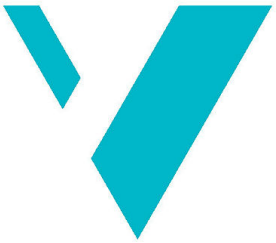


Høgskulen
på Vestlandet

Bacheloroppgave:

BO22EB-06
STYRING AV FESTO MODELLER

Patrick Rindarøy
Fredrik Hundstad
Vegard Valen Lindtveit

29. Mai 2022

Dokumentkontroll

Rapportens tittel: BO22E-06 Styring av Festo modeller	*Dato/Versjon* 29. mai. 2022/R2.0
	Rapportnummer: BO22E-06
Forfatter(e): Patrick Rindarøy Fredrik Hundstad Vegard Valen Lindtveit	*Studieretning:* AUTB19
	Antall sider m/vedlegg 104
Høgskolens veileder: Inge Vivås	*Gradering:* Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: Høgskulen på Vestlandet	*Oppdragsgivers referanse:* Inge Vivås
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): Inge Vivås - 55 58 75 81 - inge.vivas@hvl.no	

Revisjon	**Dato**	**Status**	**Utført av**
R0.0	18.04.2022	Utforming av rapport	VVL
R1.0	23.05.2022	Første utkast	PR/FH/VVL
R1.1	25.05.2022	Revidert utkast, tilbakemelding fra veileder	Inge Vivås
R2.0	29.05.2022	Siste utkast	PR/FH/VVL

Forord

Denne rapporten er skrevet som vår bacheloroppgave ved studiet Automatisering med robotikk på Høgskulen på Vestlandet, campus Bergen.

I løpet av prosjektet har flere utfordringer dukket opp og gitt oss muligheten til å anvende kunnskapen vi har fått i tidligere fag gjennom studiet. Samtidig har vi tilegnet oss ny kunnskap underveis for å løse problemene som har møtt oss på veien. Foruten det faglige har vi lært mye om planlegging og gjennomføring av et langvarig prosjekt, i tillegg til arbeid mot krav og forventninger fra arbeidsgiver.

Vi vil gjerne takke våre veiledere Inge Vivås og Geir Omar Berland for veiledning, tildeling av utstyr og kontor for å ha plass til gjennomføring av arbeidet. I tillegg ønsker vi å takke Endre Håland for hjelp til praktiske oppgaver og innkjøp av utstyr, og Olav Sande for inspirasjon til emulator, hjelp med HMI og diverse koding. En spesiell takk rettes også til 2.års studentene som svarte på vår undersøkelse, inkludert student Fredrik Skavlem som også testet labøvelsene våre og ga gode tilbakemeldinger.

Sammendrag

Høgskulen på Vestlandet har gitt oss en bacheloroppgave der det skulle utformes nye labøvelser i faget ELE304 - PLS programmering. Til dette formålet har det blitt kjøpt inn tre læringsmodeller fra Festo Didactic, kalt Festo MecLab, som vi har lagd labøvelser og løsningsforslag til.

Det er blitt utformet fullstendige labøvelser og tilhørende løsningsforslag, hvorav fire av øvelsene er konstruert for gjennomføring i programmeringsspråket SCL, og to i programmeringsspråket GRAPH. Labøvelsene ble laget for to språk slik at forskjellige studier, med ulik programmeringsbakgrunn, skal klare å gjennomføre dem. Videre ble det også utviklet en instruks til labingeniørene for gjennomføring av labøvelsene, og emulatorer for de fysiske modulene, slik at øvelsene kan løses uten tilgang til PLS, HMI og læringsmodellene.

Alle krav og utvidelser av oppgaven har blitt gjennomført, foruten fjernstyring av modulene. Dette viste seg å være en omfattende utvidelse, og det ble dermed bestemt i tråd med veileder at det ikke var verdt å gjennomføre. Resterende krav er oppfylt som spesifisert av veileder; labøvelsene er skrevet på en forståelig måte, er mulig å gjennomføre på normert tid og HMI-panelet er oversiktlig og anvendelig. Emulatorprogrammene er designet for å samsvare med de fysiske modulene.

Etter fullført prosjekt har vi tilegnet oss ny kunnskap om generell PLS- og HMI-programmering, samt blitt mer erfarne i prosjekt- og dokumentasjonsarbeid, i det å utnytte tiden og delegere arbeidsoppgaver deretter.

Innholdsfortegnelse

1. Innledning

1.1 Organisering av rapport

Omtrent halvparten av innholdet i denne rapporten består av vedlegg, ettersom oppgavens hovedmål har vært å utvikle labøvelser til fremtidige studenter, instruks til labingeniør og løsningsforslag. De forutnevnte dokumentene, samt PLS*-programmer og templater i TIA Portal* vil ligge etter kapittel 7.

Fremtredende ord, forkortelser og begrep markeres med en stjerne *, og vil bli forklart nærmere i Appendiks A - Forkortelser og ordforklaringer.

1.2 Oppdragsgiver

Oppgaven er gitt til oss fra Høgskulen på Vestlandet[1] ved campus Bergen. HVL* ble etablert 1. januar 2017 og med omtrent 17 000 studenter er dette en av de største utdanningsinstitusjonene i Norge. Høyskolen er fordelt på fem campuser på Vestlandet. Det er instituttet for datateknologi, elektroteknologi og realfag som har ansvaret for denne oppgaven. Instituttet er en del av Fakultet for ingeniør- og naturvitenskap, som er fakultetet for alle ingeniørstudenter på HVL på tvers av alle campuser. Våre veiledere ved HVL er Inge Vivås og Geir Omar Berland.

1.3 Problemstilling

Prosjektet går ut på å utvikle labøvelser for PLS-faget, ELE304[2], der læringsmodeller fra selskapet Festo Didactic[3] benyttes. Det er ønskelig at disse modulene kan operere i et samlet system. Øvelsene skal utformes i to forskjellige programmeringsspråk, SCL* og GRAPH*. SCL er et høyklasse-programmeringsspråk som vil rettes mot studenter med høyere programmeringskunnskap. Labøvelsene som fremstilles til GRAPH vil være noe lettere, og rettes mot studenter med mindre programmeringskunnskap samt vil fungere som en introduksjon til PLS.

Det vektlegges at labøvelsene formuleres på en konsis måte, skal være enkle å tolke, og det legges mer fokus på programmering. Ved utforming av øvelsene må den faglige utviklingen til studentene gjennom kurset tas i betraktning. Tilhørende hver labøvelse skal løsningsforslag medfølge. Det må opprettes emulatorer, med templater og tilhørende programkode, for å kunne løse labøvelsene med simulerte modeller i stedet for de fysiske modulene. Instruks til labingeniør skal utformes, der klargjøring av utstyr og kjente feil redegjøres.

2. Kravspesifikasjon

Kravspesifikasjoner er utarbeidet, og diskutert fortløpende, i samarbeid med veileder for å sikre sluttresultat. Derfor tas det forbehold om at enkelte krav har endret seg i løpet av prosjektet.

2.1 Labøvelser med løsningsforslag

Det første og viktigste kravet, som danner grunnmuren for hele bacheloroppgaven, er å lage minimum en labøvelse med løsningsforslag til hver enkelt modul vi har fått. Dette medfører at vi må lage løsningsforslag i det respektive programmeringsspråket for de oppgavene vi lager, både i SCL og GRAPH.

Se Appendiks C - Labøvelser.

2.2 Instruks til labingeniør

Et annet krav er å lage en utfyllende instruks til labingeniørene. Dette inkluderer hva som må gjøres i forkant av, underveis og etter øvelsene. Det viktigste vil være å lage en klar ramme for hva som er forventet av studentene under gjennomføringen av labøvelsene, og formidle dette på forståelig vis i instruksen slik at labingeniøren kan sørge for at dette blir gjennomgått. Det bør også gjøres rede for potensielle problemer som kan oppstå og løsninger på disse.

Se Appendiks D - Instruksjon til labingeniør.

2.3 HMI-panel

Oppdragsgiver uttrykte et ønske om HMI*-panel som forenkler og muliggjør kommunikasjon mellom PLS og moduler. Målet vil være å lage et enkelt og lettlest templat til HMI-skjermen. Dette for at kjøring av modulene, ved bruk av studentenes programkode, vil fungere intuitivt og nært arbeidslivet. HMI-panelet skal kunne brukes til å gjennomføre alle modulene. Panelet vil monteres til platen modulene er festet på og skal enkelt kunne fjernes om ønskelig.

2.4 Simulering

Mulighet for gjennomføring av labøvelser uten tilgang til fysiske moduler eller PLS ble etterspurt av veileder. Dette skulle løses gjennom bruk av emulator-program og animasjoner av de gitte modulene. Emulatorene skal utvikles slik at innganger og utganger til modulene manipuleres i “Default tag table” gjennom emulatorprogrammet til hver enkelt modul. På denne måten vil emulatorene ha tilnærmet lik virkemåte som de fysiske modellene, og vil fungere som en fysisk modul sammen med en student sitt program. Det er ønskelig at emulatorene som utformes er inspirert av de som brukes i faget per våren 2022.

2.5 Fjernstyring

Oppdragsgiver forespurte fjernstyring av systemet gjennom HMI-panelet eller VPN, men ble påpekt lavest viktighetsgrad. Ønsket var hovedsakelig å kunne styre PLSen og modulene uten å være fysisk til stede, enten i undervisningsøyemed eller ved styring hjemmefra. Dette blir kun fokusert på om tiden strekker til, da de foregående ønskene har høyere prioritet.

3. Analyse av problemet

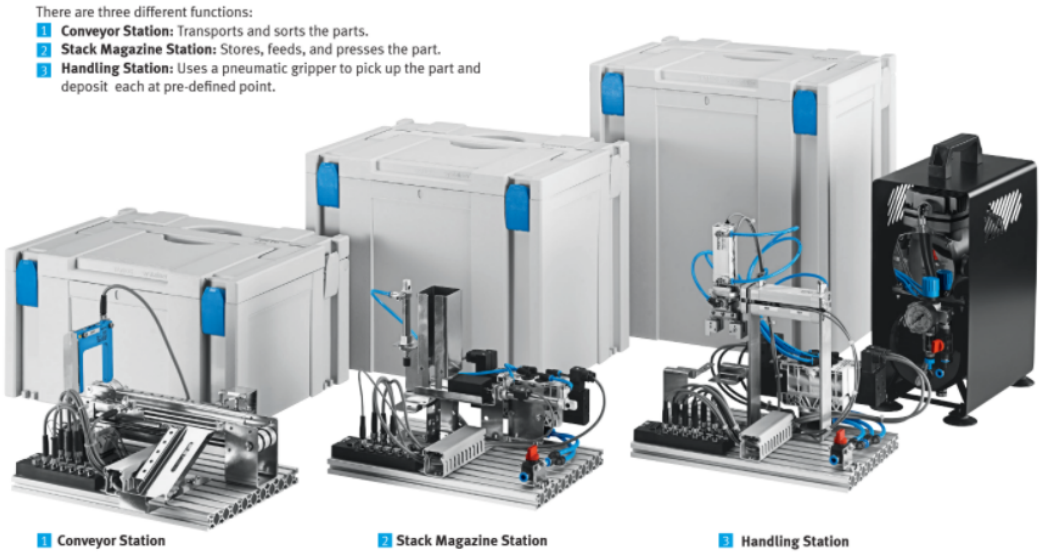
Det vil i dette kapittelet gjennomgås grunnlaget for hovedoppgaven ved prosjektet, og hvordan labøvelsene som blir utviklet bygger på tidligere øvelser i faget ELE304. Det vil også gis et innblikk i studentenes tanker og ønsker til disse øvelsene. Mulige løsninger for gjennomføring av prosjektet vil drøftes, før en begrunnet konklusjon avslutter dette kapittelet.

