmaskin_Spinn := TRUE;
43                  korkmaskin_spinn_timer_start := TRUE;
44                  IF korkmaskin_spinn_ferdig THEN
45                          korkmaskin_spinn_timer_start := FALSE;
46                          Korkmaskin_Spinn := FALSE;
47                          korksekvenstilstand := 5;
48                  END_IF;
49
50          5: (*Korkmomentmaskin opp*)
```

```
51                  Korkmaskin_ned_opp :=TRUE;
52                  korkmaskin_opp_timer_start := TRUE;
53                  IF korkmaskin_opp_timer_ferdig THEN
54                          korkmaskin_opp_timer_start := FALSE;
55                          Korkmaskin_ned_opp := FALSE;
56                          korksekvenstilstand := 6;
57                  END_IF;
58
59          6: (*Flaskestopper 4 slipper ut flasken*)
60                  FS4 := FALSE;
61                  Flaske_ut_timer := TRUE;
62                  IF Flaske_ute THEN
63                          Flaske_ut_timer :=FALSE;
64                          korksekvenstilstand := 7;
65                  END_IF;
66
67          7:  (*Flaksestopper slipper inn ny flaske. Grunnet ingen betingelser blir programmet omdirrigert til tilstand 1
veldig fort*)
68                  FS3 := FALSE;
69                  korksekvenstilstand := 1;
70
71      END_CASE;
```

A.3 Program KorkmomentTest

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 5/11/2021 1:06:21 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

1-9-1.Programs

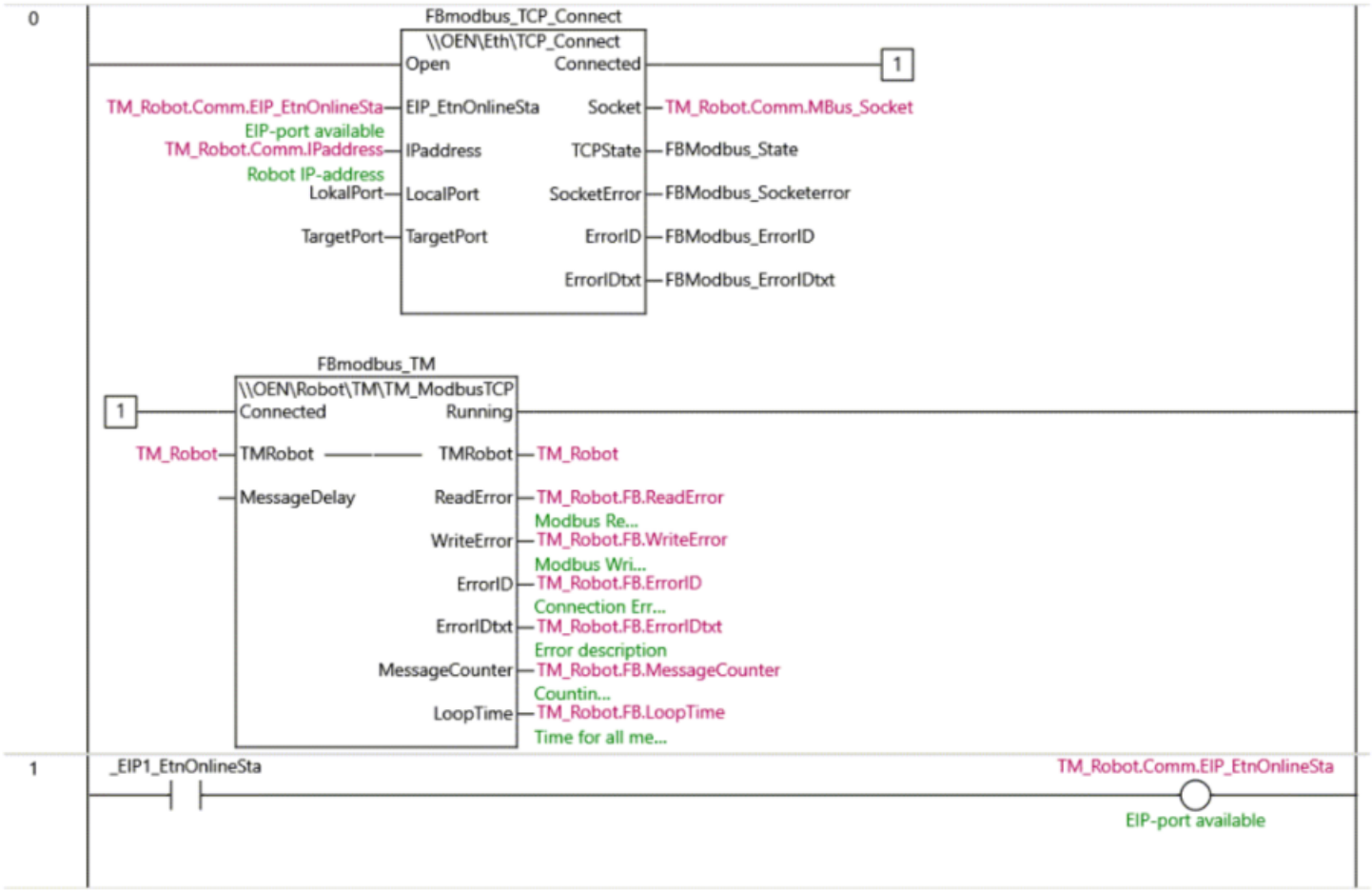
1-9-1-3.MomentTestsekvens

1-9-1-3-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
LokalPort	UINT	0		False	False	
TargetPort	UINT	502		False	False	
FBmodbus_TCP_Connect	OEN\Eth\TCP_Connect			False	False	
FBmodbus_TM	OEN\Robot\TM\TM_ModbusTCP			False	False	
FBModbus_State	_eCONNECTION_STATE			False	False	
FBModbus_Socketerror	BOOL			False	False	
FBModbus_ErrorID	WORD			False	False	
FBModbus_ErrorIDtxt	STRING[100]			False	False	
Teller_Korkmomentfeil	Teller			False	False	
TM_RobotStickfunksjoner	TM_Stickfunksjoner			False	False	
_EIP1_EtnOnlineSta	BOOL			False	False	
antall_flasker_momentTestet	INT			False	False	
flaskeregulering	INT			False	False	
Flaskestopperstimer	TON			False	False	
flaskestoppertimer_start	BOOL			False	False	
flaskestoppertimer_ferdig	BOOL			False	False	
korkmomenttest_ferdig	BOOL			False	False	
Reflektiv_sensor_timer	ton			False	False	
Flaskedetektert_RS7	BOOL			False	False	
Relfeltiv_sen	ton			False	False	

	sor_timer2						
	flaskesjekk	BOOL			False	False	
	flaske_ikke_detektert	BOOL			False	False	
	ikkeibruk	TIME			False	False	
VAR_EXTERNAL							
	TM_Robot	OEN\nRobot\nTM\sRobot				False	
	Reset_korkmomentfeil	BOOL				False	
	TM_Error_text	String[256]				False	
	Korkfeil	INT				False	
	PlayPause	BOOL				False	
	Stop_TM	BOOL				False	
	Pluss	BOOL				False	
	Minus	BOOL				False	
	ToggleMode	BOOL				False	
	ProgramSpeed	INT				False	
	CurrentMode	OEN\nROBOT\nTM\eMODE				False	
	korkfeil_prosent	REAL				False	
	RS7	BOOL				False	
	FS6	BOOL				False	
	FS7	BOOL				False	
	FS5	BOOL				False	

1-9-1-3-2.Section0



2 P_First_Run

1

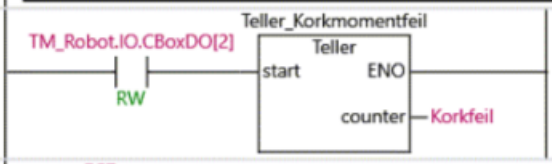
1

```
1  //Activate Modbus Reading and Writing
2  TM_Robot.DataSelect.Coordinates_Reg1B59thru1B94:= TRUE; // Activate reading of
   coordinates
3  TM_Robot.DataSelect.Others_Reg1CACthru1CB7 := TRUE;      // Activate reading of
   general signals
4  TM_Robot.DataSelect.Status := TRUE;                      // Activate reading of
   Status signals
5  TM_Robot.DataSelect.ErrorCode := TRUE;                   // Activate reading of
   Last Error Code
6  TM_Robot.DataSelect.Stick := TRUE;                       // Activate reading of
   Speed and Mode
7  TM_Robot.DataSelect.Light := TRUE;                       // Activate reading of
   Light Color
8  TM_Robot.DataSelect.UserINT := TRUE;                     // Activate reading of
   RegisterOutput#9000-9063 and Writing of RegisterOutput#9100-9163
9  TM_Robot.DataSelect.UserBOOL := TRUE;                    // Activate reading of
   RegisterOutput#9000-9063 and Writing of RegisterOutput#9100-9163
10 TM_Robot.DataSelect.UserFloat := TRUE;                   // Activate reading of
   RegisterOutput#9000-9463 and Writing of RegisterOutput#9100-9663
11 TM_Robot.DataSelect.CBoxDI := TRUE;                      // Activate reading of
   reading of ControlBox Input signals Status
12 TM_Robot.DataSelect.CBoxDO := TRUE;                      // Activate reading of
   Outout signals Status
13
14
15 //More Modbus Data avilable below if needed, but each one will slighty reduce
   comm response
16 //if you set all to TRUE, the total resposetime is about 0.3s using 1ms CycleTime
17 TM_Robot.DataSelect.CBoxAI := FALSE;
18 TM_Robot.DataSelect.CBoxAO := FALSE;
19 TM_Robot.DataSelect.EndAI := FALSE;
20 TM_Robot.DataSelect.EndDI := FALSE;
21 TM_Robot.DataSelect.EndDO := FALSE;
22 TM_Robot.DataSelect.Inertia_MassCenter := TRUE;
23 TM_Robot.DataSelect.JointTorque := TRUE;
24 TM_Robot.DataSelect.TouchStop := FALSE;
25 TM_Robot.DataSelect.CollabMode := TRUE;
26 TM_Robot.DataSelect.SafetyStopCriteria := TRUE;
```

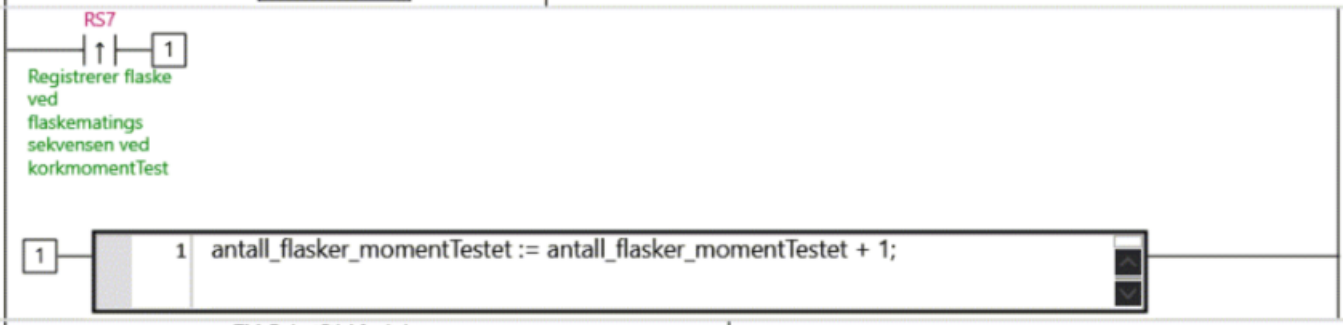
3

```
IF TM_Robot.Status.Error THEN
    TM_Error_text := TM_Robot.Status.ErrorCodeTxt;
END_IF;

IF Reset_korkmomentfeil THEN
    Korkfeil := 0;
    korkfeil_prosent := 0;
END_IF;

IF TM_Robot.IO.CBoxDO[13] OR TM_Robot.Button.Stick_Stop THEN
    TM_Robot.IO.CBoxDO[2] := FALSE;
END_IF;

korkfeil_prosent := (Korkfeil/antall_flasker_momentTestet)*100;
//Play_Pause := TM_Robot.Stick.Play_Pause;
```

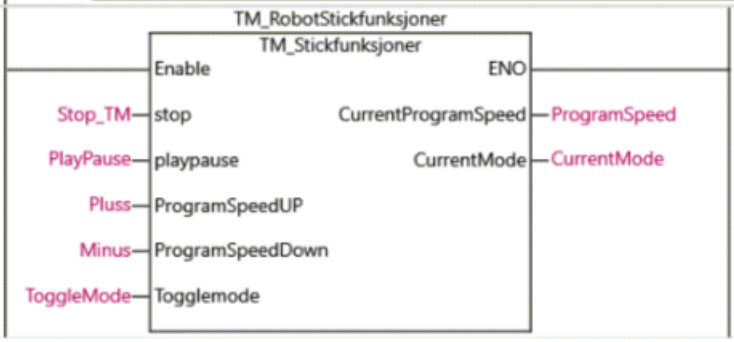
4



5



6



7

```
1
2  Flaskestopperstimer(In:=flaskestoppertimer_start, PT:=T#0.5s,
   Q=>flaskestoppertimer_ferdig, ET=>ikkeibruk);
3
4  IF flaskeregulering=7 THEN
5      Reflektiv_sensor_timer(In:=RS7, PT:=T#0.5s, Q=>Flaskedetektert_RS7,
   ET=>ikkeibruk);
6  END_IF;
7
8  Relfeltiv_sensor_timer2(In:=flaskesjekk, PT:=T#3s, Q=>flaske_ikke_detektert,
   ET=>ikkeibruk);
9
10 CASE flaskeregulering OF
11
12     0:
13         IF RS7 THEN
14             flaskeregulering := 1;
15         END_IF;
16
17     1:
18         FS6 := TRUE;
19         FS7 := TRUE;
20         flaskestoppertimer_start := TRUE;
21         IF flaskestoppertimer_ferdig THEN
22             flaskestoppertimer_start := FALSE;
23             flaskeregulering := 2;
24         END_IF;
25
26     2:
27         FS5 := TRUE;
28         FS6 := FALSE;
29         flaskestoppertimer_start := TRUE;
30         IF flaskestoppertimer_ferdig THEN
31             flaskestoppertimer_start := FALSE;
32             flaskeregulering := 3;
33         END_IF;
34
35     3:
36         FS5 := FALSE;
37         FS6 := TRUE;
38         flaskestoppertimer_start := TRUE;
39         IF flaskestoppertimer_ferdig THEN
40             flaskestoppertimer_start := FALSE;
41             flaskeregulering := 4;
42         END_IF;
43
44     4:
45         FS5 := TRUE;
46         IF korkmomenttest_ferdig THEN
47             flaskeregulering := 5;
48         END_IF;
49
50     5:
51         FS6:= FALSE;
52         FS7 := FALSE;
53         flaskestoppertimer_start := TRUE;
54         IF flaskestoppertimer_ferdig THEN
55             flaskestoppertimer_start := FALSE;
56             flaskeregulering := 6;
```

```
        flaskeregulering := 0;
    END_IF;

6:
    FS5 := FALSE;
    FS6:= TRUE;
    FS7 := TRUE;
    IF korkmomenttest_ferdig THEN
        flaskeregulering := 7;
    END_IF;

7:
    FS7 := FALSE;
    flaskesjekk := TRUE;
    IF RS7 AND Flaskedetektert_RS7 THEN
        flaskesjekk := FALSE;
        flaskeregulering := 8;
    ELSIF flaske_ikke_detektert THEN
        flaskesjekk := FALSE;
        flaskeregulering := 0;
    END_IF;

8:
    FS7:=TRUE;
    flaskestoppertimer_start:=TRUE;
    IF flaskestoppertimer_ferdig THEN
        flaskestoppertimer_start :=FALSE;
    END_IF;
END_CASE;
```

A.4 Program Veiesekvens

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 4/26/2021 6:18:18 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

1-9-1.Programs

1-9-1-4.Veiesekvens

1-9-1-4-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
timer_stabilisering_start	BOOL			False	False	timer start for stabilisering (Ts)
timer_stabilisering_ferdig	BOOL			False	False	timer ferdig for stabilisering (Ts)
vektmaaling	INT			False	False	Case tilstandsvariabel
antall_målinger	INT			False	False	Hvor mange målinger som er gjort (1000 per sek)
timer_flaskemåling_start	BOOL			False	False	timer start for flaskemåling (Tm)
timer_flaskemåling_ferdig	BOOL			False	False	timer ferdig for flaskemåling (Tm)
ikkeibruk1	UINT			False	False	"filler" variabel
ikkeibruk2	UINT			False	False	"filler" variabel
ikkeibruk3	_sTimer			False	False	"filler" variabel
ikkeibruk4	_sTimer			False	False	"filler" variabel
Flaske_vekt_test_summering	REAL			False	False	Summen av alle målingene som er tatt i intervallet (Tm)
Flaske_på	BOOL			False	False	boolsk verdi for om flasken er på veiecellen
dysenummer	INT	6		False	False	dysenummer (1-6) teller fra høyre til venstre på prosesstegning
Flaske_vekt_test	REAL			False	False	Flaskevekt resultatet som testes opp mot toleransegrensen
dysenummer1_antall_fyllinger	INT			False	False	
dysenummer6_antall_fyllinger	INT			False	False	
dysenummer5_antall_fyllinger	INT			False	False	
dysenummer4_antall_fyllinger	INT			False	False	

dysenummer3_antall_fyllinger	INT			False	False	
dysenummer2_antall_fyllinger	INT			False	False	
VAR_EXTERNAL						
onsket_verdi	REAL				False	
Filtrert_uskalert_vekt_kg	REAL				False	
Filtrert_skalert_vekt_kg	REAL				False	
skalert_vekt_kg	REAL				False	
nivåfeil_dysenummer5_prosent	REAL				False	
nivåfeil_dysenummer4_prosent	REAL				False	
nivåfeil_dysenummer3_prosent	REAL				False	
nivåfeil_dysenummer2_prosent	REAL				False	
nivåfeil_dysenummer1_prosent	REAL				False	
antall_flasker_veiet	INT				False	
Flaskenivåfeil	REAL				False	
nivåfeil_dysenummer6_prosent	REAL				False	
nivåfeil_dysnummer1	REAL				False	
nivåfeil_dysnummer2	REAL				False	
nivåfeil_dysnummer3	REAL				False	
nivåfeil_dysnummer4	REAL				False	
nivåfeil_dysnummer5	REAL				False	
nivåfeil_dysnummer6	REAL				False	
flaskenivåfeil	REAL				False	

	_prosent						

1-9-1-4-2.Program Body

```
1
2       Timer(In:=timer_stabilisering_start, PT:=14(*=1.4sek*), TimerDat:=ikkeibruk3, Q=>timer_stabilisering_ferdig,
ET=>ikkeibruk1);
3       Timer(In:=timer_flaskemåling_start, PT:=4(*0.4sek*), TimerDat:=ikkeibruk4, Q=>timer_flaskemåling_ferdig,
ET=>ikkeibruk2);
4
5
6       CASE vektmaaling OF
7           0:          (*resetting av variabler, Sjekker om detektert vekt er >0.2kg for sekvens start og Tellervariabler for å
holde styr på dyser og antall flasker*)
8                       Flaske_vekt_test_summering := 0;
9                       antall_målinger := 0;
10                      Flaske_vekt_test := 0.0;
11                      IF skalert_vekt_kg > 0.2 THEN
12                              antall_flasker_veiet := antall_flasker_veiet + 1;
13                              IF dysenummer = 6 THEN (*holder styr på hvor mange ganger de forskjellige dysene har
fyllet opp en flaske (for prosent regningen)*)
14                                      dysenummer6_antall_fyllinger := dysenummer1_antall_fyllinger +1;
15                              ELSIF dysenummer = 5 THEN
16                                      dysenummer5_antall_fyllinger := dysenummer1_antall_fyllinger +1;
17                              ELSIF dysenummer = 4 THEN
18                                      dysenummer4_antall_fyllinger := dysenummer1_antall_fyllinger +1;
19                              ELSIF dysenummer = 3 THEN
20                                      dysenummer3_antall_fyllinger := dysenummer1_antall_fyllinger +1;
21                              ELSIF dysenummer = 2 THEN
22                                      dysenummer2_antall_fyllinger := dysenummer1_antall_fyllinger +1;
23                              ELSIF dysenummer = 1 THEN
24                                      dysenummer1_antall_fyllinger := dysenummer1_antall_fyllinger +1;
25                              END_IF;
26
27                              IF dysenummer = 1 THEN (*Holder styr på hvilken fylledyse som tilhører hvilken flaske*)
28                                      dysenummer := 6;
29                              ELSE
30                                      dysenummer := dysenummer - 1;
31                              END_IF;
32
33                              Flaske_på := TRUE;
34                              vektmaaling := 1;
35                      END_IF;
36
37          1:(*aktiverer timer for forhandsbestemt stabiliseringstid*)
38                      timer_stabilisering_start := TRUE;
39                      IF timer_stabilisering_ferdig THEN
40                              timer_stabilisering_start := FALSE;
41                              vektmaaling := 2;
42                      END_IF;
43
44          2:          (*Aktiverer timer for selve flaskevekt målingen*)
45                      timer_flaskemåling_start := TRUE;
46                      IF timer_flaskemåling_ferdig THEN
47                              timer_flaskemåling_start := FALSE;
48                              vektmaaling := 3;
49                      ELSE
```

```
50                          antall_målinger := antall_målinger +1;
51                          Flaske_vekt_test_summering := Flaske_vekt_test_summering + skalert_vekt_kg;
52                          Flaske_vekt_test := Flaske_vekt_test_summering/antall_målinger;
53                  END_IF;
54
55                  3: (*Verifiserer om flaskevekten er innfor terskelverdiene*)
56                          IF 1.513 < Flaske_vekt_test AND Flaske_vekt_test < 1.575 THEN   (*+/-2% toleransesjekk*)
57                                  vektmaaling := 4;
58                          ELSE
59
60                                  Flaskenivåfeil := Flaskenivåfeil + 1;
61                                  flaskenivåfeil_prosent := (Flaskenivåfeil/antall_flasker_veiet)*100;
62
63                                  IF dysenummer = 1 THEN (*konfigurerer flaskenivåfeilen til respektiv fylledyse*)
64                                          nivåfeil_dysnummer6 := nivåfeil_dysnummer6 + 1;
65                                          nivåfeil_dysenummer6_prosent := (nivåfeil_dysnummer6/
dysenummer6_antall_fyllinger)*100;
66                                  ELSIF dysenummer = 2 THEN
67                                          nivåfeil_dysnummer5 := nivåfeil_dysnummer5 + 1;
68                                          nivåfeil_dysenummer5_prosent := (nivåfeil_dysnummer5/
dysenummer5_antall_fyllinger)*100;
69                                  ELSIF dysenummer = 3 THEN
70                                          nivåfeil_dysnummer4 := nivåfeil_dysnummer4 + 1 ;
71                                          nivåfeil_dysenummer4_prosent := (nivåfeil_dysnummer4/
dysenummer4_antall_fyllinger)*100;
72                                  ELSIF dysenummer = 4 THEN
73                                          nivåfeil_dysnummer3 := nivåfeil_dysnummer3 + 1;
74                                          nivåfeil_dysenummer3_prosent := (nivåfeil_dysnummer3/
dysenummer3_antall_fyllinger)*100;
75                                  ELSIF dysenummer = 5 THEN
76                                          nivåfeil_dysnummer2 := nivåfeil_dysnummer2 + 1;
77                                          nivåfeil_dysenummer2_prosent := (nivåfeil_dysnummer2/
dysenummer2_antall_fyllinger)*100;
78                                  ELSIF dysenummer = 6 THEN
79                                          nivåfeil_dysnummer1 := nivåfeil_dysnummer1 + 1;
80                                          nivåfeil_dysenummer1_prosent := (nivåfeil_dysnummer1/
dysenummer1_antall_fyllinger)*100;
81                                  END_IF;
82
83                                  vektmaaling := 4;
84                          END_IF;
85
86                  4: (*sjekker om flasken har avstiget lastcellen*)
87                          IF skalert_vekt_kg < 0.2 THEN
88                                  Flaske_på := FALSE;
89
90                                  vektmaaling := 0;
91                          END_IF;
92
93
94      END_CASE;
95
96      onsket_verdi := 1.544;
97
```

A.5 Program Bildeanalyse

bildeanalysesekvens

Author:
Created: 4/16/2021 9:21:49 AM
Last Modified: 4/28/2021 8:13:05 PM
Comment:

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.new_Controller_0

1-8.POUs

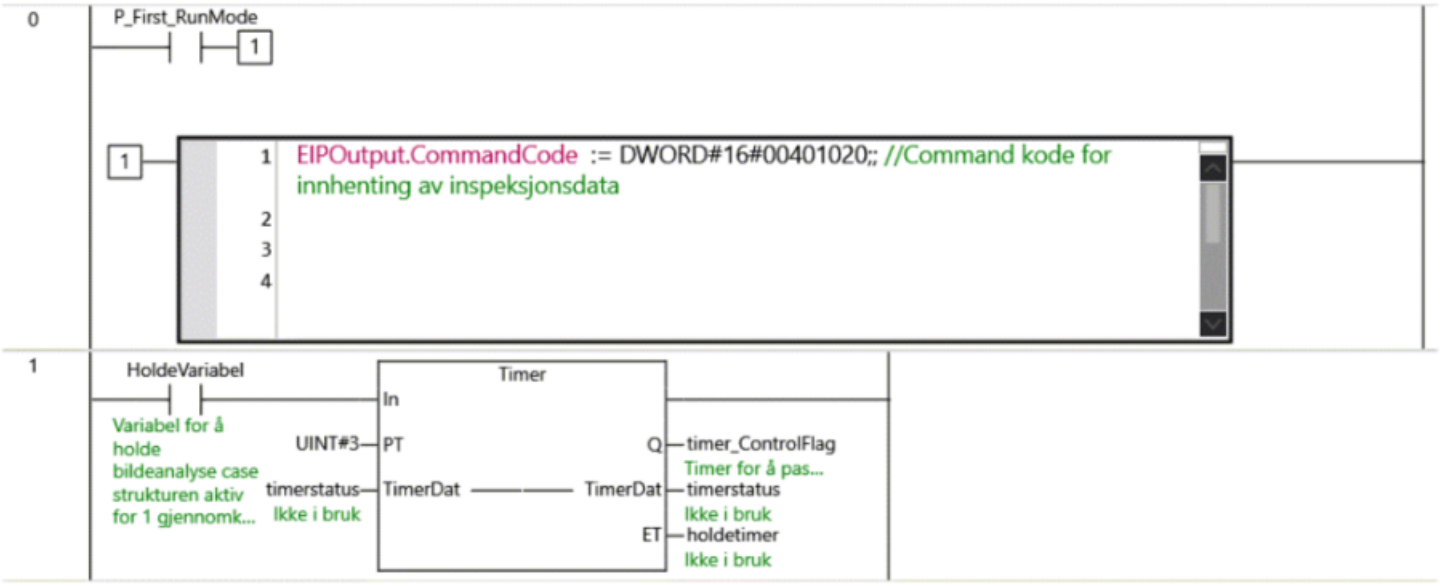
1-8-1.Programs

1-8-1-1.Bildeanalyse

1-8-1-1-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
HoldeVariabel	BOOL			False	False	Variabel for å holde bildeanalyse case strukturen aktiv for 1 gjennomkjøring
timerstatus	_sTimer			False	False	Ikke i bruk
holdetimer	UINT			False	False	Ikke i bruk
bildeanalyse	INT			False	False	Tilstandsvariabel for case strukturen
timer_ControlFlag	BOOL			False	False	Timer for å passe på data blir mottatt ved dataetterspørsel
Korkfeil_teller	BOOL			False	False	
VAR_EXTERNAL						
EIPInput	S_EIPInput				False	Strukturvariabel for input av data til PLS
EIPOutput	S_EIPOutput				False	Strukturvariabel for output av data til ekstern komponent (FQ2)
FlaskeSensor	BOOL				False	Sensor for aktivering av triggersignal til smartkameraet og kodegjennomkjøring
Bildeanalyse_korkfeil	REAL				False	Variabel for lagring av antall korkfeil
antall_flasker_analysert_reset	BOOL				False	Variabel for resetting av korkfeil variabel
Bildeanalyse_korkfeil_prosent	REAL				False	Prosentfremstilling av antall korkfeil
antall_flasker_analysert	REAL				False	Antall flasker analysert, brukt for utregning av prosent korkfeil

1-8-1-1-2.Section0



2

HoldeVariabel

1

Variabel for å holde bildeanalyse case strukturen aktiv for 1 gjennomk...

FlaskeSensor

Sensor for aktivering av triggersignal til smartkameraet og kodegjenno...

1