3.1 Hovedoppgave

Hovedoppgaven til dette prosjektet består av flere elementer. Det skal opprettes labøvelser der Festo læringsmodeller tas i bruk, løsningsforslag (PLS-kode) til disse, samt en instruks til labingeniør der det skal beskrives hvordan øvelsene bør gjennomføres på best mulig vis. Det er viktig at øvelsene vil gi tilfredsstillende læringsutbytte til studentene, og er en forbedring på de nåværende labøvelsene. Dette blir gått dypere inn på under 3.2 Forbedring av tidligere labøvelser.

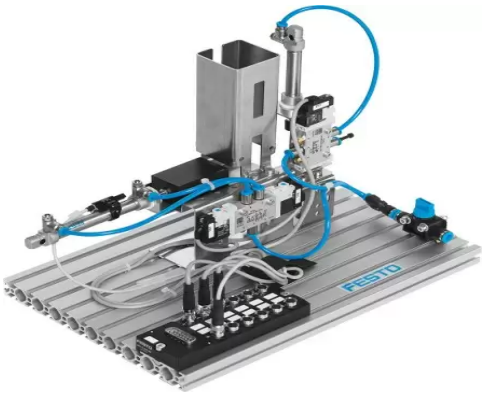
Modifisering og montering av modulene vil også være en omfattende del av oppgaven. Dette er et viktig aspekt av prosjektet som vil bli beskrevet ytterligere under 4.3 Modifikasjoner.

3.1.1 Festo Meclab-moduler



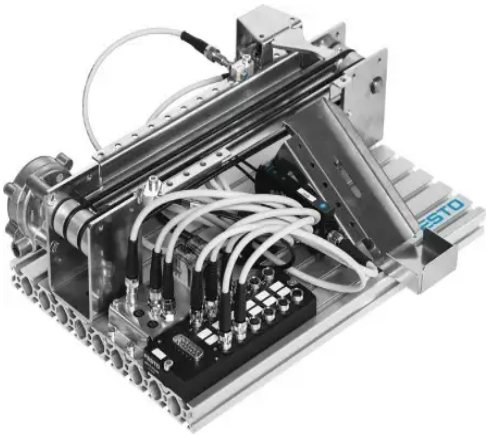
Figur 1 - MecLab Mechatronics Training System

HVL har tildelt oss tre moduler fra Festo Didactic, i en pakke kalt Festo MecLab[4], som det skal lages labøvelser og løsningsforslag til. Målet er å opprette separate oppgaver til hver enkelt modul, der sluttresultatet vil bli et helhetlig system som skal kunne kjøres sammen. Emneplanen i faget må tas i betraktning for å kunne lage en stigende læringskurve for studenter under gjennomføring av labøvelsene. Modulene skal styres av en Siemens S7-1500 PLS som vi har fått tildelt av HVL.



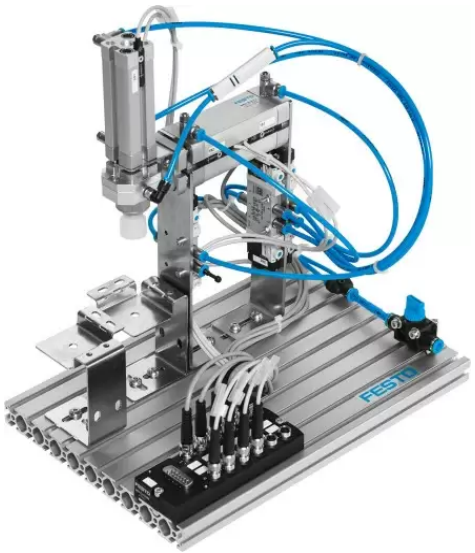
Figur 2 - Stacking Module

Stacking-modulen[5] består i hovedsak av to stempler. 1A som operer horisontalt og 2A som opererer vertikalt. Disse styres av pneumatiske solenoidstyrte retningsventiler i varierende konfigurasjon. Det må også nevnes at 2A er fjærbelastet slik at det vil være i bakre stilling så lenge det ikke settes trykk på den ene inngangen dens. Posisjonsdetektor 1S1, en sensor plassert bak på stempel 1A, gir tilbakemelding når stempelet er i bakre posisjon. Stacking-modulen sitt formål er å skyve ut en puck og presse den sammen med en tilhørende underdel.



Figur 3 - Conveyor Module

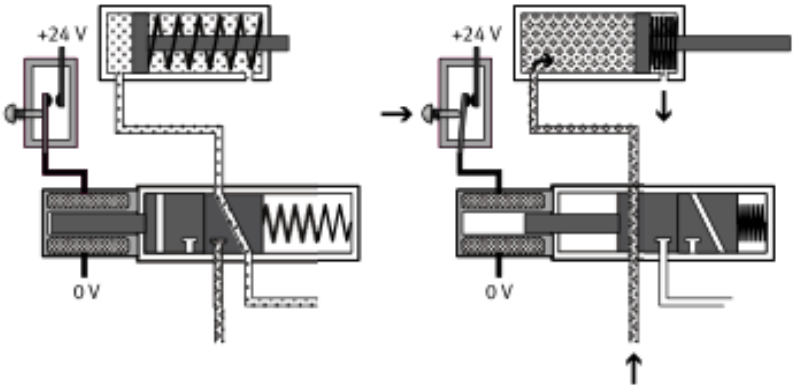
Conveyor-modulen[6] baserer seg på et relestyrt samlebånd, der pucker av to forskjellige materialer vil sorteres fra hverandre. Det er også montert en optisk sensor (OPT) og en induktiv proximity sensor (IND), der OPT er plassert i starten av samlebåndet og IND er plassert på midten. En deflektor (DEF), styrt av en solenoid* er plassert ved siden av den induktive sensoren. Denne har som funksjon å lede objekter ut på en rampe på siden av samlebåndet. Det brukes to releer: K1 for å kjøre motoren til samlebåndet, og K2 for å kjøre i revers. K2 må være aktiv samtidig som K1 for at samlebåndet skal kjøre i revers.



Figur 4 - Handling Module

Handling-modulen[7] har lignende konfigurasjon som Stacking-modulen. Stempel 3A operer horisontalt, stempel 4A operer vertikalt og er montert på enden av 3A. På denne måten beveger de to stemplene seg sammen, noe som må tas ekstra hensyn til ved kjøring av modulen. Det er to sensorer montert på 3A; 3S1 som registrerer om stempelet er i indre posisjon og 3S2 som registrerer om det er i ytre posisjon. Det samme gjelder for stempel 4A, her registrerer 4S1 indre posisjon og 4S2 ytre posisjon. På enden av stempel 4A er det en vakuumgriper 5A. Stemplene og vakuumgriperen styres også av pneumatiske solenoidstyrte retningsventiler. Hovedfunksjonen til denne modulen er å løfte en puck fra en plattform, flytte den til en annen og slippe den av.

Posisjonen til stemplene styres ved bruk av solenoidstyrte retningsventiler. Disse har varierende konfigurasjon av innganger og utganger, men prinsippet er å lede lufttrykket til riktig inngang på stempelet. Ved signal inn på solenoiden, på en side av ventilen, endres posisjonen til retningsventilen og luft tilføres slik at stempelet beveges. Se bilde nedenfor som illustrerer hvordan ventil 2M1 styrer 2A på Stacking-modulen.



Figur 5 - Illustrasjon av ventil 2M1 og stempel 2A

3.1.2 Utforming av labøvelser

Utformingen av labøvelsene må være på en måte som gir studentene best mulig læring. Det bør i stor grad legges vekt på at studenter skal kode selv og bli godt kjent med TIA Portal. Slik det har vært tidligere skulle studentene skrive minimalt med egen kode før siste øvelse, og fokusere på å koble eksisterende funksjonsblokker* (FB) sammen, for så å observere hva som skjer. Dette mener vi ikke gir tilstrekkelig læringsutbytte.

Labøvelsene burde være slik at studentene gjør mest mulig selv og det vil ikke ligge ferdig kode i templatet. Det er også ønskelig at koden som skal utvikles i stor grad vil være praktisk, da dette skiller PLS-programmering fra annen programmering studentene har vært igjennom. Største delen av labøvelsene bør være koding, men det burde også legges til noen teorispørsmål som kan bygge studentenes grunnforståelse rundt verktøyene de har tilgjengelig for å løse oppgavene.

Det er viktig at labøvelsene er beskrevet på en konsis måte for å unngå misforståelser. Av egen erfaring er det vanlig å mistolke oppgaver som er beskrevet i labøvelsene. Ved for mye misforståelser vil muligens studenter bli demotivert og risikoen for at ikke alle studenter rekker å gjennomføre labøvelsene på normert tid kan øke.

Tilhørende hver labøvelse er det nødvendig med løsningsforslag. Disse burde vise god programmeringsskikk og effektive løsninger, i tillegg til gode svar på teorispørsmål. Det kan være hensiktsmessig å skrive kodeforslag før selve oppgavetekstene blir utformet, slik at gruppen har en løsning i tankene når oppgavene skrives. Dette vil kunne føre til at fallgruver ved programmering kan dekkes gjennom en god oppgavetekst.

Det ble stilt krav til innhold i en instruks til labingeniør fra veiledere, som nevnt under 2.2 Instruks til labingeniør. Denne bør inkludere en konsis gjennomgang av øvelsene og introduksjon til de medfølgende PLS-programmene. Informasjon om gjentakende feiltakelser, problemer og medfølgende løsning på disse vil være hensiktsmessig å merke seg.

3.2 Forbedring av tidligere labøvelser

Motivasjonen bak dette prosjektet var å forbedre utformingen av labøvelsene som de er per våren 2022. Opphavet for denne motivasjonen kommer fra hvor lite læringsutbytte vi selv opplevde gjennom øvelsene som har vært. For å forsikre oss om at andre studenter delte vår oppfatning, sendte vi ut en spørreundersøkelse til dagens 2.års studenter for å se hvordan de opplevde labøvelsene. Antall studenter som deltok i undersøkelsen var på elleve personer.

Hva synes du om ELE304 PLS-programmerings faget?			
Spennende og lærerikt	Spennende men mangler læringsutbytte	Bra faglig men uinteressant	Uinteressant
0	11	0	0

Hvor godt læringsutbytte opplever du fra labøvelsene i faget?				
Veldig bra	Bra	Helt ok	Dårlig	Veldig dårlig
0	1	3	4	3

Synes du at faget har blitt lagt opp på en måte som gir deg en tilstrekkelig læringskurve?		
Ja, på en god måte	Middels	Nei, jeg opplever tilnærmet ingen læringskurve i faget
0	5	6

Får du nok erfaring med egen programmering av PLS i faget?	
Ja	Nei
2	8

Har du fått jobbe med en fysisk PLS?	
Ja	Nei
6	5

Synes du at faget hadde blitt mer interessant og lærerikt om det ble lagt opp til arbeid med fysiske modeller/moduler?	
Ja	Nei
11	0

Føler du at å jobbe med en fysisk PLS ville hjelpe med din forståelse i faget?		
Ja	Nei	Ingen formening
10	0	1

Figur 6 - Spørreundersøkelse om ELE304

Tilbakemeldingen ga inntrykk av at flere studenter deler vår erfaring. Faget har potensiale, men labøvelsene som benyttes per våren 2022 gir ikke tilstrekkelig læringsutbytte. Undersøkelsen viser at studenter ønsker å ta i bruk fysiske læringsmodeller og mer praktisk arbeid i faget ELE304. Det er dette labøvelsene som utvikles i dette prosjektet vil gjøre. Sannsynligvis må faget også gjennomgå enkelte modifikasjoner, slik at grunnkunnskapen for å gjennomføre labøvelsene vil være mer tilstrekkelig. Det må også nevnes at studentene som gjennomførte undersøkelsen hadde deler av faget under Covid-19 pandemien, som trolig vil ha påvirket responsen til studentene. Formatet til spørreundersøkelsen er endret for å bedre passe inn i rapporten. Spørreundersøkelsen er lagt ved som pdf-fil i Appendiks E - Andre filer.