```
1   HoldeVariabel := TRUE;
2   CASE bildeanalyse OF
3      0: // Sender Dataetterspørsel for inspeksjonsfunksjonene i smartkameraet
4         EIPOutput.ControlFlag.F[0] := TRUE;
5         IF timer_ControlFlag THEN
6            bildeanalyse:= 1;
7         END_IF;
8      1: // tilstanden til alle 3 inspeksjonsfunksjonene må være godkjent for å
   passere som godkjent kork
9         EIPOutput.ControlFlag.F[0] := FALSE; //dataetterspørsel av
10        IF EIPInput.OutputData[0] + EIPInput.OutputData[1] + EIPInput.OutputData
   [2] <= -1000 THEN
11           bildeanalyse := 2;
12        ELSE
13           bildeanalyse := 3;
14        END_IF;
15     2: //ikke godkjent kork
16        Korkfeil_teller := TRUE;
17        bildeanalyse := 3;
18
19     3: //godkjent kork
20        Korkfeil_teller := FALSE;
21        antall_flasker_analysert := antall_flasker_analysert +1.0;
22        bildeanalyse := 0;
23        HoldeVariabel := FALSE;
24  END_CASE;
25
```

3

Korkfeil_teller

1

1

```
1  Bildeanalyse_korkfeil := Bildeanalyse_korkfeil +1.0; //global feilvariabel for
   bildeanalyse resultat
```

4

```
1  Bildeanalyse_korkfeil_prosent := (Bildeanalyse_korkfeil/antall_flasker_analysert)*100;
```

A.6 Program HMI Auto

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 4/25/2021 3:25:02 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

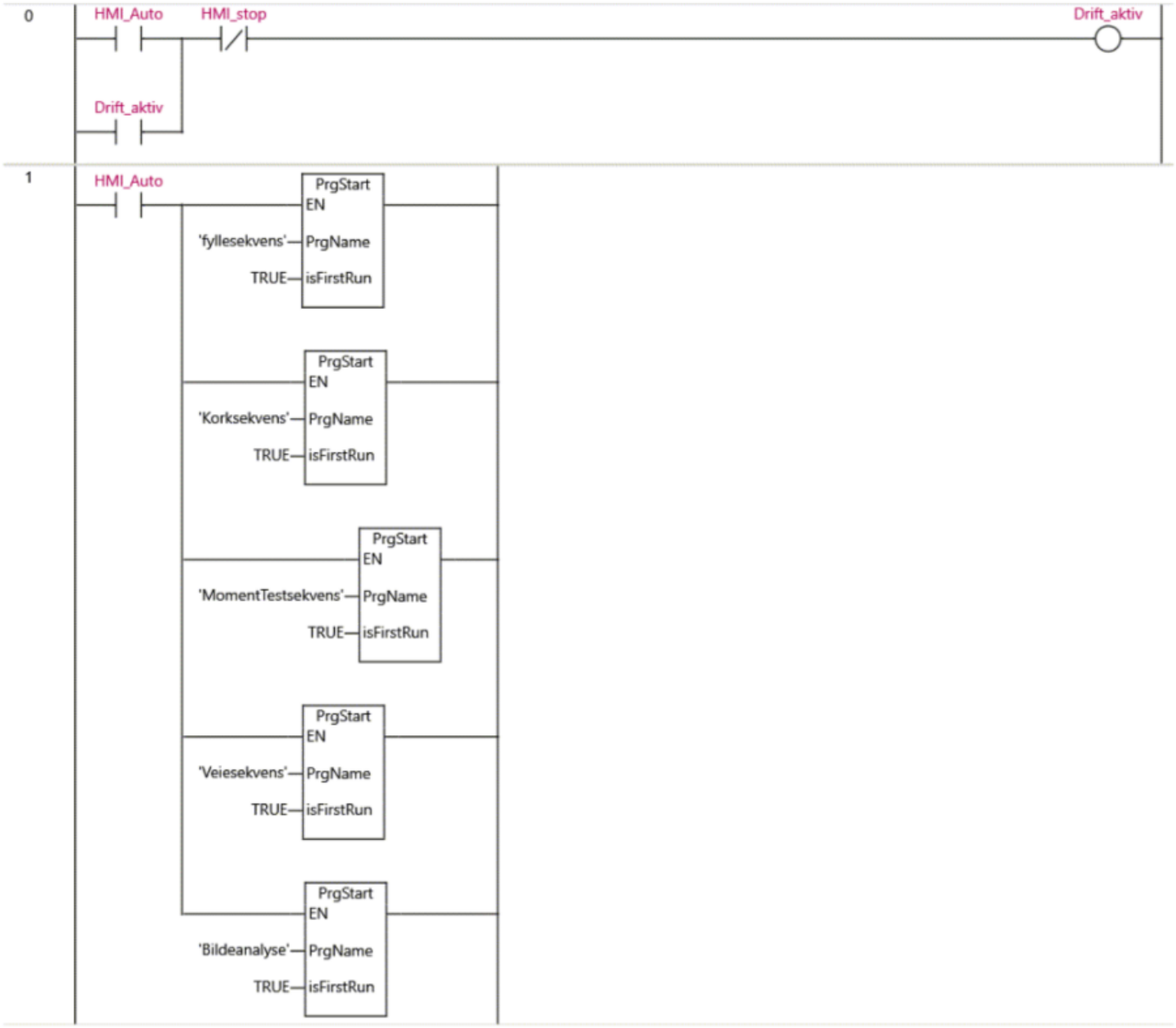
1-9-1.Programs

1-9-1-8.hovedprogram_auto

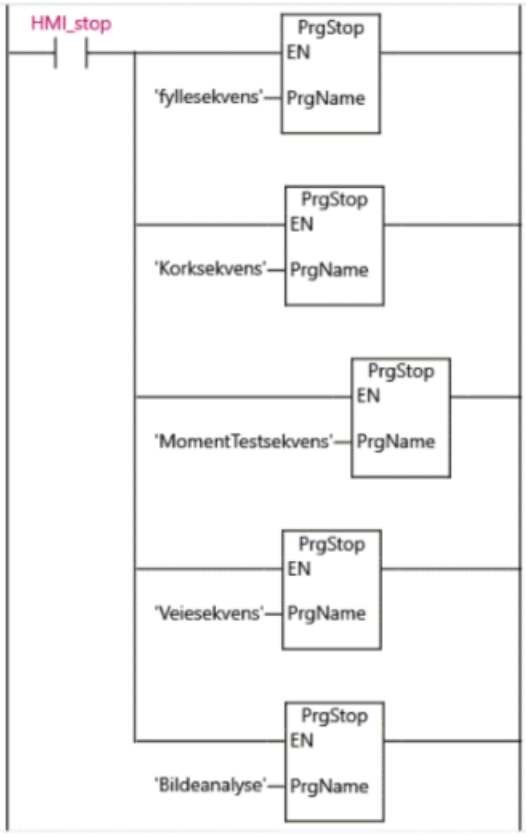
1-9-1-8-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR_EXTERNAL						
HMI_Auto	BOOL				False	
HMI_stop	BOOL				False	
Drift_aktiv	BOOL				False	

1-9-1-8-2.Section0



2


HMI_stop
PrgStop
EN
'fyllesekvens'
PrgName
PrgStop
EN
'Korksekvens'
PrgName
PrgStop
EN
'MomentTestsekvens'
PrgName
PrgStop
EN
'Veiesekvens'
PrgName
PrgStop
EN
'Bildeanalyse'
PrgName

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 4/25/2021 3:25:02 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

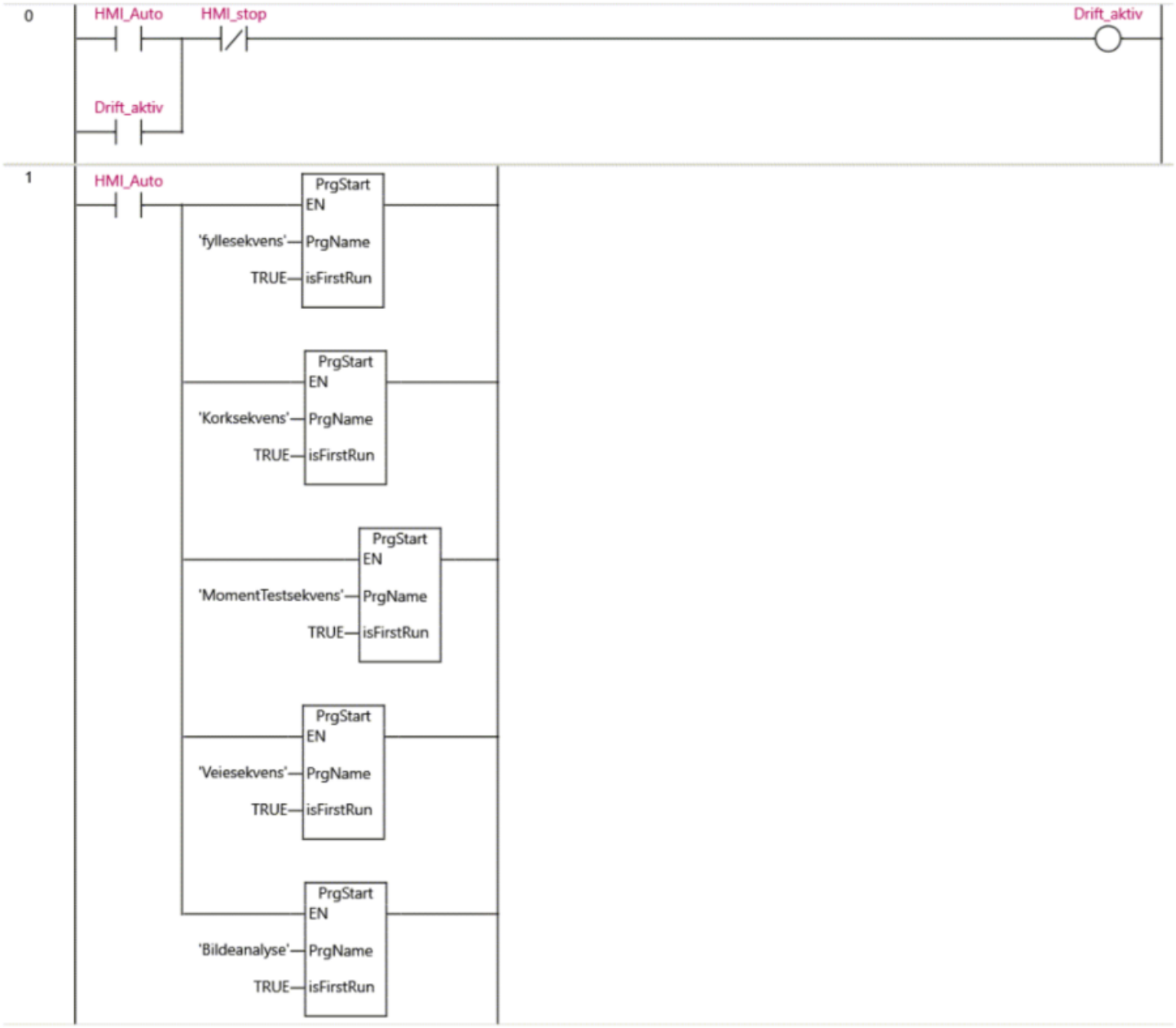
1-9-1.Programs

1-9-1-8.hovedprogram_auto

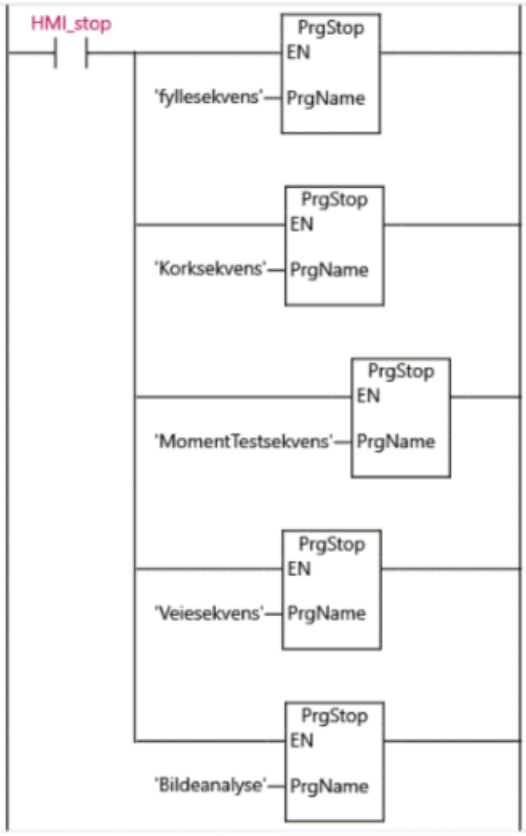
1-9-1-8-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR_EXTERNAL						
HMI_Auto	BOOL				False	
HMI_stop	BOOL				False	
Drift_aktiv	BOOL				False	

1-9-1-8-2.Section0



2


HMI_stop
PrgStop
EN
'fyllesekvens'
PrgName
PrgStop
EN
'Korksekvens'
PrgName
PrgStop
EN
'MomentTestsekvens'
PrgName
PrgStop
EN
'Veiesekvens'
PrgName
PrgStop
EN
'Bildeanalyse'
PrgName

A.7 Manuell styring

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 4/27/2021 3:03:50 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

1-9-1.Programs

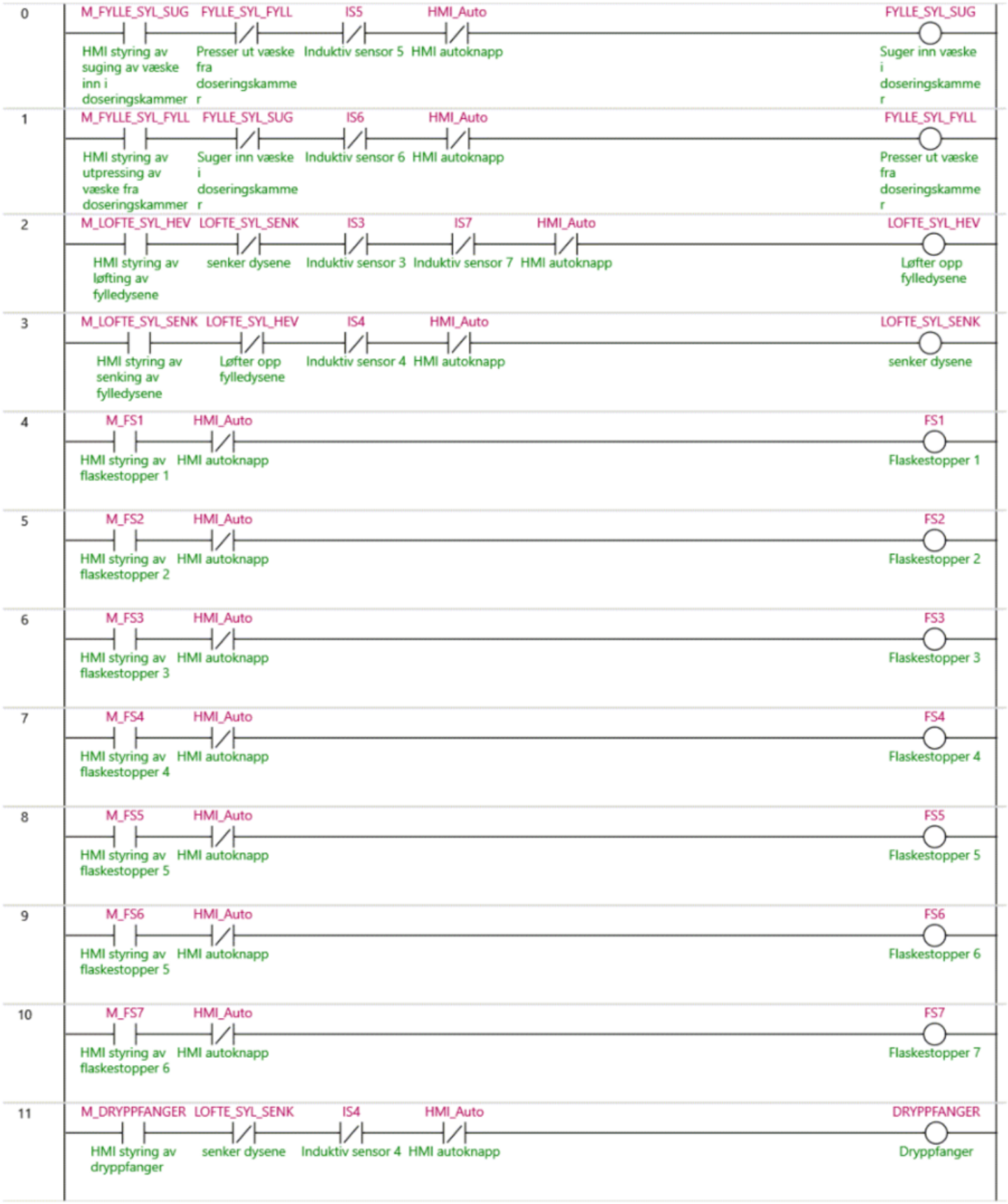
1-9-1-7.ManuellKjøring

1-9-1-7-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR_EXTERNAL						
FYLLE_SYL_FYLL	BOOL				False	Presser ut væske fra doseringskammer
LOFTE_SYL_HEV	BOOL				False	Løfter opp fylledysene
FS1	BOOL				False	Flaskestopper 1
FS2	BOOL				False	Flaskestopper 2
FS3	BOOL				False	Flaskestopper 3
FS4	BOOL				False	Flaskestopper 4
DRYPPFANGER	BOOL				False	Dryppfanger
HOVEDVENTIL	BOOL				False	hoverdventil for pneumatikk
LOFTE_SYL_SENK	BOOL				False	senker dysene
FYLLE_SYL_SUG	BOOL				False	Suger inn væske i doseringskammer
IS5	BOOL				False	Induktiv sensor 5
IS6	BOOL				False	Induktiv sensor 6
IS3	BOOL				False	Induktiv sensor 3
IS4	BOOL				False	Induktiv sensor 4
IS7	BOOL				False	Induktiv sensor 7
Transportband	BOOL				False	aktivering av transportband for transport av flasker i prosessen
FS5	BOOL				False	Flaskestopper 5
FS6	BOOL				False	Flaskestopper 6
FS7	BOOL				False	Flaskestopper 7
M_FYLLE_SYL_SUG	BOOL				False	HMI styring av suging av væske inn i doseringskammer
M_LOFTE_SYL_HEV	BOOL				False	HMI styring av løfting av fylledysene
M_FS1	BOOL				False	HMI styring av flaskestopper 1
M_FS2	BOOL				False	HMI styring av flaskestopper 2
M_FS3	BOOL				False	HMI styring av flaskestopper 3
M_FS5	BOOL				False	HMI styring av flaskestop

						per 5
M_FS4	BOOL				False	HMI styring av flaskestopper 4
M_FS6	BOOL				False	HMI styring av flaskestopper 5
M_FS7	BOOL				False	HMI styring av flaskestopper 6
M_DRYPPFANGER	BOOL				False	HMI styring av dryppfanger
M_HOVEDVENTIL	BOOL				False	HMI styring av hoverdventil for pneumatikk
M_KORK_SPINN	BOOL				False	HMI styring av korkmaskin spinn
M_FYLLE_SYL_FYLL	BOOL				False	HMI styring av utpressing av væske fra doseringskammer
M_KORK_OPP_ELLER_NED	BOOL				False	HMI styring av korkmaskin
M_LOFTE_SYL_SENK	BOOL				False	HMI styring av senking av fylledysene
M_transportband	BOOL				False	HMI styring av
HMI_Auto	BOOL				False	HMI autoknapp
Korkmaskin_ned_opp	BOOL				False	Korkmaskin [1]ned og [2] deretter opp (Impuls styrt)
Korkmaskin_Spinn	BOOL				False	Korkmaskin spinn

1-9-1-7-2.Section0




12
M_HOVEDVENTIL
HMI_Auto
HOVEDVENTIL
HMI styring av hoverdventil for pneumatikk
HMI autoknapp
hoverdventil for pneumatikk
13
M_KORK_SPINN
HMI_Auto
Korkmaskin_Spinn
HMI styring av korkmaskin spinn
HMI autoknapp
Korkmaskin spinn
14
M_KORK_OPP_ELLER_NED
HMI_Auto
Korkmaskin_ned_opp
HMI styring av korkmaskin
HMI autoknapp
Korkmaskin [1]ned og [2]deretter opp (Impuls styrt)
15
M_transportband
HMI_Auto
Transportband
HMI styring av
HMI autoknapp
aktivering av transportband for transport av flasker i prosessen

A.8 Kalibreringsprogram

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 5/8/2021 2:42:02 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

1-9-1.Programs

1-9-1-6.VektSkalering

1-9-1-6-1.Variables

Name	Data Type	Initial Value	AT	Retain	Constant	Comment
VAR						
Gjennomsnitts_minvekt	REAL			False	False	
Gjennomsnitts_maksvekt	REAL			False	False	
Kalibrering_minvekt_timer	TIME			False	False	
Skalering	Skaleringsblokk			False	False	
uskalert_kalibreringsobjekt_maksvek	REAL			False	False	
uskalert_kalibreringsobjekt_maksimumsvekt	REAL			False	False	
uskalert_kalibreringsobjekt_minimumsvekt	REAL			False	False	
Kalibreringsobjekt_minimumsvekt	Gjennomsnittsvekt			False	False	
Kalibreringsobjekt_maksimumsvekt	Gjennomsnittsvekt			False	False	
VAR_EXTERNAL						
Reset_kalibrering	BOOL				False	
skalert_vekt_kg	REAL				False	
Start_kalibrering_maxvekt	BOOL				False	
Start_kalibrering_minvekt	BOOL				False	
Uskalert_Flaskevekt	REAL				False	
Uskalert_vekt_kg	REAL				False	
Kalibreringso	REAL				False	

	bjektets_minimumsvekt						
	Kalibreringsobjektets_maksimumsvekt	REAL				False	
	Kalibrering_timer_minimumsvekt	TIME				False	
	Kalibrering_timer_maksimumsvekt	TIME				False	

1-9-1-6-2.VektSkalering

A.9 Funksjonsblokker

A.9.1 Gjennomsnittsvektblokk

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 4/26/2021 3:02:44 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

1-9-2.Function Blocks

1-9-2-1.Gjennomsnittsvekt

1-9-2-1-1.Variables

Name	Data Type	Edge	Initial Value	AT	Retain	Constant	Comment
VAR							
teller_maalinger	REAL				False	False	
samlet_verdi	REAL				False	False	
timer1	TOF				False	False	
timer_paa	BOOL				False	False	
VAR_INPUTOUTPUT							
ENO	BOOL	No Edge			False	False	
Start	BOOL	No Edge			False	False	Starter timer på 10 sekund
Vekt_inn	REAL	No Edge			False	False	Uskalert vekt inn
Reset	BOOL	No Edge			False	False	resetter timer
Vekt_ut	REAL	No Edge			False	False	gjennomsnittlig vekt
Kalibreringstimer	TIME	No Edge			False	False	Viser timer verdi under telling
VAR_EXTERNAL							
KORK_FOTO_SENSOR	BOOL					False	

1-9-2-1-2.Program Body