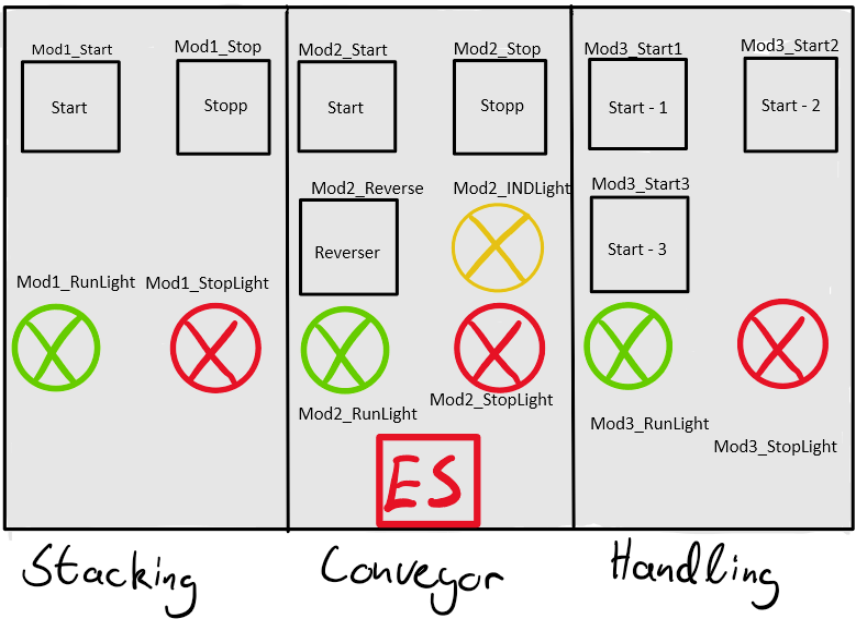
3.3 Mulige løsninger

Det skal utformes minst tre labøvelser, der studentene skal styre Festo-modellene ved hjelp av en Siemens S7-1500 PLS og en Siemens TP700 HMI-skjerm. Mulig løsning på dette er én enkelt øvelse for hver modell, i tillegg til en øvelse der alle modellene blir satt sammen til å utføre en sammenhengende prosess. Oppgavene skal kunne løses både med de fysiske modellene og med emulator. Det er viktig at disse samsvarer slik at det ikke fører til unødvendig forvirring for studentene. Det bør derfor lages emulatorer for å simulere hver enkelt modul.

Hver labøvelse skal utvikles for både SCL og GRAPH. Vi har ikke fått noen klare retningslinjer fra veilederne våre på hvordan oppgavene skal utformes, eller hva de forskjellige modulene skal gjøre. Med utgangspunkt i dette, er det naturlig at oppgavene går ut på å styre modulene slik det står beskrevet i “563063_Teaching_With_MecLab” som fulgte med MecLab-pakken Festo. For eksempel er den ene modulen et lite samlebånd for små pucker med en induktiv sensor og en solenoidstyrt deflektor som dytter vekk pucker fra samlebåndet. På denne modulen vil en naturlig oppgave være å sortere plast- og metallpucker.

Oppgaver rundt fysisk kobling mellom PLS og modulene for studentene kan også være en mulighet, slik at de får praktisk erfaring med kobling av utstyr mot PLS. Det er lite praktisk arbeid gjennom studiet, og dette er noe som studentene ønsker mer av, jamfør undersøkelsen som har blitt gjennomført (se 3.2 Forbedring av tidligere labøvelser). Implementasjon av kobling vil kunne gi studenter en bredere forståelse for faget og et bedre innsyn i hvordan PLS, sensorer og aktuatorer henger sammen.

Det skal også designes et HMI som skal brukes til å styre modulene. Her ønsker vi både å kunne bruke et fysisk HMI-panel fra Siemens, det vil si en touch-skjerm, og et simulert panel i TIA. Det må derfor lages templater for dette. Disse vil i all hovedsak bestå av driftlys, start- og stoppknapper. Enten må det utformes en samlet skjerm for å styre alle modulene, eller et panel til hver modul. Å bruke et HMI-panel vil være mer intuitivt og nærmere arbeidslivet, fremfor å manipulere verdien på innganger og utganger med små rubrikker gjennom instansdatablokker* (IDB) i TIA Portal.



Figur 7 - Skisse av HMI-templat

For at labøvelsene skal ha stigende vanskelighetsgrad, må det vurderes hvilken rekkefølge de tre modulene skal inkorporeres i øvelsene. Conveyor-modulen vil trolig være et logisk sted å starte da man unngår at studentene må arbeide med pneumatikk, som vil være nytt for mange, i første øvelse. Deretter burde i så fall Stacking-modulen jobbes med før Handling-modulen. Dette ettersom disse modulene er meget like, men Handling-modulen har et større antall sensorer og aktuatorer. Hvordan modulene vil monteres på kryssfinerplaten vil også være en avgjørende faktor for rekkefølgen modulene brukes i labøvelsene. I "563063_Teaching_With_MecLab" er rekkefølgen Stacking->Conveyor->Handling ved gjennomføring av øvelser til modulene. Med denne rekkefølgen vil man også spre labøvelsene rundt Stacking- og Handling-modulen over et større tidsrom, og studentene unngår å løse to meget like labøvelser etter hverandre.

3.4 Valg av løsning

Det skal utformes minimum en labøvelse for hver modul. I samråd med veileder har det blitt bestemt at det kun skal produseres to labøvelser i GRAPH, til Stacking- og Conveyor-modulene, da GRAPH-programmering vil fases ut i årene fremover. Dette resulterer i fire oppgaver med løsningsforslag for SCL og to for GRAPH, altså totalt seks oppgaver med tilhørende løsningsforslag. For de fire labøvelsene i SCL, vil labøvelse 1 ta for seg kommunikasjon og testing av HMI. Labøvelse 2, 3 og 4 vil ta for seg modulene Stacking, Conveyor og Handling, i denne rekkefølgen. Det har blitt konkludert med at dette vil være den optimale rekkefølgen for studenters læring, da vanskelighetsgraden vil øke gradvis for hver labøvelse, i tillegg til at man møter forskjellige utfordringer. Som nevnt tidligere er dette også rekkefølgen som Festo selv bruker i øvelsene de har produsert. Modulene vil monteres Handling, Conveyor til Stacking fra venstre til høyre. Det vil si at labøvelsene beveger seg gjennom modulene fra høyre til venstre. Etter gjennomført arbeid med to eller tre moduler, vil man kjøre de foregående modulene sammen.

PLS-programmene som studentene skal utvikle i øvelsene vil være praktisk kode. Det er ønskelig at studentene skal lære mest mulig om hvordan anvende forskjellige programmeringsprinsipp, og forbedre forståelsen for programmering av PLS. Mindre fokus legges på logging, navngivning av variabler og standardiserte normer, da dette er tidkrevende, og ikke nødvendigvis øker studentenes programmeringsferdigheter. Dette aspektet kan fokuseres på i andre deler av faget.

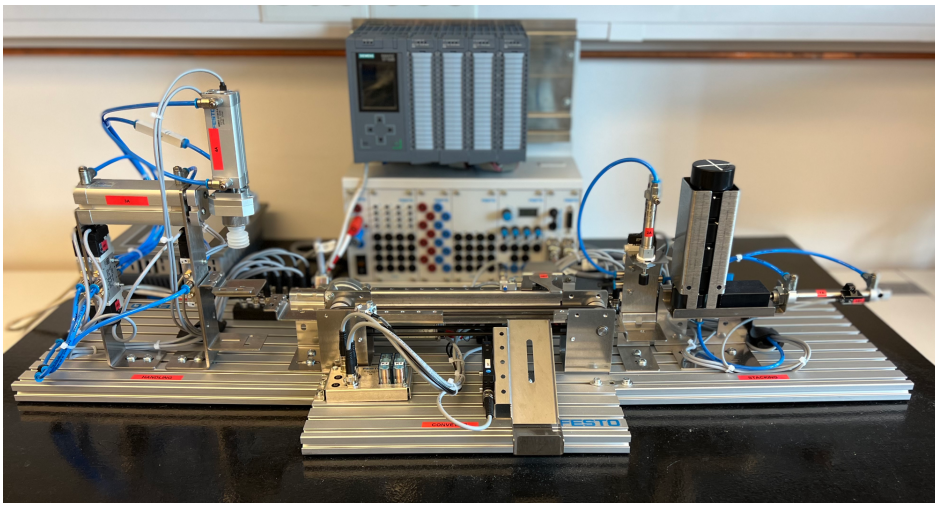
Installasjon og bruk av HMI var først og fremst et ønske fra veiledere, fremfor et krav, men for å gjennomføre labøvelsene slik vi ønsker, vil et HMI-panel være en nødvendighet.
Det vil derfor utformes templat for HMI i TIA Portal. Denne skal brukes til å styre modulene og vil bli utformet som beskrevet i forrige delkapittel, med en HMI-skjerm der alle knapper og lys vil være fordelt på tre segmenter av skjermen.

Tidsrammen studentene har til å gjennomføre øvelsene er fire skoletimer, som tilsvarer tre klokketimer. Dette medfører at øvelsene må legges opp slik at studenter klarer å gjennomføre innenfor planlagt tidsbruk. Øvelsene kommer ikke til å være bygd opp slik at studenter må koble opp fysiske kabler selv, men vil være ferdigkoblet når de kommer på labben. Det hadde vært ønskelig at alle studentene får prøvd seg på å koble opp PLS mot fysisk utstyr, ved bruk av kabler med bananplugger, for å unngå slitasje på utstyr. Det har også kommet tilbakemelding fra veileder om at dette vil kreve mer tid og praktisk kunnskap enn det studentene innehar. Ettersom koblingen mellom PLS og modulene foregår gjennom datakabler til multi-pin* tilkoblingene i dette prosjektet, kan det diskuteres om læringsutbytte vil være tilstrekkelig.

Fjernstyring gjennom HMI-panel, eller ved hjelp av VPN, er noe vi ikke kommer til å fokusere på. Dette blir nedprioritert ettersom det må tas hensyn til hver enkelt student sin situasjon i forhold til nettverk, personvern og lignende (GDPR*), samt at arbeidsmengden vil strekke seg utover vårt arbeidsomfang. Det åpner også som nevnt muligheten for at utstyret kan bli hacket av folk som ikke skal ha tilgang. Et annet argument mot å implementere fjernstyring er at det ikke vil være mulig å legge puckene på modulene eller skru på kompressor og stengeventiler. Det vil si at dersom en skulle fjernstyrt det, måtte det uansett vært noen til stede for å overvåke at de fysiske komponentene er tilrettelagt for gjennomførelse.

4. Realisering av valgt løsning

Under dette kapittelet vil realiseringen av løsningene valgt i forrige kapittel beskrives. Utforming av labøvelsene, med tilhørende løsningsforslag og emulatorer, samt instruksen til labingeniør gjøres rede for. Diverse modifiseringer og montering av utstyr vil også presenteres.



Figur 8 - Modulene i et sammensatt system

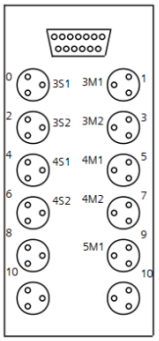
4.1 Labøvelser og utstyr

Labøvelsene er kjernen i vårt bachelorprosjekt, og mye av tiden har gått med til å utforme disse på en god og forståelig måte. Riktignok måtte det først kontrolleres at modulene fungerte som beskrevet i dokumentasjonen fra Festo. PLSen som ble tatt i bruk til dette prosjektet var en Siemens S7-1500 (CPU 1516F-3 PN/DP), med tilhørende DI*, DQ*, AI* og AQ*-moduler. Første steg i denne prosessen ble å koble modulene til PLSen, komplett med Festo EduTrainer[8]. Dette ble gjort ved å koble Festo-datakabler fra Syslink* port A og B på EduTrainer til Sub D*-inngangen på multi-pin tilkoblingene. Deretter sjekket vi kablingen mellom multi-pin tilkoblingen og de forskjellige komponentene. Informasjonen om inngangene og utgangene, fra komponentnivå til PLS, ble skrevet ned i skjemaer og ble lagt ved i hver labøvelse. Se bilde under for eksempel hentet fra "Lab 4 - Handling Module".