```
1     timer1(In:=start, PT:=T#10s, Q=>timer_paa, ET=>Kalibreringstimer);
2
3     IF timer_paa THEN
4         teller_maalinger := teller_maalinger +1.0;
5         samlet_verdi := samlet_verdi + Vekt_inn;
6     END_IF;
7
8     Vekt_ut := (1/teller_maalinger)*samlet_verdi;
9
10    IF reset THEN
11        vekt_ut := 0;
12        teller_maalinger := 0;
13        samlet_verdi := 0;
14    END_IF;
15
16
17
18
19
20
```

A.9.2 Skaleringsblokk

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 4/26/2021 3:02:44 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

1-9-2.Function Blocks

1-9-2-2.Skaleringsblokk

1-9-2-2-1.Variables

Name	Data Type	Edge	Initial Value	AT	Retain	Constant	Comment
VAR							
a	REAL				False	False	Skaleringsfaktor
b	REAL				False	False	Forskyvning fra nullvekt
VAR_INPUTOUTPUT							
ENO	BOOL	No Edge			False	False	
Start	BOOL	No Edge			False	False	Starter skalering
y1	REAL	No Edge			False	False	Kalibreringsobjektets maksimumsvekt
y0	REAL	No Edge			False	False	Kalibreringsobjektets minimumsvekt
x1	REAL	No Edge			False	False	målt uskalert kalibreringsobjekt maksimumsvekt
x0	REAL	No Edge			False	False	målt uskalert kalibreringsobjekt minumumsvekt
x	REAL	No Edge			False	False	Uskalert vekt inn
y	REAL	No Edge			False	False	Skalert vekt ut
VAR_EXTERNAL							
Start_kalibrering_maxvekt	BOOL					False	

1-9-2-2-2.Program Body

```
1     IF Start THEN
2     (*skaleringsfaktor*)
3         a := (y1 - y0)/
4               (x1 - x0);
5
6         b:=x0;
7         (*ferdig skalert vekt i kg*)
8         y := a*(x - b);
9
10        (*passer på at vekta ikke går under 0*)
11         IF y <= 0.0 THEN
12                      y := 0.0;
13         END_IF;
14    END_IF;
15
```

A.9.3 TM_Stickfunksjonerblokk

Fullstendig_flaskefyllingsprogram

Author: Lasse
Created: 4/19/2021 3:12:53 PM
Last Modified: 5/8/2021 2:42:02 PM
Comment: Starter hver deloppgave med dette oppsettet

Sysmac Studio Module Version: 1.40.0.64008

Table of Contents

1.NX102

1-9.POUs

1-9-2.Function Blocks

1-9-2-4.TM_Stickfunksjoner

1-9-2-4-1.Variables

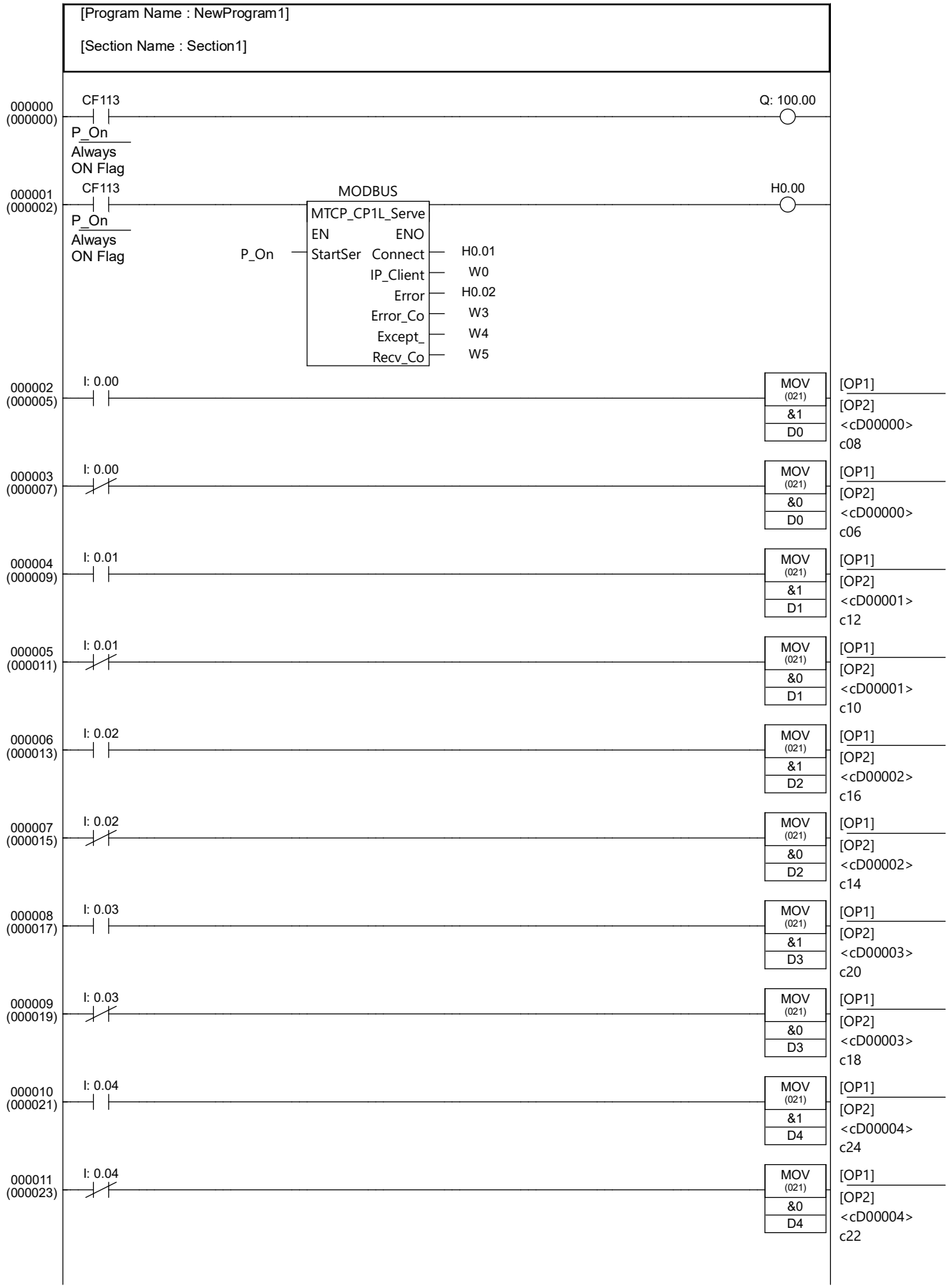
Name	Data Type	Edge	Initial Value	AT	Retain	Constant	Comment
VAR							
stage	INT				False	False	
Autosekvens	BOOL				False	False	
VAR_INPUTOUTPUT							
ENO	BOOL	No Edge			False	False	
Enable	BOOL	No Edge			False	False	
stop	BOOL	No Edge			False	False	
playpause	BOOL	No Edge			False	False	
ProgramSpeedUP	BOOL	No Edge			False	False	
ProgramSpeedDown	BOOL	No Edge			False	False	
Togglemode	BOOL	No Edge			False	False	
CurrentProgramSpeed	INT	No Edge			False	False	
CurrentMode	OEN\nROBOT\nTM\eMODE	No Edge			False	False	
VAR_EXTERNAL							
TM_Robot	OEN\nRobot\nTM\sRobot					False	

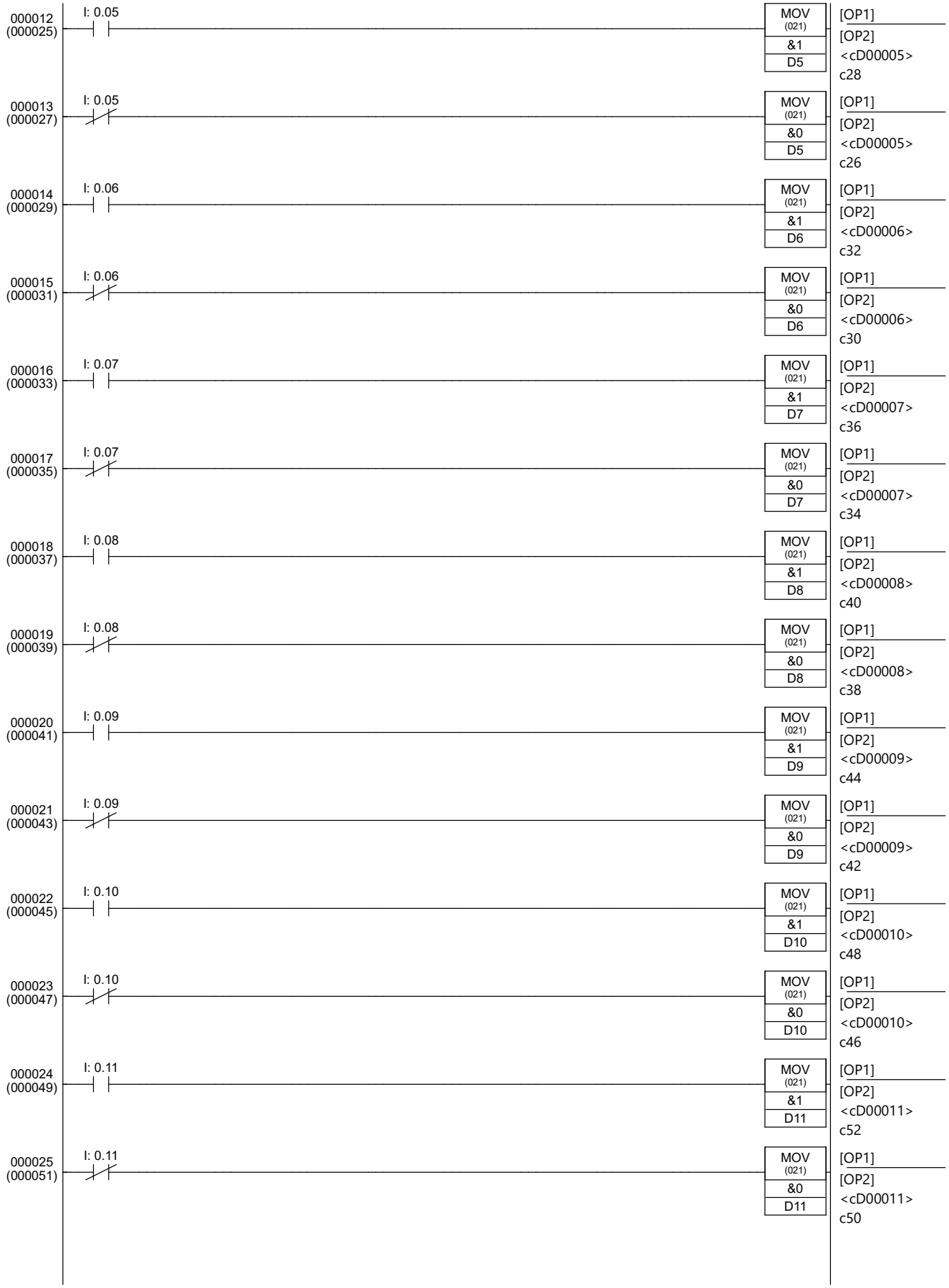
1-9-2-4-2.Program Body

```
1       IF Enable THEN
2           //INPUTS
3           TM_Robot.IO.CBoxDO[13] := stop;
                    // Aktiverer Cbox digital utgang 13 for å stoppe programmet (flanke styrt)
4           TM_Robot.IO.CBoxDO[12] := playpause;
            // Aktiverer Cbox digital utgang 12 for å Pause eller Starte programmet (flanke styrt)
5           TM_Robot.IO.CBoxDO[8] := Togglemode;                                                    //
Aktiverer Cbox digital utgang 8 for å toggle veksle mellom manuell og auto (flanke styrt)
6
7           IF Autosekvens=FALSE THEN
                                    // for å forhindre bruker å trykke på + og - under overgang til Auto mode
8                   TM_Robot.IO.CBoxDO[7] := ProgramSpeedDOWN;          // Aktiverer CBox digital utgang 7
for å senke programfart (flankestyrt)
9                   TM_Robot.IO.CBoxDO[6] := ProgramSpeedUP;                                        // Aktiverer CBox
digital utgang 6 for å øke programfart (flankestyrt)
10          END_IF;
11
12          //OUTPUTS
13          CurrentProgramSpeed := TM_Robot.Stick.Speed;
14          CurrentMode := TM_Robot.Stick.Mode;
15
16          IF Togglemode  THEN // betingelse togglemode (flankestyrt) og at programmer er i manuell for å starte
sekvensen
17                  stage := 1;
18                  Autosekvens := TRUE;
19
20          //sekvens for å starte auto mode
21                  CASE stage OF
22                          1: TM_Robot.IO.CBoxDO[6] := TRUE;
23                          IF TM_Robot.IO.CBoxDI[9] THEN
24                                  stage := 2;
25                          END_IF;
26
27                          2:
28                                  TM_Robot.IO.CBoxDO[6] := FALSE;
29                                  IF TM_Robot.IO.CBoxDI[9] = FALSE THEN
30                                          stage :=3;
31                                  END_IF;
32
33                          3:
34                                  TM_Robot.IO.CBoxDO[7] := TRUE;
35                                  IF TM_Robot.IO.CBoxDI[10] = TRUE THEN
36                                          stage := 4;
37                                  END_IF;
38
39                          4:
40                                  TM_Robot.IO.CBoxDO[7] := FALSE;
41                                  IF TM_Robot.IO.CBoxDI[10] = FALSE THEN
42                                          stage := 5;
43                                  END_IF;
44
45                          5:
```

```
                    TM_Robot.IO.CBoxDO[6] := TRUE;
                    IF TM_Robot.IO.CBoxDI[9] = TRUE THEN
                        stage := 6;
                    END_IF;

                6:
                    TM_Robot.IO.CBoxDO[6] := FALSE;
                    IF TM_Robot.IO.CBoxDI[9] = FALSE THEN
                        stage := 7;
                    END_IF;

                7:
                    TM_Robot.IO.CBoxDO[6] := TRUE;
                    IF TM_Robot.IO.CBoxDI[9] = TRUE THEN
                        stage := 8;
                    END_IF;

                8:
                    TM_Robot.IO.CBoxDO[6] := FALSE;
                    IF TM_Robot.IO.CBoxDI[9] = FALSE THEN
                        stage := 9;
                    END_IF;

                9:
                    TM_Robot.IO.CBoxDO[7] := TRUE;
                    IF TM_Robot.IO.CBoxDI[10] = TRUE THEN
                        stage := 10;
                    END_IF;

                10:
                    TM_Robot.IO.CBoxDO[7] := FALSE;
                    IF TM_Robot.IO.CBoxDI[10] = FALSE THEN
                        Autosekvens := FALSE;
                        stage := 0;
                    END_IF;
            END_CASE;
        END_IF;
    END_IF;

```

A.10 Modbus server CX-program


[Program Name : NewProgram1]
[Section Name : Section1]
000000
(000000)
CF113
P_On
Always
ON Flag
Q: 100.00
000001
(000002)
CF113
P_On
Always
ON Flag
MODBUS
MTCP_CP1L_Serve
EN
ENO
P_On
StartSer
Connect
IP_Client
Error
Error_Co
Except_
Recv_Co
H0.01
W0
H0.02
W3
W4
W5
H0.00
000002
(000005)
I: 0.00
MOV
(021)
&1
D0
[OP1]
[OP2]
<cD00000>
c08
000003
(000007)
I: 0.00
MOV
(021)
&0
D0
[OP1]
[OP2]
<cD00000>
c06
000004
(000009)
I: 0.01
MOV
(021)
&1
D1
[OP1]
[OP2]
<cD00001>
c12
000005
(000011)
I: 0.01
MOV
(021)
&0
D1
[OP1]
[OP2]
<cD00001>
c10
000006
(000013)
I: 0.02
MOV
(021)
&1
D2
[OP1]
[OP2]
<cD00002>
c16
000007
(000015)
I: 0.02
MOV
(021)
&0
D2
[OP1]
[OP2]
<cD00002>
c14
000008
(000017)
I: 0.03
MOV
(021)
&1
D3
[OP1]
[OP2]
<cD00003>
c20
000009
(000019)
I: 0.03
MOV
(021)
&0
D3
[OP1]
[OP2]
<cD00003>
c18
000010
(000021)
I: 0.04
MOV
(021)
&1
D4
[OP1]
[OP2]
<cD00004>
c24
000011
(000023)
I: 0.04
MOV
(021)
&0
D4
[OP1]
[OP2]
<cD00004>
c22


000012
(000025)
I: 0.05
MOV
(021)
&1
D5
[OP1]
[OP2]
<cD00005>
c28
000013
(000027)
I: 0.05
MOV
(021)
&0
D5
[OP1]
[OP2]
<cD00005>
c26
000014
(000029)
I: 0.06
MOV
(021)
&1
D6
[OP1]
[OP2]
<cD00006>
c32
000015
(000031)
I: 0.06
MOV
(021)
&0
D6
[OP1]
[OP2]
<cD00006>
c30
000016
(000033)
I: 0.07
MOV
(021)
&1
D7
[OP1]
[OP2]
<cD00007>
c36
000017
(000035)
I: 0.07
MOV
(021)
&0
D7
[OP1]
[OP2]
<cD00007>
c34
000018
(000037)
I: 0.08
MOV
(021)
&1
D8
[OP1]
[OP2]
<cD00008>
c40
000019
(000039)
I: 0.08
MOV
(021)
&0
D8
[OP1]
[OP2]
<cD00008>
c38
000020
(000041)
I: 0.09
MOV
(021)
&1
D9
[OP1]
[OP2]
<cD00009>
c44
000021
(000043)
I: 0.09
MOV
(021)
&0
D9
[OP1]
[OP2]
<cD00009>
c42
000022
(000045)
I: 0.10
MOV
(021)
&1
D10
[OP1]
[OP2]
<cD00010>
c48
000023
(000047)
I: 0.10
MOV
(021)
&0
D10
[OP1]
[OP2]
<cD00010>
c46
000024
(000049)
I: 0.11
MOV
(021)
&1
D11
[OP1]
[OP2]
<cD00011>
c52
000025
(000051)
I: 0.11
MOV
(021)
&0
D11
[OP1]
[OP2]
<cD00011>
c50

A.11 Modbus klient python program