Innganger og utganger til Handling-modulen (ta i bruk "Tag" i koden):

PLS-Inputs	Multipin-Inputs	Komponent	Tag
I1.0	0	Sensor 3A, indre posisjon	3S1
I1.1	2	Sensor 3A, ytre posisjon	3S2
I1.2	4	Sensor 4A, indre posisjon	4S1
I1.3	6	Sensor 4A, ytre posisjon	4S2

PLS-Outputs	Multipin-Outputs	Komponent	Tag
Q1.0	1	Solenoid 3A, ytre posisjon	3M1
Q1.1	3	Solenoid 3A, indre posisjon	3M2
Q1.2	5	Solenoid 4A, ytre posisjon	4M1
Q1.3	7	Solenoid 4A, indre posisjon	4M2
Q1.4	9	Solenoid 5A, vakuum på	5M1



Figur 9 - Inngang/utgangs-skjema

Etter oppkoblingen mellom PLS og moduler var gjennomført kunne det observeres at sensorene fungerte som ønsket, med noen avvik (les mer under 4.3.1 Modifikasjon av moduler). Under testing av Conveyor-modulen ble funksjonen til samlebåndet og deflektoren kontrollert mot virkemåten spesifisert i medfølgende dokumentasjon. Den eneste feilen som ble oppdaget var at sensoren OPT måtte flyttes og vinkles om, slik at pucker ble observert ved starten av samlebåndet. Pneumatikken på Stacking- og Handling-modulen ble ikke testet på dette stadiet, ettersom vi hadde problemer med kompressoren og derfor ikke fikk satt trykkluft på anlegget. Til tross for dette ble solenoidene på alle retningsventiler testet og så ut til å fungere som tiltenkt.

Da vi hadde kontrollert virkemåten til komponentene, lagde vi noen primitive testprogram som senere ble videreutviklet til løsningsforslagprogram. Dette gjorde det også lettere å se hva minstekravet for at modulen skulle utføre sin funksjon var. Ettersom vi fremdeles ikke hadde trykkluft, måtte en på gruppen fysisk bevege stemplene til ønsket posisjon.

Det ble også produsert templat til HMI-skjermen for lettere å styre modulene. HMIen som ble supplert av veileder var av typen Siemens TP700 Comfort (Modellnr. 6AV2 124-0GC01-0AX0). Templatet er utformet for å være lettleselig og anvendelig. Se mer om utfordringene ved utvikling under 6.2.1 HMI-panel.

Figur 10 - Bilde av HMI-skjerm

4.1.1 Oppgaveark

Labøvelsene er laget med utgangspunkt i malen som HVL tidligere har brukt til labøvelser i faget ELE304, men det har blitt foretatt enkelte endringer. Oppgavearkene som har blitt utviklet gjennom prosjektet har ikke rubrikker der man skal lime inn programkode, da dette er lite intuitivt for både student og labingeniør. I stedet legges det opp til at studentene laster opp sitt TIA-prosjekt (i zippet mappe) til Canvas sammen med oppgavearket med besvarte tekst- og teorispørsmål.

Det har etter beste evne blitt å skrive så formålsrettet og lettleselig som mulig, slik at det ikke oppstår forvirring for studentene. Dersom oppgaveteksten blir for lang eller innviklet, kan studentene miste overblikket eller overse vesentlige detaljer for krav til gjennomføring. Oppgavene ble nøye gjennomgått og testet, også av ekstern testperson, slik at oppgavene har blitt kvalitetssikret etter kravene listet tidligere i avsnittet.

Tiden som beregnes per labøvelse er tre timer. Hvor mye tid studentene i realiteten bruker kommer til å variere fra øvelse til øvelse, da programmene som skal konstrueres vil bli gradvis større. Planen er at studentene vil ha mer kunnskap og erfaring med PLS-programmering når de skal utvikle større program, og derfor vil klare å løse disse øvelsene på normert tid.

Oppbygningen til øvelsene er som nevnt inspirert av øvelsene tidligere brukt i ELE304-faget, og vil være konstruert som følger.

Forside: Her står navnet på oppgaven, sammen med rubrikk der studentene skriver inn navn og gruppenummer. Det vil også nevnes formål med øvelsen, hvordan den skal gjennomføres og hvilket utstyr som trengs, beskrevet i stikkordsform.

Innføringsinfo: Denne delen har vi forsøkt å holde så kort og konsis som mulig, slik at studentene ikke vil droppe å lese den. Det gis en kort introduksjon av modulen som brukes, dens virkemåte, og informasjon om de forskjellige programmeringskonseptene som bør brukes i koden som skal konstrueres i oppgavene. Skjema over innganger og utganger som skal tas i bruk og hvor disse er koblet til på multi-pin tilkoblingene foreligger under innføringsinfo, sammen med bilde av modulen som skal tas i bruk.

Oppgavetekst: Her gjentas i korte trekk det som ble beskrevet i innføringsinfo, deretter beskrives hvordan programmet skal utvikles punktvis. Dette vil gjøre det lettere for studentene å teste og feilsøke underveis i programutviklingen. Hint og tips er også lagt inn i oppgaveteksten, slik at ikke studenter setter seg fast ved simple aspekter som hvor man finner ønskelige blokker og lignende.

Testing med fysiske moduler eller emulator: Det skal være mulig å teste programmet sitt både med fysiske moduler eller emulator, og da må man gå frem på to forskjellige måter. Hvis man tester med fysisk modul har det blitt beskrevet hvor man skal legge eventuelle pucker, og hvordan man tester de forskjellige funksjonene utviklet i programmet. Ved test gjennom emulator må man operere litt annerledes, og hvordan man da skal gå frem står beskrevet. Det vil også foreligge en enkel beskrivelse av hvordan emulatoren fungerer, slik at man enda tydeligere skal se sammenhengen mellom de fysiske modulene og emulatorene.

Teoridel: Labøvelsene som skal løses med SCL-programmering vil ha omtrent tre teorispørsmål etter at studentene har utviklet et fungerende program. Disse spørsmålene vil ta for seg noen av konseptene eller virkemåten til komponenter brukt i den gitte labøvelsen. GRAPH-øvelsene vil ikke ha denne teoridelen.

Levering: Siste side av rapporten beskriver hvordan man leverer besvarelsen av labøvelsen på Canvas. Besvarelsen skal ikke leveres før labingeniør har godkjent den.

Det tas forbehold om at de forskjellige labøvelsene vil være skrevet på litt forskjellig vis, men i all hovedsak vil de være bygd opp som beskrevet over.

Se Appendiks C - Labøvelser.

4.1.2 Løsningsforslag

Løsningsforslagprogrammene ble utviklet parallelt med labøvelsene, slik at eventuelle utfordringer og fallgruver ville vise seg mens oppgavene i øvelsene ble skrevet. Det ble også enklere å vite hvordan optimalt geleide studentene gjennom øvelsene, når vi har forsøkt å løse øvelsene selv. Da program til styring av de forskjellige modulene var ferdig konstruert ble det foretatt revisjoner av disse, for at de skulle stemme overens med oppgaveteksten i øvelsene.

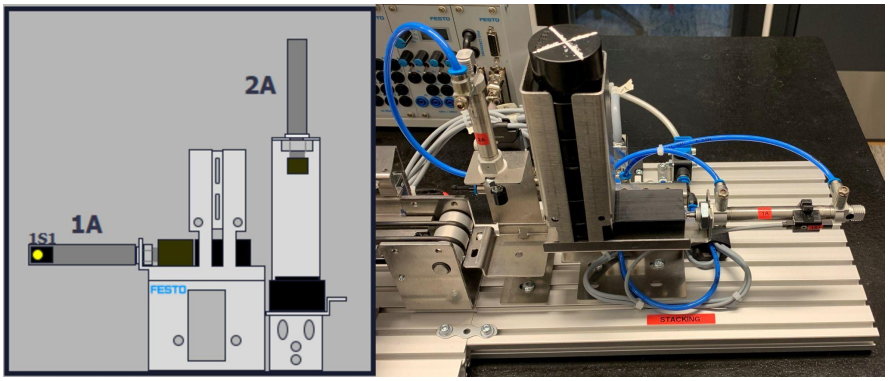
Disse programmene ble stort sett skrevet uten vanskeligheter, selv om gruppen ikke har arbeidet med PLS på ett år. Løsningsforslaget til Handling-modulen viste seg å være utfordrende og krevde en del feilsøking før et ferdig program ble konstruert. Det er som nevnt ikke fokusert mye på logging, navngiving av variabler og standardiserte normer noe som gjenspeiles i løsningsforslagprogrammene.

Løsningsforslagene til programmeringsdelen av øvelsene er kun en av mange løsninger, da det vil være flere måter å skrive programkode på som gir samme resultat og virkemåte. Det anbefales å gi løsningsforslagprogrammene til studentene i etterkant av øvelsene. Disse kan også brukes av studenter som ikke har løst øvelsene på optimalt vis, eller ikke har gjennomført alle deloppgavene i øvelsene. Med dette hjelpemiddelet kan de lære av sine feil, og opparbeide seg et bedre utgangspunkt til neste labøvelse.

Teoridelen, det vil si teorispørsmålene, på slutten av hver lab (gjelder ikke for GRAPH-øvelsene), vil ha løsningsforslag som en del av instruksen til labingeniør. Det er ønskelig at teorispørsmålene, og løsningsforslaget til disse, revideres årlig slik at de til enhver tid vil samsvare med emneplanen.

4.1.3 Emulatorer

Vi har konstruert emulatorer for alle modulene og produsert dem slik at de er så like de fysiske modulene som mulig. Som tidligere nevnt i 2.4 Simulering, la veileder frem et ønske om at emulatorene skulle utformes inspirert av emulatorene som brukes i faget per våren 2022. Dette har blitt gjennomført etter beste evne, men emulatorkoden som vi har utviklet vil være noe simplere. Det har derimot blitt lagt mer arbeid i konstruksjon av animasjonene i HMI-templatene, slik at det skal være mulig å se betydelig likhet mellom fysisk modul og animert modul. Se bilde under av Stacking-modul, animert og fysisk.



Figur 11 - Stacking modul (animert og fysisk)

Emulatorene for de tre modulene vil simulere alle sensorer, stempler og andre aktuatorer. Inngang- og utgangsverdiene til disse vil da endres i “Default tag table” til PLSen i TIA Portal ut ifra hva som skjer i programmet som studentene utvikler. Studentene kan da bruke alle innganger og utganger som går til de fysiske modulene, og disse vil endre verdier og samhandle som i virkeligheten, gjennom emulatorene. I tillegg vil forskjellige animasjoner utføres ved endring i verdi hos en sensor eller aktuator. For eksempel hvis releet som styrer motoren til samlebåndet (K1) settes høy, begynner det simulerte samlebåndet å kjøre, og pucken vil bevege seg om den har blitt lagt på båndet. Det har også blitt lagt til lys på alle sensorer som indikerer om de har høy verdi. Dette er også lagt til på vakuumgriperen 5A, for å bedre kunne overvåke dens tilstand. Les mer om dette under 5.3 Test av emulator.