```
from pymodbus.client.sync import ModbusTcpClient as modbus
pls_IP = '192.168.250.1'
client = modbus(pls_IP)
PLS = client.connect()

if PLS:
    print('Modbus kommunikasjon er etablert til IP:', pls_IP)
else:
    print('Modbus kommunikasjon ikke etablert.')


def input_0():
    result = client.read_holding_registers(0, 12)
    input_0 = bool(result.registers[0])
    return input_0


def input_1():
    result = client.read_holding_registers(0, 12)
    input_1 = bool(result.registers[1])
    return input_1


def input_2():
    result = client.read_holding_registers(0, 12)
    input_2 = bool(result.registers[2])
    return input_2


def input_3():
    result = client.read_holding_registers(0, 12)
    input_3 = bool(result.registers[3])
    return input_3


def input_4():
    result = client.read_holding_registers(0, 12)
    input_4 = bool(result.registers[4])
    return input_4


def input_5():
    result = client.read_holding_registers(0, 12)
    input_5 = bool(result.registers[5])
    return input_5


def input_6():
    result = client.read_holding_registers(0, 12)
    input_6 = bool(result.registers[6])
    return input_6


def input_7():
    result = client.read_holding_registers(0, 12)
```

```
56      input_7 = bool(result.registers[7])
57      return input_7
58
59
60  def input_8():
61      result = client.read_holding_registers(0, 12)
62      input_8 = bool(result.registers[8])
63      return input_8
64
65
66  def input_9():
67      result = client.read_holding_registers(0, 12)
68      input_9 = bool(result.registers[9])
69      return input_9
70
71
72  def input_10():
73      result = client.read_holding_registers(0, 12)
74      input_10 = bool(result.registers[10])
75      return input_10
76
77
78  def input_11():
79      result = client.read_holding_registers(0, 12)
80      input_11 = bool(result.registers[11])
81      return input_11
82
83
84  client.close()
```

A.12 OPC UA server python program

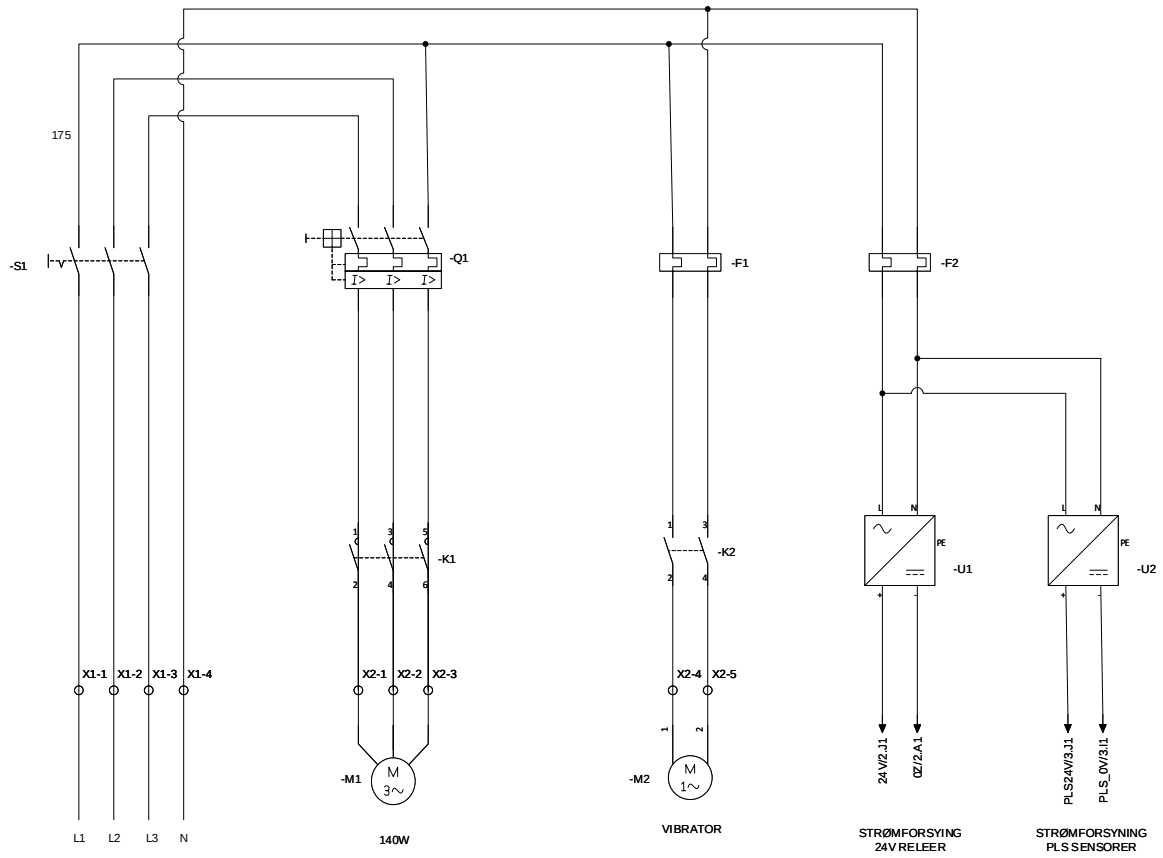
```
# Importerer OPCUA serverfunksjoner fra opcua bibliotek samt time funksjon og ... scriptet modbus.py
from opcua import Server
import time
import modbus

server = Server()
IP_raspberry = "opc.tcp://192.168.250.2:4840"
server.set_endpoint(IP_raspberry)

navn = "OPCUA server"
addspace = server.register_namespace(navn)

# Oppretter nodeklassen objects
node = server.get_objects_node()
# Knytter noden Inputs til nodeklassen objects
Inputs = node.add_object(addspace, "Inputs")
#Legger inn variabler som skal leses
input0 = Inputs.add_variable(addspace, "Input_00", 0)
input1 = Inputs.add_variable(addspace, "Input_01", 0)
input2 = Inputs.add_variable(addspace, "Input_02", 0)
input3 = Inputs.add_variable(addspace, "Input_03", 0)
input4 = Inputs.add_variable(addspace, "Input_04", 0)
input5 = Inputs.add_variable(addspace, "Input_05", 0)
input6 = Inputs.add_variable(addspace, "Input_06", 0)
input7 = Inputs.add_variable(addspace, "Input_07", 0)
input8 = Inputs.add_variable(addspace, "Input_08", 0)
input9 = Inputs.add_variable(addspace, "Input_09", 0)
input10 = Inputs.add_variable(addspace, "Input_10", 0)
input11 = Inputs.add_variable(addspace, "Input_11", 0)

input0.set_writable()
input1.set_writable()
input2.set_writable()
input3.set_writable()
input4.set_writable()
input5.set_writable()
input6.set_writable()
input7.set_writable()
input8.set_writable()
input9.set_writable()
input10.set_writable()
input11.set_writable()

server.start()
print("Server startet i {}". format(IP_raspberry))

while True:

    # Henter statusverdier fra PLS via modbus.py

    Input0 = modbus.input_0()
    Input1 = modbus.input_1()
    Input2 = modbus.input_2()
    Input3 = modbus.input_3()
```

```
55      Input4 = modbus.input_4()
56      Input5 = modbus.input_5()
57      Input6 = modbus.input_6()
58      Input7 = modbus.input_7()
59      Input8 = modbus.input_8()
60      Input9 = modbus.input_9()
61      Input10 = modbus.input_10()
62      Input11 = modbus.input_11()
63
64      #print('Input0 = ', Input0,'Input1 = ', Input1,'Input2 = ', Input2,'Input3 = ', ...
            Input3,'Input4 = ', Input4,
65           # 'Input5 = ', Input5,'Input6 = ', Input6,'Input7 = ', Input7,'Input8 = ', ...
                Input8,'Input9 = ', Input9,
66           # 'Input10 = ', Input10,'Input11 = ', Input11)
67
68      # Setter verdiene
69      input0.set_value(Input0)
70      input1.set_value(Input1)
71      input2.set_value(Input2)
72      input3.set_value(Input3)
73      input4.set_value(Input4)
74      input5.set_value(Input5)
75      input6.set_value(Input6)
76      input7.set_value(Input7)
77      input8.set_value(Input8)
78      input9.set_value(Input9)
79      input10.set_value(Input10)
80      input11.set_value(Input11)
81
82      time.sleep(2)
```

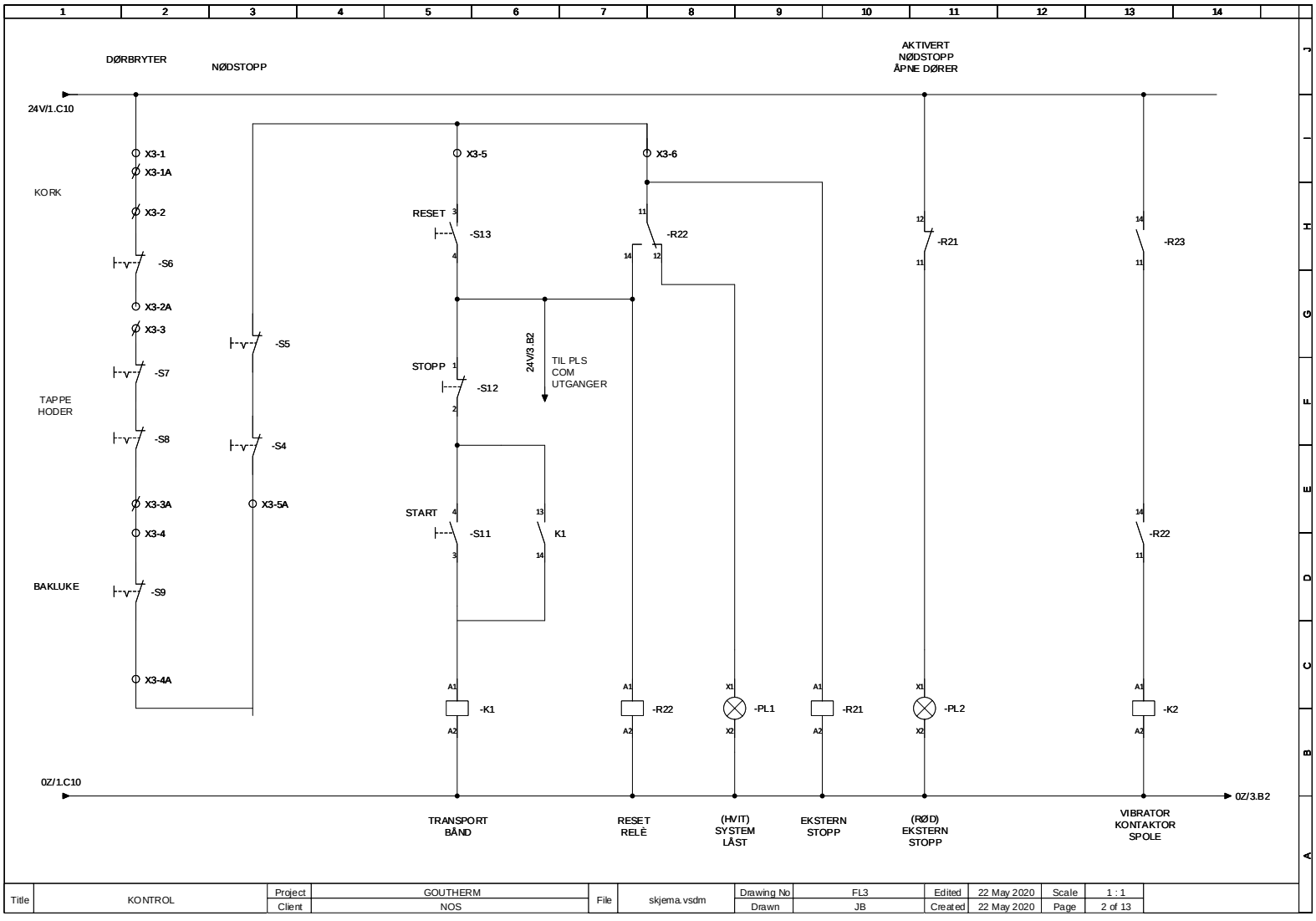
Vedlegg B

Skjemategninger

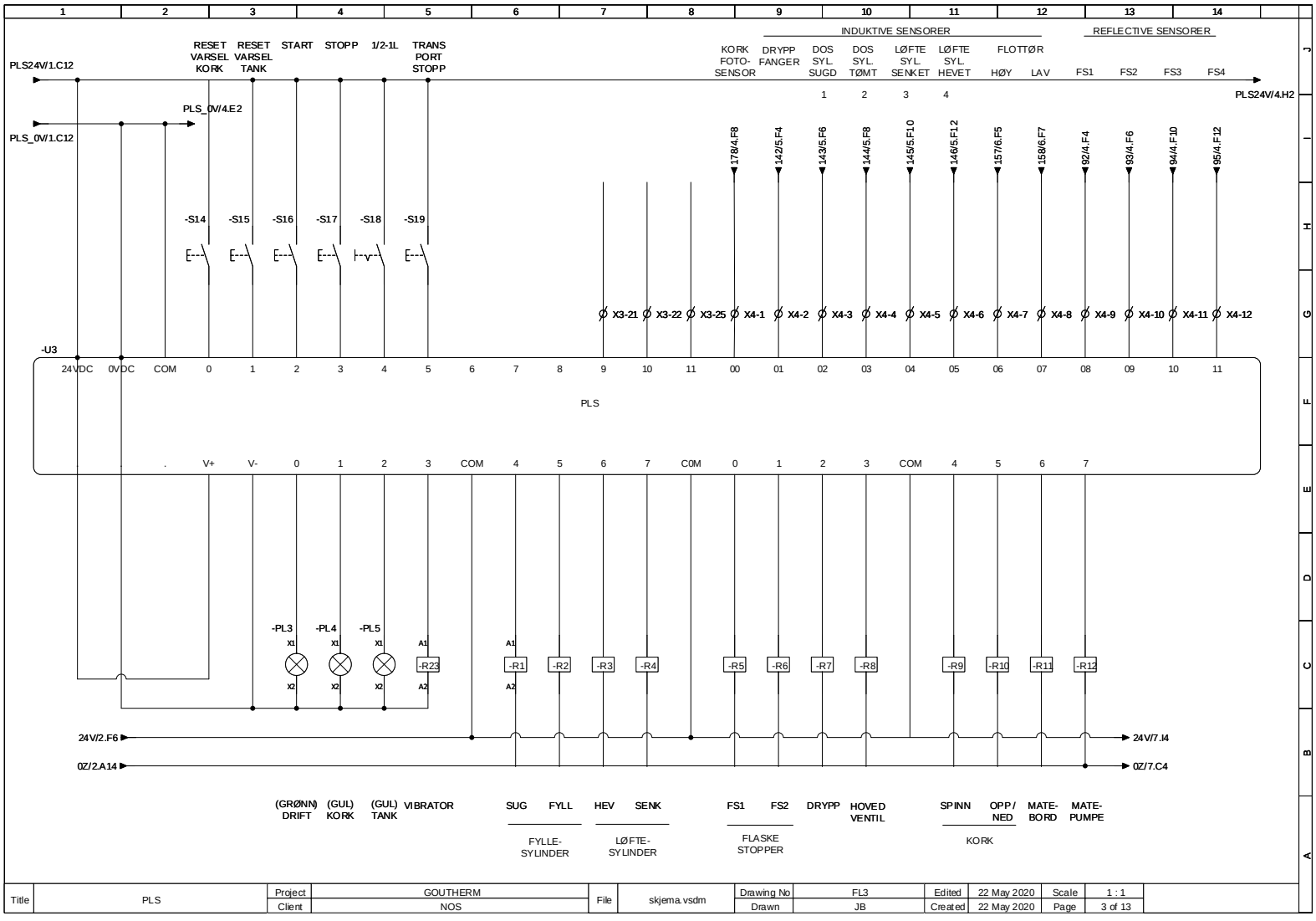
B.1 Skjemategning for opprinnelig funksjonalitet



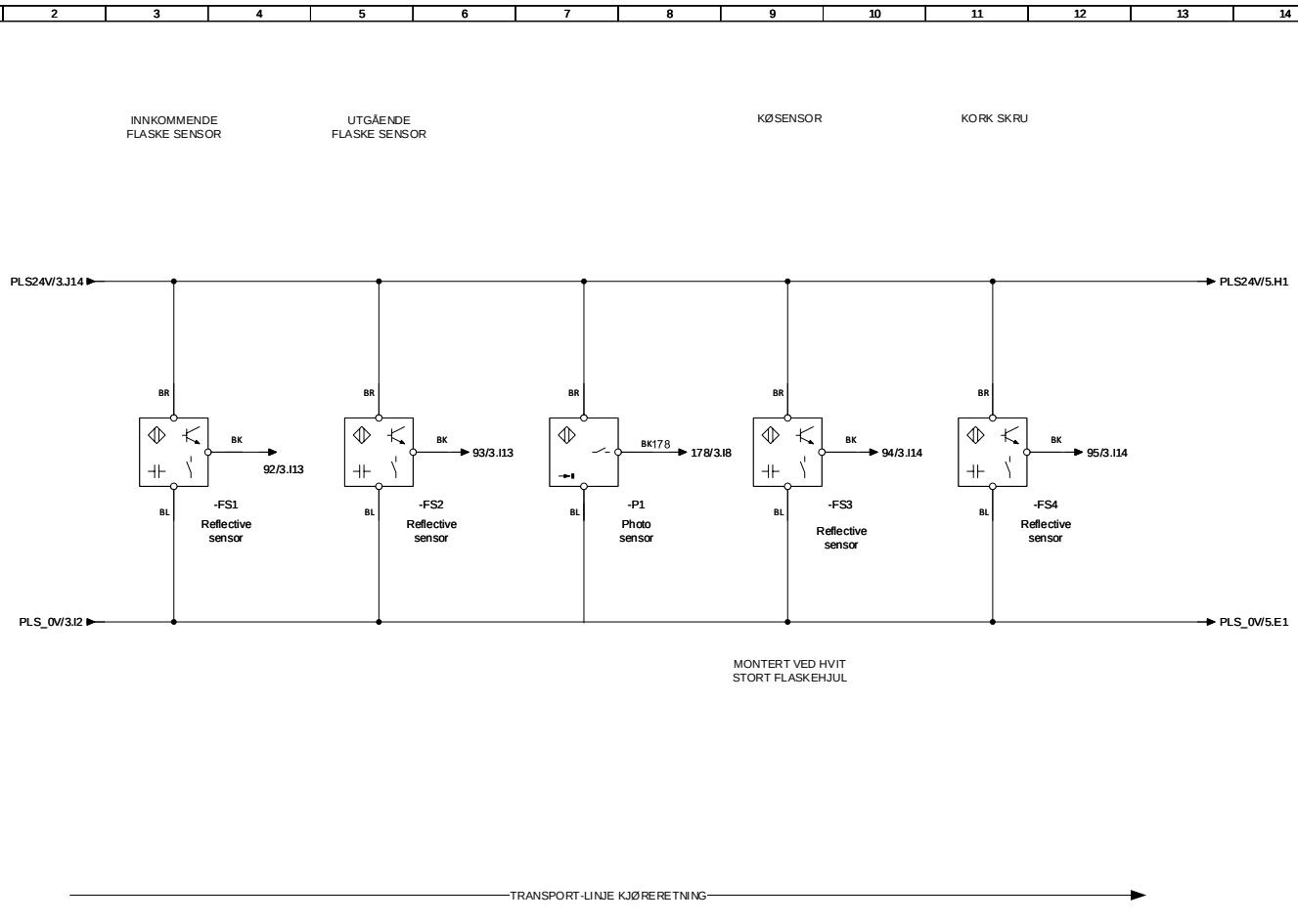
Title	HOVED	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	1 of 13



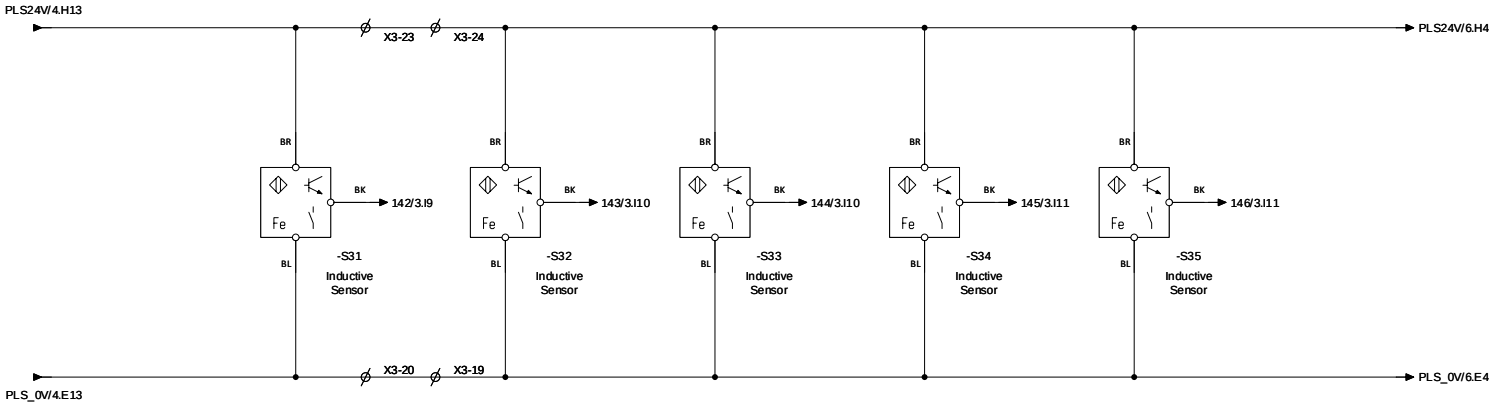
Title	KONTROL	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	2 of 13



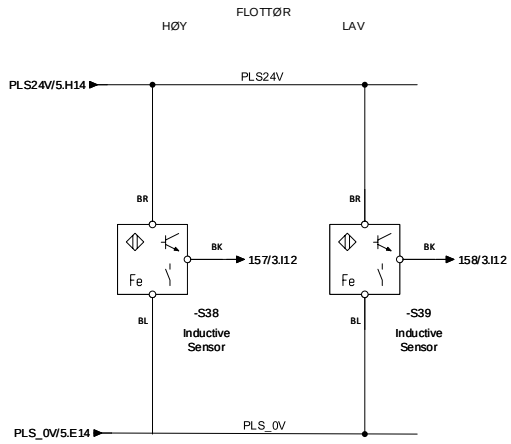