4.2 Instruks til labingeniør

Instruksen skal gi labingeniørene et godt utgangspunkt for å veilede studenter gjennom labøvelsene uten å selv ha gjennomført øvelsene. Selve instruksen er delt opp i flere deler. Forarbeid er den delen som må gjøres klart før studentene kommer, slik at det blir brukt minst mulig tid på oppsett. I instruksen foreligger en anbefalt gruppestørrelse på tre personer, i samsvar med anbefaling fra “563063_Teaching_With_MecLab”. Det har også blitt redegjort for en del kjente feil med løsninger på disse, slik at man ikke vil miste tid på feilsøking. Ettersom kompressoren kan virke lite intuitiv er en liten bruksanvisning for denne fremstilt.

Det gås dypere inn på hvordan templatet til labøvelsene, “Mal - MecLab”, er utviklet og hva det inneholder. Labingeniør bør studere denne malen i TIA Portal for å kunne assistere studenter under gjennomføring av øvelsene. Virkemåten til emulatorprogrammene beskrives også mer utfyllende i dette dokumentet. Ettersom funksjonsblokkene med emulatorkode er skrivebeskyttet, vil passordet for tilgang være notert her.

Et kort sammendrag av hver labøvelse er beskrevet, slik at labingeniør ikke behøver å lese gjennom hele oppgavearket før hver eneste gjennomføring. Her vil også virkemåten til modulene bli kort beskrevet.

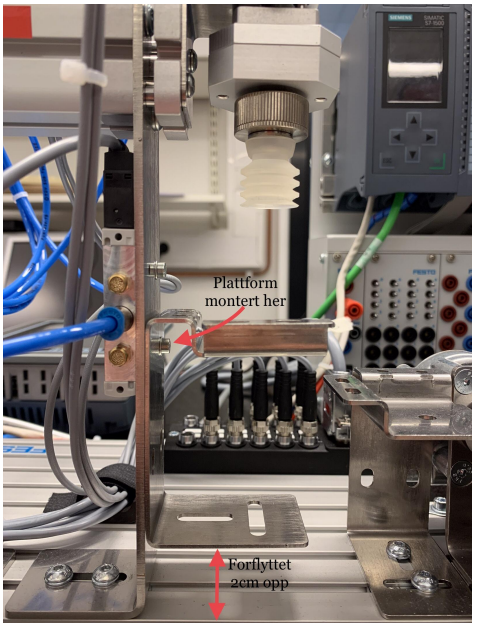
Til slutt beskrives det avsluttende arbeidet for labingeniøren når øvelsen er fullført. Dette går hovedsakelig ut på gjennomgang av øvelsen med studentene som har fullført, for å se at alt stemmer. Det er også viktig å stenge alle ventiler, slå av og tømme kompressoren, samt fjerne program som er lastet opp til PLSen.

4.3 Modifikasjoner

Dette delkapittelet tar for seg uoverensstemmelser mellom dokumentasjon fra Festo og modellene vi mottok, enkelte mangler og hvordan dette ble løst. I tillegg blir det beskrevet hvordan modellene og utstyr ble montert. Dette har blitt delt opp i to delkapittel, der første tar for seg modifikasjoner av modulene, og det andre hvordan alt ble montert.

4.3.1 Modifikasjon av moduler

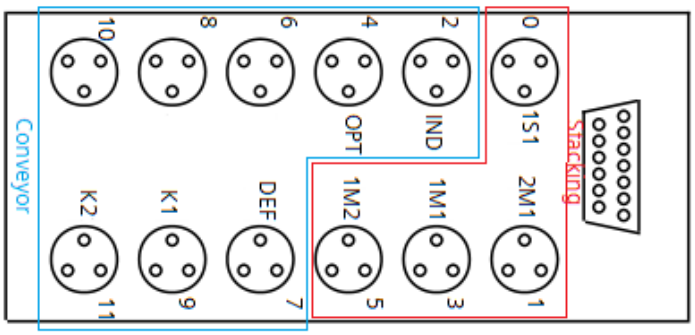
Det viste seg å være flere forskjeller mellom beskrivelsen av modulene i dokumentasjonen fra Festo og hvilke komponenter som var brukt på de fysiske modulene. Uoverensstemmelsen som førte til den største utfordringen var vakuumgriperen, 5A, som modulen kom levert med. Slik denne sto beskrevet ville den gripe på sidene av pucken, mens den faktiske griperen fungerte som en sugekopp mot bunnen av pucken. Når en puck ble plassert på plattformen under vakuumgriperen, nådde ikke griperen bunnen av pucken, og klarte derfor ikke å løfte den. Løsningen på dette problemet ble å montere plattformen under 5A, i startposisjon, omtrent to cm høyere. Dette ble gjennomført ved å bore et hull i den vertikale siden av plattformen og skru den sammen med ventil 5M1.



Figur 12 - Modifikasjon av Handling

På Conveyor-modulen ble det også oppdaget diskrepans mellom dokumentasjon og den fysiske modulen, spesifikt på den optiske sensoren (OPT) som skal gi tilbakemelding i det en puck entrer samlebåndet. Sensoren som var montert på modulen var både av annen konfigurasjon enn dokumentert, samt at den ikke klarte å detektere når et element entret samlebåndet (les mer under 6.2.2 Modifisering av moduler). På grunn av dette ble sensoren flyttet nærmere starten av samlebåndet og vinklet slik at den ga tilbakemelding når en puck dyttes ut på samlebåndet.

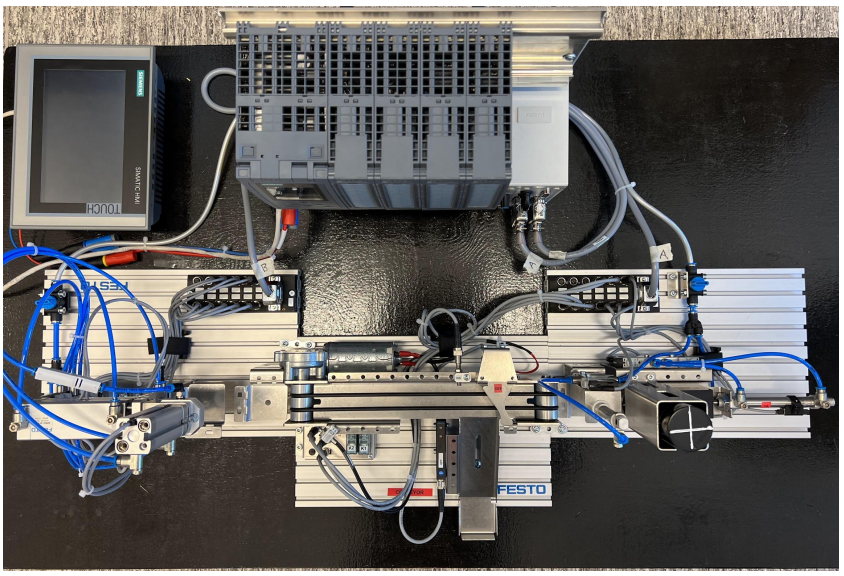
Det viste seg å være problematisk å samkjøre tre moduler ved bruk av PLSen og EduTrainer-systemet som skulle tas i bruk. Hver modul har en tilegnet multi-pin tilkobling der sensorene (innganger) er tilkoblet partall på venstresiden og aktuatorer (utganger) oddetall på høyresiden. Dette var ferdigkoblet ved levering. Festo EduTrainer har derimot bare to porter til dette bruket, som vil si at man kun kan koble til to multi-pin tilkoblinger. Løsningen på dette ble å fjerne multi-pin tilkoblingen fra Conveyor-modulen og heller flytte kablene som var tilkoblet her til tilkoblingen på Stacking-modulen. Det er totalt tolv innganger/utganger, så da er det mulig å få alle komponentene til både Stacking- og Conveyor-modulen på denne tilkoblingen. Dette medførte også at Conveyor-modulen ble mer ryddig og at multi-pin tilkoblingene ble plassert bak modulene. På bildet under er komponenter tilkoblet de nummererte portene og PLSen er tilkoblet Sub D porten til høyre.



Figur 13 - Multi-pin tilkobling for Stacking og Conveyor

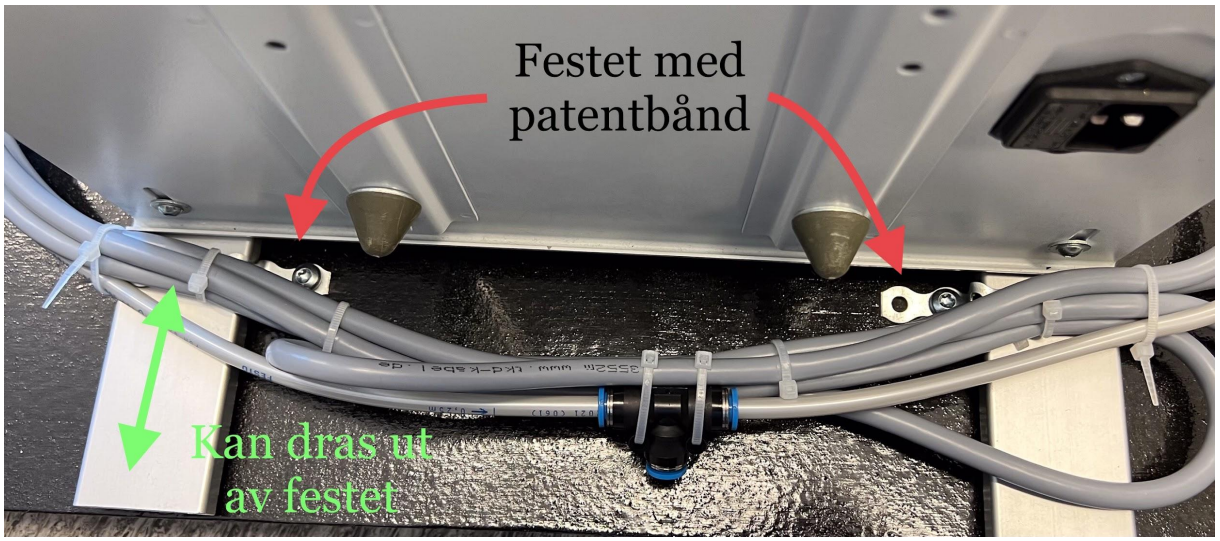
4.3.2 Montering av utstyr

Ettersom modulene skal brukes i undervisningssammenheng ønsket vi at de skulle være montert på en ordentlig måte. Derfor ble en kryssfinerplate tilpasset en tralle på 60x100cm. Platen vil ikke festes i trallen, da vekten av platen, modulene og PLS holder den på plass. Med denne løsningen vil systemet bli mobilt, og det vil være enkelt å løfte platen med modulene fra trallen slik at studentene kan plassere den ved sin pult når de skal arbeide med dem. Kompressoren og eventuelt ekstrautstyr legges i nederste nivå på trallen.



Figur 14 - Utstyr montert på plate

Modulene er festet fra undersiden ettersom det ikke var ønskelig å bore hull i modulene. Conveyor-modulen måtte roteres 180 grader ved montering, for at de tre modulene skulle fungere sammen. PLSen har etter ønske fra veileder blitt festet slik at den lett kan byttes og benyttes i andre prosjekter. Festeløsningen er primitiv, med bruk av patentbånd*, men fungerer.



Figur 15 - Feste av PLS

5. Testing

Kapittelet gjennomgår ulike tester utført i løpet av prosjektet. Deriblant testing av det fysiske utstyret, løsningsforslagprogrammer, emulatorer og til slutt test utført på en 2.års student som hadde faget parallelt med prosjektperioden.

5.1 Test av fysisk utstyr

Det fysiske utstyret måtte monteres og testes før oppgaver med løsningsforslag kunne fremstilles. Vi stolte ikke hundre prosent på at utstyret fungerte likt som spesifisert i dokumentasjonen fra Festo, etter å ha sett flere forskjeller mellom dokumentasjon og Festo-modulene. Kompressoren som ble levert fra Festo var ikke designet for 230V med norsk støpsel, som førte til at modulenes egenskaper måtte testes uten trykkluft i første omgang. Alle innganger og utganger fra moduler til PLSen fungerte som antatt.