Title	PLS	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	3 of 13



Title	REFLECTIVE	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	4 of 13

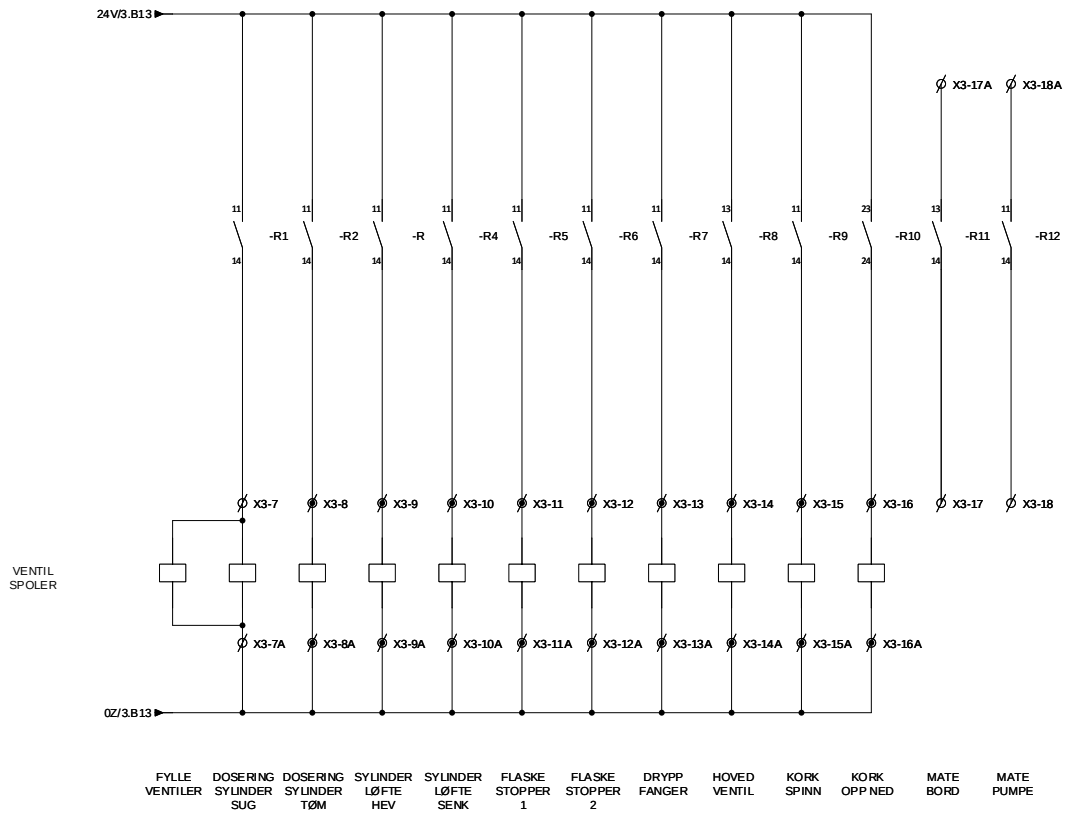


Title	IND	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	5 of 13

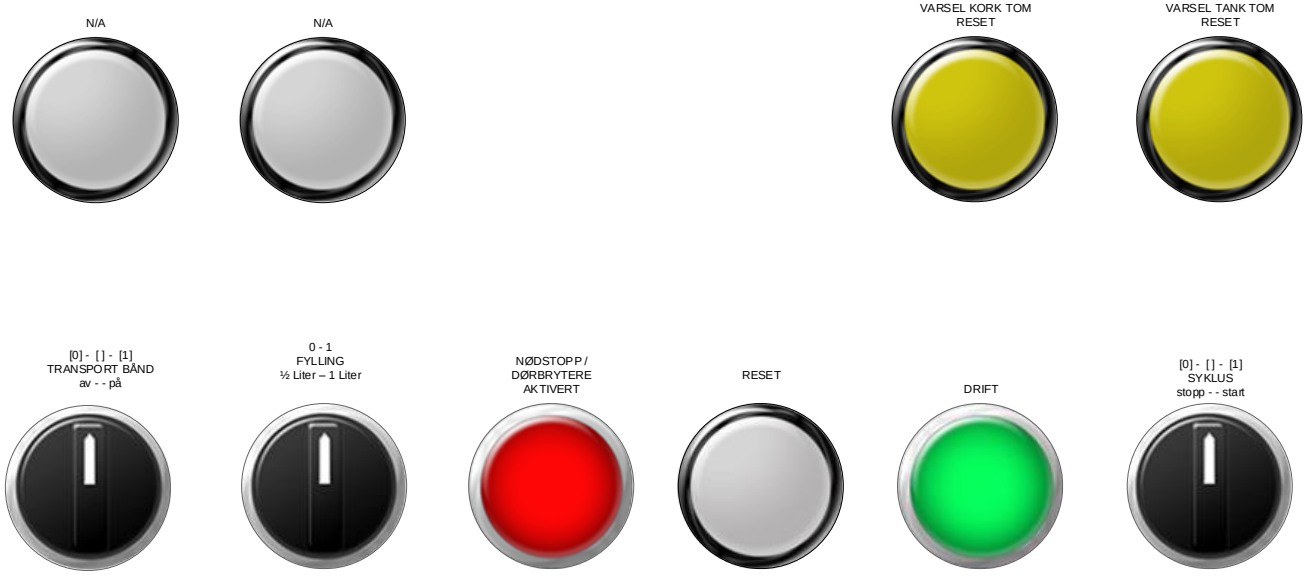


FLOTTØR SENSORER

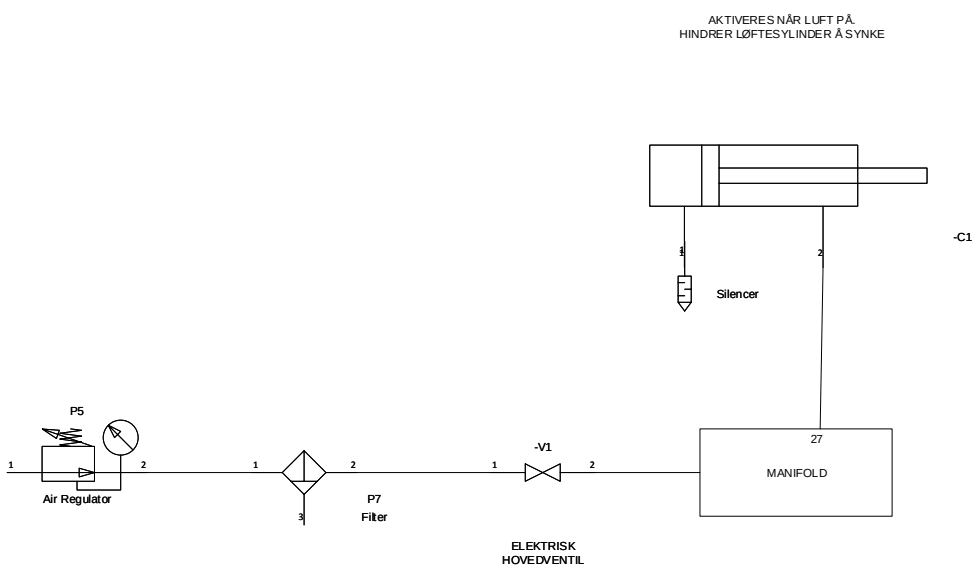
Title	TANK	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	6 of 13



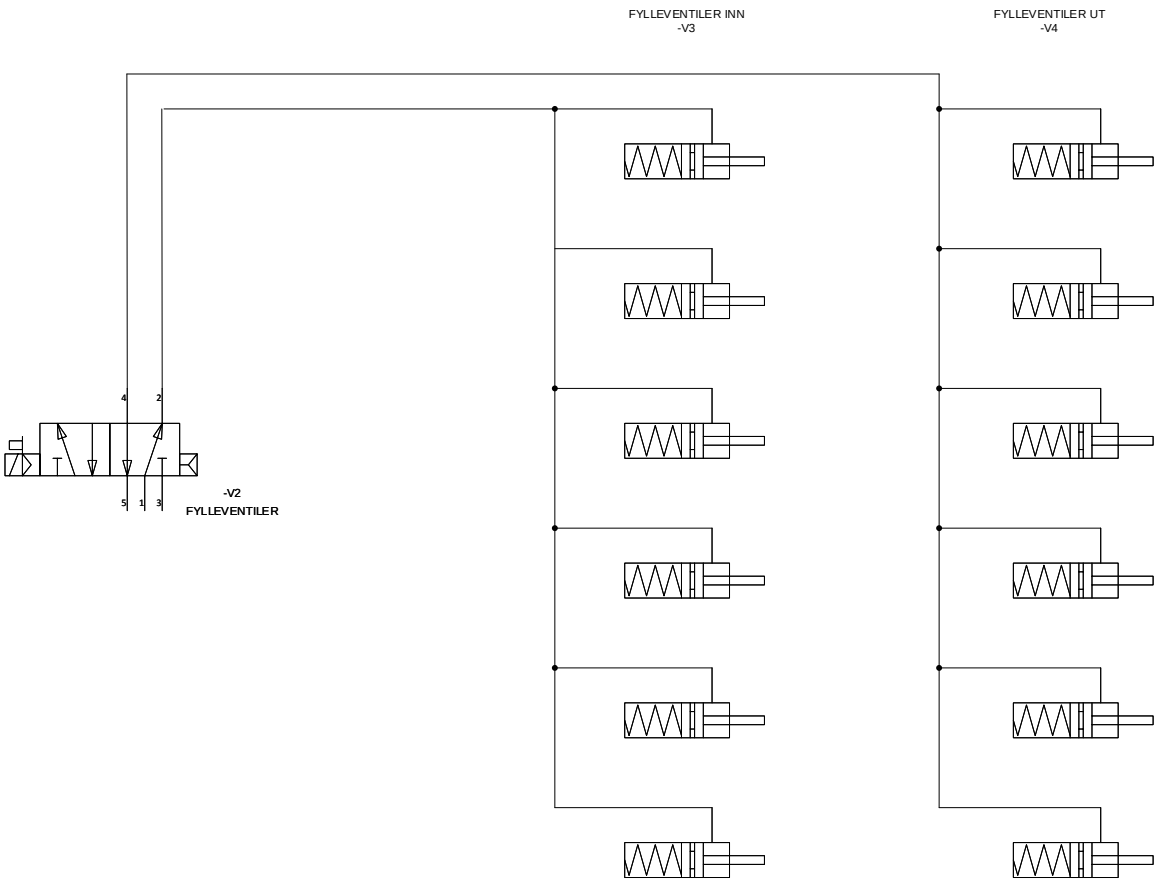
Title	VENTIL	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1		
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	7 of 13		



Title	PANEL	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	8 of 13

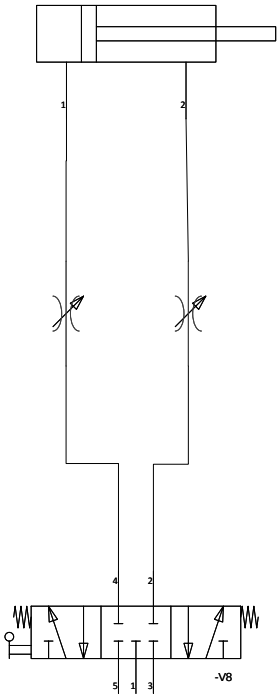


Title	HOVEDLUFT	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	9 of 13



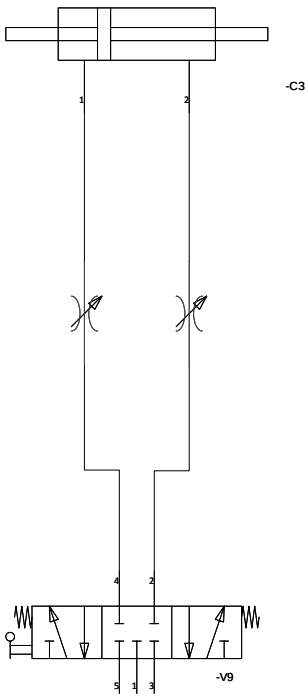
Title	FYLLEVENTIL	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1		
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	10 of 13		

SYLINDER FOR DOSERING

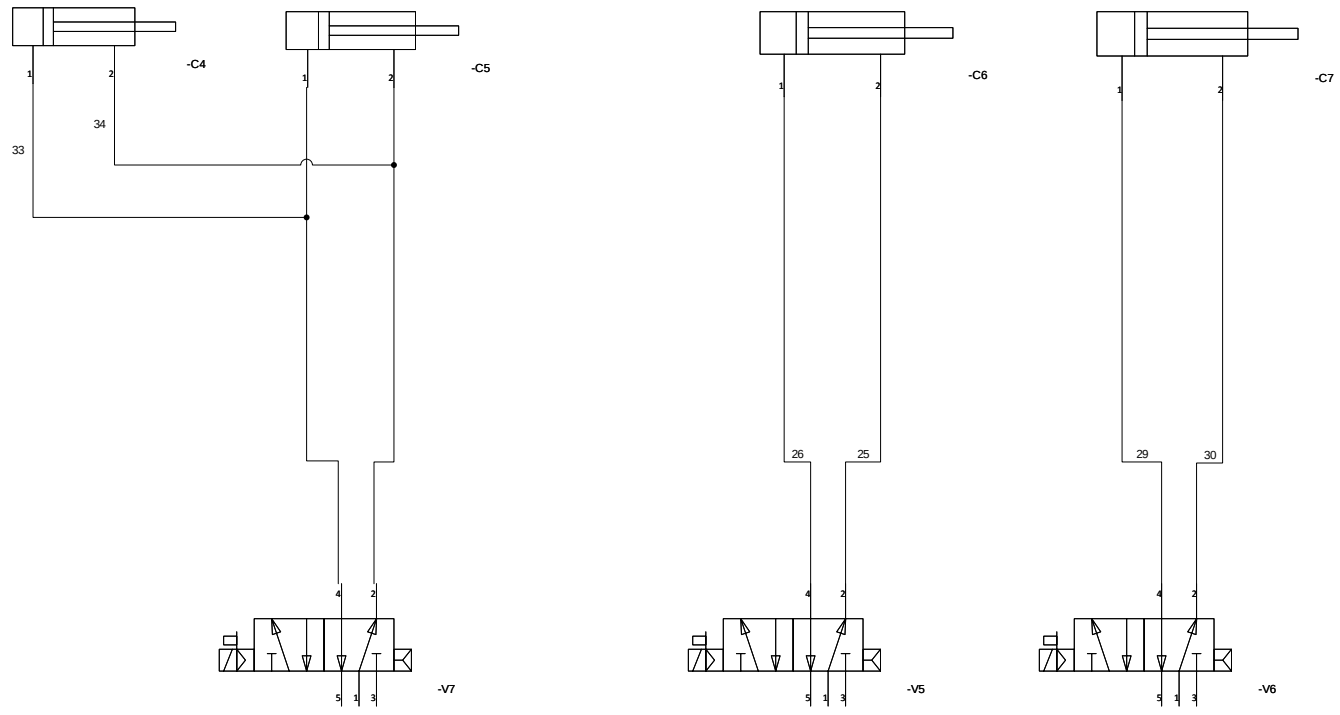


P1 – P4
LOKALISERT PÅ
PANEL

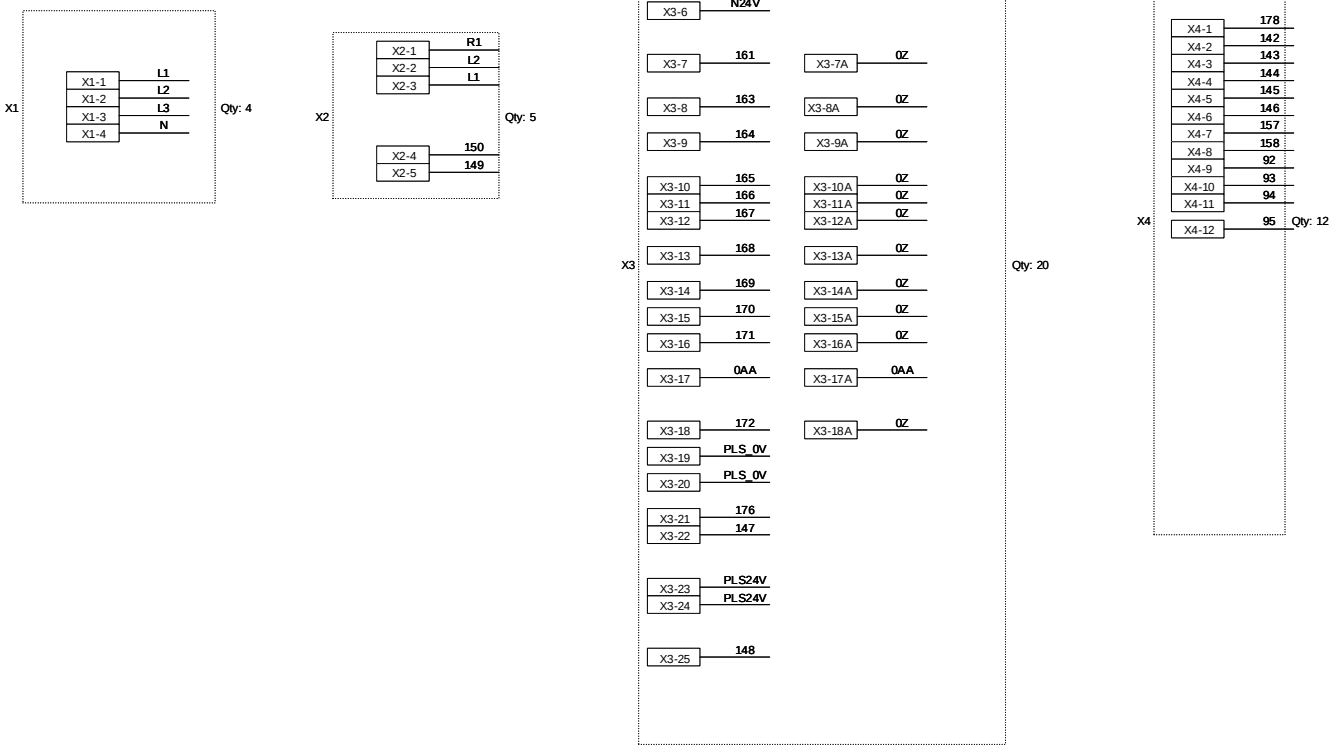
SYLINDER FOR LØFTE
TAPPEHODER



Title	DOSERING	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	11 of 13



Title	FLASKE	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1		
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	12 of 13		



Title	rekke	Project	GOUTHERM	File	skjema.vsdm	Drawing No	FL3	Edited	22 May 2020	Scale	1 : 1
		Client	NOS			Drawn	JB	Created	22 May 2020	Page	13 of 13