Ved testing av modulene med luftkompressor ble det oppdaget flere diskrepanser mellom beskrivelsen av utstyret og det som ble levert. Den optiske sensoren (OPT) klarte ikke å detektere pucker ved starten av samlebåndet når den var montert slik som leverandøren hadde gjort. Det største problemet som ble oppdaget ved testing var at vakuumgriperen som var montert på enden av stempel 4A ikke klarte å gripe puckene, da plattformen disse skulle ligge på var for lav. Ifølge dokumentasjonen fra Festo skulle Handling-modulen ha en annen type griper enn den som ble levert til oss, og med denne griperen hadde ingen problem oppstått. Hvordan disse problemene ble løst utdypes under 4.3 Modifikasjoner og 6.2.2 Modifisering av moduler.

Det ble også oppdaget at kompressoren må supplere minimum 3Bar for at pneumatikken i Stacking- og Handling-modulene skal fungere. Om retningsventilene ikke blir forsynt med tilstrekkelig lufttrykk vil ikke stemplene eller vakuumgriperen fungere. Dette er vesentlig informasjon for at modulene skal kunne opereres og har blitt formidlet i “Instruks til Labingeniør”.

5.2 Test av løsningsforslag-programmer

Løsningsforslagene ble testet underveis i utviklingsprosessen og det ble gjennomgått mange iterasjoner av programmene før løsningsforslagene ble ansett som ferdige.
SCL-programmene ble ferdigutviklet først, og derfor ble disse også testet først. Ettersom labøvelsene øker i vanskelighetsgrad fra labøvelse 1 til og med labøvelse 4, virket det naturlig å begynne med testing av labøvelse 1. Det oppsto til tider problemer med denne programkoden da den i stor grad baserer seg på kommunikasjon mellom PLS og HMI, noe som vi tidlig oppdaget hadde høy risiko for feil. Programmet ble før ferdigstillelse testet flere ganger og lastet opp fra forskjellige datamaskiner, uten de nevnte problemene.

Labøvelse 2 til 4, i SCL, tar alle i bruk Festo-modellene og ved testing av disse løsningsforslagprogrammene ble det stilt høyere krav, da det er bevegelige deler og komponenter som kan bli ødelagt ved feil i programkode. Til tross for høyere krav fungerte disse løsningsforslagprogrammene generelt uten problemer. Labøvelse 2 og 3, som bruker Stacking- og Conveyor-modulene, ble løst med nokså enkel programkode og bød på lite til ingen feil ved testing.

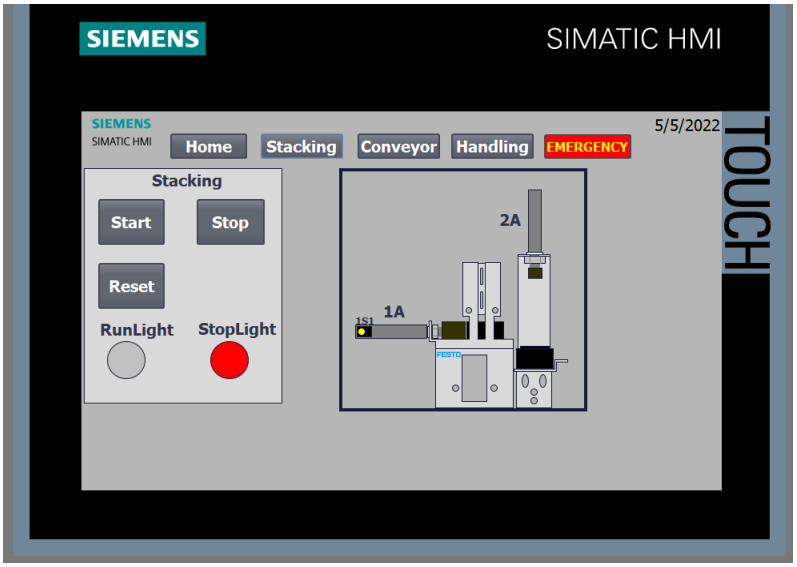
Handling-modulen bydde på større utfordringer ved testing, da løsningsforslagprogrammet til denne modulen ble meget omfattende med flere mulige feil. Som tidligere nevnt i kapittel 4.1.2 Løsningsforslag, var det tidkrevende å utvikle dette programmet. Dette førte da til lengre tid brukt på testing av programkoden og det dannet seg en del problemer rundt case-strukturene og rekkefølgene kodesegmentene skulle utføres. Som følge av dette fungerte heller ikke driftlysene optimalt, og et problem som inntraff var at driftlyset som indikerer at stemplene er i bevegelse ble avslått mens stemplene fremdeles var i bevegelse. Denne feilen oppsto til tider igjen, men har ikke vært konsekvent. Av denne grunn har ikke denne feilen blitt undersøkt nærmere.

Det har også blitt opprettet løsningsforslagprogrammer for de to labøvelsene som skal løses ved GRAPH-programmering. Da GRAPH er et simplere programmeringsspråk, og øvelsene som er designet til studenter som bruker dette språket har lavere vanskelighetsgrad, var ikke testingen av disse programmene spesielt omfattende. Bachelorgruppen hadde ingen erfaring med dette programmeringsspråket og det var krevende å lære seg nye standarder, syntax og hva som er god programmeringsskikk for dette språket. Av denne grunn kan løsningsforslagene fremstå som noe ufullstendige og kunne trolig vært konstruert på en bedre måte. Til tross for dette fungerte programmene som ønsket, det vil si tilsvarende programkoden utviklet i SCL. Enkelte småfeil som ble oppdaget under testing består av driftlys som ikke skifter tilstand til riktig tid. For eksempel vil IND-lyset være påslått frem til deflektoren (DEF) er tilbake i sin originale posisjon. Disse feilene har ikke blitt reparert ettersom de ikke er alvorlige.

5.3 Test av emulator

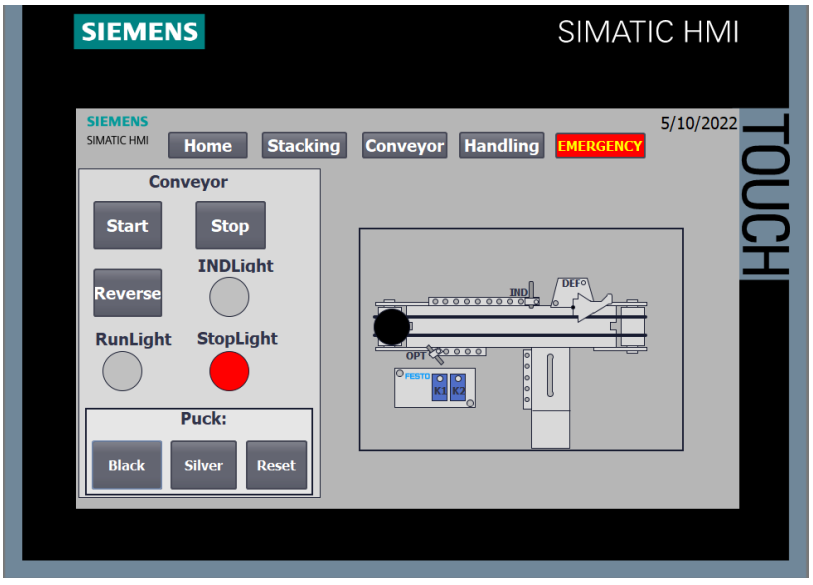
Emulatorprogammene har gjennomgått mange iterasjoner og har blitt testet gjentatte ganger. Disse programmene ble utviklet sent i prosjektet, og var ikke ferdigstilt før siste måneden. Fra starten av utviklingsprosessen til emulatorene antok vi at det kom til å oppstå mange potensielle fallgruver ved gjennomføring og testing av programkode sammen med emulatorene. Av denne grunn har emulatorene blitt testet med opptil tre forskjellige program, løsningsforslaget til SCL, løsningsforslaget til GRAPH, og programmene utviklet av 2.års student (se 5.4 Test gjennomført av 2.års student). Gruppen har lagt inn mye arbeid i å fjerne så mange av disse fallgruvene som mulig, men noen feil vedvarer selv etter testing.

Som nevnt har det blitt produsert tre forskjellige emulator-program, en for hver modul. Ved testing av Stacking-emulatoren ble det ikke funnet noen tydelige feil. Dette var også den simpleste emulatoren å utforme og det var derfor ikke overraskende at testingen av denne gikk smertefritt. Se bilde under av HMI-templat til Stacking-emulatoren.



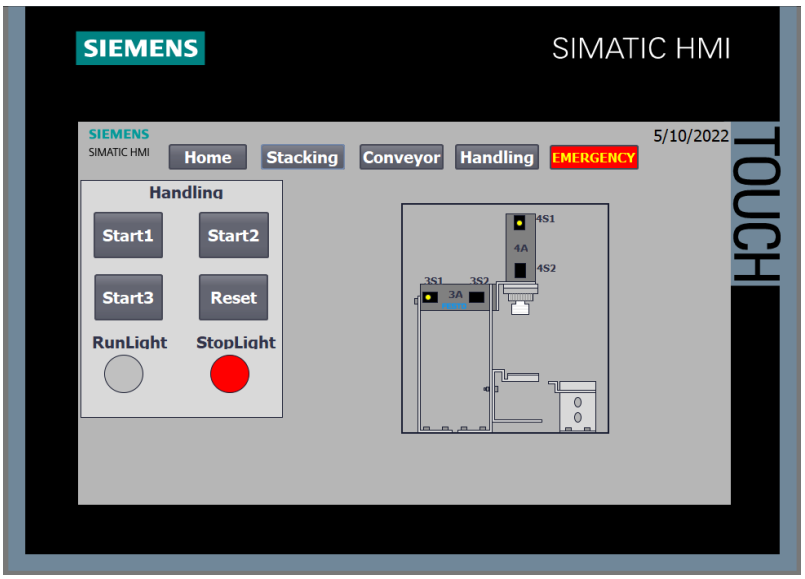
Figur 16 - Emulator animasjon av Stacking

Testing av Conveyor-emulatoren førte til flere revisjoner av programmet, da det oppsto mange feil med horisontal og vertikal bevegelse av puckene. På grunn av dette valgte vi å endre på hvordan emulatoren fungerer i praksis. Med disse endringene implementert er det kun mulig å sende en puck på samlebåndet av gangen og man må bruke reset-knappen før man sender en ny puck, eller vente til en puck har forlatt samlebåndet. Emulatoren fungerer som ønsket, men det er nok fremdeles mulige feil som ikke har blitt funnet selv etter den omfattende testingen som har blitt utført. Se bilde under av HMI-templat til Conveyor-emulatoren.



Figur 17 - Emulator animasjon av Conveyor

Handling-modulen måtte revurderes før og under testing, da det ikke var noen måte å vite hvor en puck vil være i prosessen ut ifra programmet som blir utviklet til å styre modulen. Det er mulig å vite posisjonen til stemplene og hvilke sensorer som er aktivert, men hvordan sekvensene utføres og hvordan pucken skal bevege seg i HMIen, er ikke mulig å vite med den informasjonen som overføres fra modulens styringsprogram til emulatoren. Ut ifra dette ble det tatt en avgjørelse i gruppen om å fjerne pucken fra denne animasjonen, og man må i stedet fokusere på at stemplene beveger seg i riktig rekkefølge som står spesifisert i oppgaven. Det er også lagt til et lys på vakuumgriperen i HMI-templatet, slik at det er mulig å overvåke om vakuum er aktivert. Se bilde under av HMI-templat til Handling-emulatoren.



Figur 18 - Emulator animasjon av Handling

Etter testing av alle emulatorene ble det oppdaget en feil som førte til at emulatorene ikke lenger fungerte med GRAPH-programmene. Denne feilen oppsto sannsynligvis under overføring av programmene. Dette ble ikke oppdaget før mot slutten av prosjektet og ble med det for tidkrevende å rette opp i. Dette problemet må løses i fremtiden om man ønsker å bruke emulatorene sammen med GRAPH.

5.4 Test gjennomført av 2.års student

Midtveis i prosjektet begynte labøvelsene å fremstå som ferdige, og derfor hadde gruppen et ønske om å teste ut labøvelsene på en utenforstående. Veileder kontaktet en daværende 2.års student som hadde faget ELE304, slik at han kunne teste labøvelsene. Fokuset var å sjekke formuleringene i labøvelsene var tydelige og velformulerte, slik at det ikke ville oppstå misforståelser rundt hvordan oppgavene skulle løses, og for å se om vanskelighetsgraden ville være passende.

Studenten gjennomførte labøvelsene og påpekte uklarheter i oppgavetekstene. Det virket også som tiden beregnet per oppgave var passende, da alle labøvelsene ble gjennomført på godt under normert tid. Etter denne testingen ble siste finpuss på labøvelsene foretatt før de ble regnet som ferdigstilte.

Det må tas i betraktning at studenten som testet labøvelsene var faglig dyktig og har mer programmeringsbakgrunn enn den gjennomsnittlige student. Derfor er det mulig at noen studenter vil oppleve labøvelsene som vanskelige. Hvis faget ELE304 gjennomgår noen forbedringer, som nevnt i 3.2 Forbedring av tidligere labøvelser, bør labøvelsene være oppnåelig for studenter som har deltatt i programmeringsfag tidligere.

6. Diskusjon

I dette kapittelet blir arbeidet mot fremdriftsplanen diskutert, i tillegg til problemer som dukket opp underveis og hvordan disse ble løst. Av disse problemene blir hovedsakelig utfordringer rundt HMI-panel, modifiseringen av utstyr og levert utstyr designet for det amerikanske marked beskrevet. Utvidelser og videre arbeid som kan utføres, vil også være dokumentert her.

6.1 Fremdrift

Oppgave	Ansvar	Uke 2	Uke 3	Uke 4	Uke 5	Uke 6	Uke 7	Uke 8	Uke 9	Uke 10	Uke 11	Uke 12	Uke 13	Uke 14	Uke 15	Uke 16	Uke 17	Uke 18	Uke 19	Uke 20	Uke 21	Uke 22 (!)	Uke 23	Uke 24
		10.1 - 16.1	17.1 - 23.1	24.1 - 30.1	31.1 - 6.2	07.02 - 13.2	14.2 - 20.2	21.2 - 27.2	28.2 - 6.3	7.3 - 13.3	14.3 - 20.3	21.3 - 27.3	28.3 - 3.4	4.4 - 10.4	11.4 - 17.4	18.4 - 24.4	25.4 - 1.5	2.5 - 8.5	9.5 - 15.5	16.5 - 22.5	23.5 - 29.5	30.5 - 5.6	6.6 - 12.6	13.6 - 19.6
Forstudie (frist 7. feb)	PR/FH/VVL				Frist (7.2)!																			
Etablering av dokumenter og mapper	VVL														Påske!							Frist 1.6!		
Oppfriskning i PLS	PR/FH/VVL																							
Informasjon og kunnskapsinnhenting	PR/FH/VVL																							
Montere/koble opp Festoer og PLS	PR/FH/VVL																							
Utvikle PLS-program (SCL)	PR/FH/VVL																							
Utvikle PLS-program (GRAPH)	PR/FH/VVL																							
Utvikle oppgaver og deloppgaver	PR/FH/VVL																							
Teste oppgaver på AUTB20	PR/FH/VVL																							
Midtveis presentasjon (21.mars-1.apr)	PR/FH/VVL																Frist!	Frist!						
Utvikle simulator m/grafiske modeller	PR/FH/VVL																							
Utforske mulighet for utvidelse av oppg.	PR/FH/VVL																							
Revisjon av Bachelor	PR/FH/VVL																							
Refleksjonsnotat	PR/FH/VVL																							
Seminar(?)	PR/FH/VVL																							
Muntlig presentasjon	PR/FH/VVL																						Frist!	Frist!
EXPO-forberedelse	PR/FH/VVL																							15.6!
Dokumentasjon (*min. annenhver fredag*)	PR/FH/VVL																							

Forkortelser:
PR = Patrick Rindarøy
FH = Fredrik Hundstad
VVL = Vegard Valen Lindtveit

Figur 19 - Gantt-plan

Gantt-planen viser et oversiktlig bilde av fremdriftsplanen for arbeidsoppgaver og tidsfrister. Som påpekt av veilederne har Gantt-planen til hensikt å virke rettledende. Gruppen arbeidet seg derfor rundt denne planen og fulgte den ikke til punkt og prikke. Fristene er selvfølgelig unntak her og må holdes nøyaktig som de står i planen.

Ettersom ELE350 er et 20-studiepoengsfag, med ⅔ arbeidsmengde i forhold til en normal arbeidsuke på 37,5 time ble det beregnet 25 arbeidstimer hver uke. Med tre personer på gruppen tilsvarer det 75 effektive arbeidstimer per uke. Ved flere uker nådde vi ikke disse 75 timene, ettersom innleveringer i andre fag til tider tok mer tid enn forventet. Mot slutten av prosjektet ble det lagt inn flere arbeidstimer enn det som var tenkt, da det gjensto en del arbeid med ferdigstilling av rapporten.

Ved prosjektstart ble det opprettet en risikoliste der mulige utfordringer og deres risiko, sannsynlighet og tiltak diskuteres (se B.3 Risikoliste). Ettersom prosjektet ble gjennomført våren 2022 var fremdeles Covid-19 pandemien en realitet og alle i gruppen ble smittet av viruset i løpet av prosjektet. Dette førte til en uke, per person, der mindre arbeid ble utført.

Til tross for perioder med gnisninger har det blitt unngått konflikt i bachelorgruppen ved å snakke ut om problemene og ta tiltak for å bedre situasjonen. Det har også vært mild frustrasjon mot arbeidsgiver da vi tidvis ble henvist frem og tilbake mellom ansatte på HVL

uten konkrete svar på spørsmål. Det har som nevnt også vært en del problemer med utstyr, se mer om dette under 6.2 Problemer underveis.
Da prosjektet nærmet seg slutten innså vi at tiden kunne blitt brukt annerledes. Oppstartsfasen innebar mye ineffektivt arbeid. Her burde mer tid vært satt av til oppfriskning i PLS-programmering og arbeidsoppgavene kunne vært bedre fordelt, da alle på gruppen holdt på med samme arbeidsoppgavene parallelt. Det burde også blitt satt av mer tid til å skrive inn i rapporten underveis, ettersom det ble skrevet mye om arbeid som var utført måneder i forveien mot slutten av prosjektet.

6.2 Problemer underveis

Vi støtte på enkelte problemer underveis i prosjektet. Alle problemene var ikke av en slik grad at de stoppet opp hele prosjektet, men det ble brukt tid på feilsøking som kunne vært brukt på andre oppgaver. Hovedproblemene blir gjennomgått under.

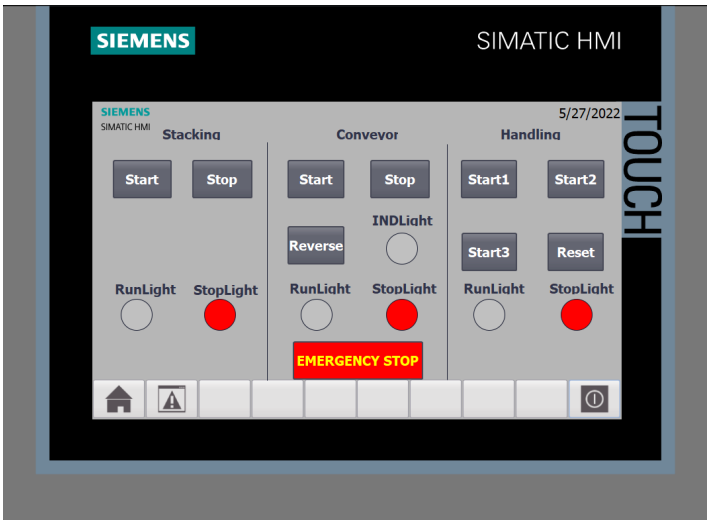
6.2.1 HMI-panel

HMI-panelet skapte de fleste og mest tidkrevende problemene. Gruppen hadde ingen erfaring med HMI før prosjektstart og det var ikke forventet at arbeidet skulle vise seg like utfordrende som det ble. Generelt har idriftsettelse og kommunikasjon til HMI-panelet vært en omfattende del av bachelorprosjektet og et viktig aspekt for å konstruere labøvelsene på pedagogisk vis for fremtidige studenter.

HMI-skjermen som ble brukt i dette prosjektet er av modellen Siemens TP700 Comfort. Det skal i teorien være intuitivt å laste opp templater til denne HMI-skjermen gjennom TIA Portal, men idet en prototyp av det ferdige HMI-templatet skulle lastes inn, oppsto problemer. Kommunikasjonen mellom TIA Portal og HMI-panelet var ufullstendig og dette medførte at det ble umulig å laste opp noe som helst til HMI-panelet. Neste steg i feilsøkingsprosessen ble da å undersøke at alle IP-adresser var korrekt, kommunikasjonsinnstillinger i både TIA Portal og panelet, prøve forskjellige kabler samt mye prøving og feiling. Etter grundig søking på Siemens sine nettforum oppdaget vi flere med samme problem og som hadde en potensiell løsning. Ettersom HMI-panelet var av en eldre modell, med utdatert programvare, var opplasting til panelet fra en nyere versjon av TIA Portal ikke mulig. Det måtte da lastes opp ny programvare, kalt “image”, til panelet slik at det kunne brukes sammen med TIA Portal V16.

Etter vellykket opplasting av et enkelt templat til HMI-panelet, var det ønskelig å opprette forbindelse og mulighet for opplasting til HMI gjennom PLSen. Dette ettersom det tidlig i utviklingsfasen av “Lab 1 - MecLab Forarbeid” var et mål at studentene skulle opprette og laste opp HMI-templat til panelet for å øke læreutbytte. Intensjonen med dette skulle være at studentene koblet seg til PLSen, bruke den som en router, for å videre kunne kommunisere med og laste opp program til HMI-panelet koblet på samme PLS. Ifølge flere tråder på Siemens forum skulle dette være mulig, og mange timer ble brukt på å implementere denne router-funksjonen til PLSen, uten hell. Etter om lag en ukes arbeid med dette problemet, ble

det avgjort at vi måtte fjerne dette aspektet av labøvelse 1. Det medfører at HMI-templatet følger med "Mal - MecLab", slik at all kommunikasjon mellom HMI og PLS vil være lik for enhver student som vil forsøke å løse labøvelsene som har blitt utviklet.

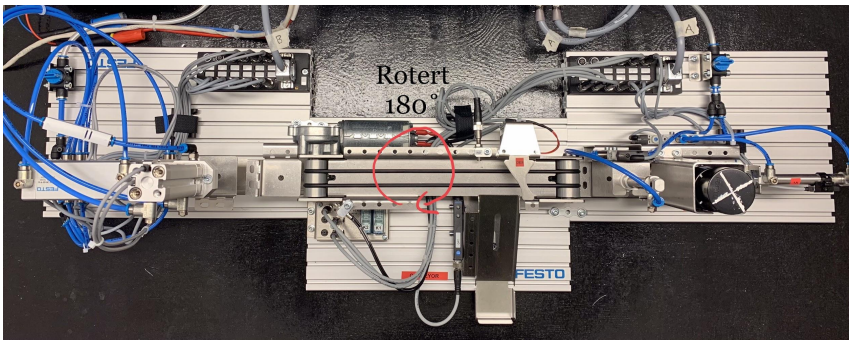


Figur 20 - HMI i TIA Portal

6.2.2 Modifisering av moduler

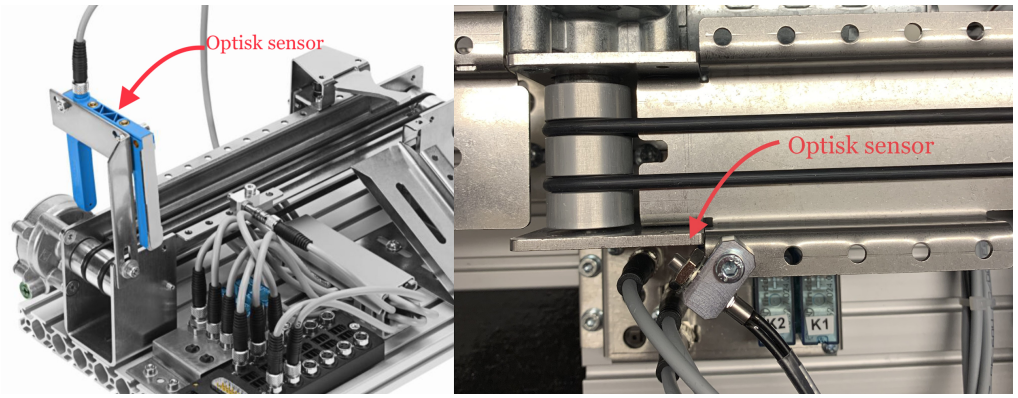
Modulene levert av Festo har som nevnt i kapittel 4.3 Modifikasjoner ikke tilsvart beskrivelsen i medfølgende dokumentasjon. Løsningene på disse problemene er dypere beskrevet i dette delkapittelet. Tidlig ble det oppdaget at modulene ikke stemte overens med illustrasjonene og bildene av modulene som fulgte med MecLab-pakken. De fysiske modulene som var levert til HVL hadde annen utforming av sensorer, aktuatorer og virket ikke å ha vært designet for å brukes sammen slik de kom levert.

For at modulene skal kunne operere sammen, det vil si at pucker kan beveges mellom de tre modulene, måtte Conveyor-modulen snus 180 grader. Dette gjør at modulene ikke vil være på linje når de er montert. Det kunne også vært en mulighet å demontere selve samlebåndet og sensorene på Conveyor-modulen og flyttet de slik at man kunne plassere de tre modulene på et mer estetisk riktig vis, men ble ikke gjennomført da det ikke var ønskelig å flytte på komponentene på modulene. Dokumentasjonen fra Festo spesifiserer flere steder at modulene skal kunne kjøres sammen som levert. Da ble den enkleste løsningen på dette problemet å snu conveyor-modulen. Se bilde under for illustrasjon.



Figur 21 - Meclab med rotert Conveyor-modul

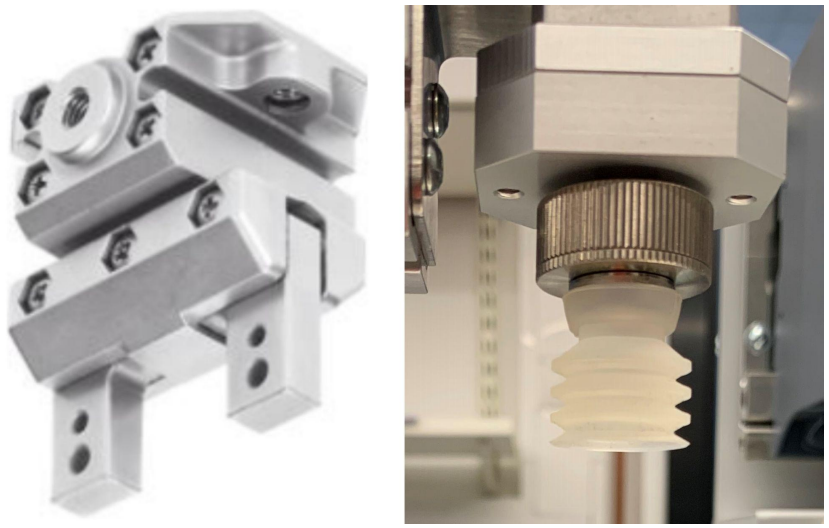
Samlebåndet til Conveyor-modulen kom også levert med en helt annen optisk sensor enn det som var avbildet i dokumentasjonen (se sammenligning av de to sensorene på bildet under).



Figur 22 - Sammenligning av optiske sensorer

Sensoren som var montert på Conveyor-modulen var også plassert slik at den ikke oppdaget om en puck var plassert ved starten av samlebåndet, som er formålet til denne sensoren. OPT-sensoren måtte da flyttes nærmere starten av samlebåndet, samt vinkles slik at en puck vil detekteres ved starten av samlebåndet. Flyttingen av sensoren medfører også at sensoren vil gi høyt signal hvis stempel 3A og 4A på Handling-modulen er i ytre posisjon. Dette resulterte i at om man bruker OPT som startsignal til samlebåndet (slik som er målet i labøvelse 4, deloppgave 2), vil samlebåndet starte uavhengig om det legges en puck på samlebåndet eller om kun det er stemplene som er i denne posisjonen. Om sensoren hadde vært av typen som er avbildet til venstre, hadde ikke dette problemet oppstått.

Vakuumgriperen 5A på Handling-modulen avvek også fra griperen som ble avbildet i "563063_Teaching_With_Meclab". Griperen som var avbildet løftet pucker ved å gripe på sidene av disse, derimot fungerer griperen på den fysiske modulen som en sugekopp på bunnen av puckene. Se bilde under for sammenligning av vakuumgriper avbildet i dokumentasjon (venstre) og den fysiske vakuumgriperen montert på modulen (høyre).

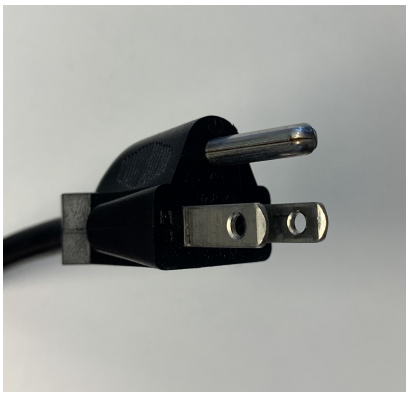


Figur 23 - Sammenligning av gripere

Det viste seg at når bunnen av en puck er plassert ned på plattformen under 5A så klarte den ikke å plukke disse opp. Om man ønsker å samkjøre Handling-modulen med Stacking-modulen er det vesentlig at puckene forflyttes mellom disse modulene med bunnen av pucken ned, ettersom det skal presses på et lokk på denne pucken på Stacking-modulen. Løsningen på dette problemet ble da å montere plattformen under 5A, i startposisjon, omtrent to cm høyere. Dette ble gjennomført ved å bore et hull i den vertikale siden av plattformen og skru den sammen med ventil 5M1. Les mer om dette under 4.3 Modifikasjoner.

6.2.3 Amerikansk utstyr

Det er tydelig at MecLab-pakken som ble levert fra Festo var designet for det amerikanske markedet. Alt av kabler til strømforsyninger, samt kompressoren kom med type B-støpsel, som er beregnet for bruk i Amerika og Japan. Kompressoren opererte som en følge av dette på 110-120VAC 60Hz, som ikke er kompatibelt med spenningen i stikkontakter i Norge (230VAC 50Hz). Så fort dette ble oppdaget kontaktet vi HVL, slik at de kunne finne ut om de ville investere i en ny kompressor, eller gå til anskaffelse av en spenningsomformer slik at den medfølgende kompressoren kunne brukes. Etter flere uker uten et konsist svar fra HVL, tok bachelor-gruppen avgjørelsen om innkjøp av spenningsomformer. Dette var trolig den beste løsningen da kompressoren levert av Festo er designet for å kunne brukes i en klasseromssammenheng, ettersom den er nærmest støyfri.



Figur 24 - Type-B støpsel

6.3 Videre arbeid og utvidelser

Det er her beskrevet noen ideer for videre arbeid og utvidelser av prosjektet. Labøvelsene kan utvides eller endres ved behov, fjernstyring kan legges til og emulatorene kan gjøres mindre finpussinger på.

6.3.1 Labøvelsene

Til tross for at det har blitt lagt mange timers arbeid i å formulere labøvelsene på en oversiktlig og forståelig måte, er det fremdeles mulighet for misforståelser ved gjennomføring av øvelsene. Mange potensielle misforståelser og fallgruver ble luket vekk etter testing av øvelsene på 2.års student, men det vil fremdeles være forbedringspotensiale her.

Utvidelse av labøvelsene er også en mulighet, og noe som oppfordres til etter hvert som læreplanen og faget utvikler seg. Som nevnt i 3.4 Valg av løsning, er labøvelsene utformet for å være praktiske og følger ikke nødvendigvis det som er standard i industrien. Det å inkorporere mer arbeidsrelevante aspekter som logging, sikkerhet og nødstoppkode, vil være fornuftig å tilføye i fremtiden. Teorispørsmålene er også en del av labøvelsene som det er ønskelig at fremtidige forelesere og labingeniører vil endre, slik at de har en tydelig sammenheng med emneplan.

6.3.2 Fjernstyring

Det er mulig å implementere fjernstyring av PLS og HMI som en utvidelse av prosjektet. Det ble studert hvordan disse løsningene kunne blitt implementert, men ble ikke vurdert som en viktig nok del av prosjektet til å sette av tilstrekkelig tid til å realisere en slik løsning. Fjernstyring og IoT* vil absolutt kunne gi fremtidige studenter god læring, og er relevant for arbeidslivet. Skulle dette vært en viktig del av bacheloroppgaven derimot måtte kravene vært omdirigert, ettersom dette er omfattende arbeid. Det er også stort potensiale for sikkerhetsproblemer, med tanke på personvern og risikoen for hacking av systemet. Om fjernstyring vil være hensiktsmessig, rent praktisk, med tanke på at man uansett må ha en person til stede ved de fysiske modulene for å flytte pucker og lignende, er også et spørsmål.

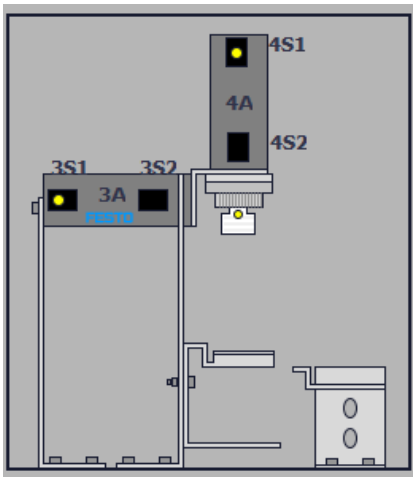
6.3.3 Utbedring av emulatorer

Animasjoner har blitt utformet som fullstendige digitale versjoner av modulene, og kan brukes om modulene ikke er tilgjengelig. Disse har blitt utformet inspirert av tidligere emulatorer i faget. For eksempel vil posisjonssensorer (inngang på PLS) på et gitt stempel styres av retningsventilen (utgang på PLS) til det samme stempelet.

Det er fremdeles rom for å forbedre disse emulatorene, både programkoden og animasjonene. Blant annet er det ikke mulig å kjøre de tre modulene i et samlet system, der man kjører en puck gjennom hele systemet. Dette kan løses ved å opprette et nytt HMI-templat med alle tre modulene på samme skjerm og en variabel for puck som overføres mellom de tre emulatorprogrammene.

For å unngå potensielle feil i Conveyor-emulatoren kan det kun kjøres en puck, enten svart eller sølvfarget, på samlebåndet av gangen. Det hadde vært ønskelig om man kunne kjøre flere pucker samtidig for å teste hvor feilsikker koden til studentene er. Derfor er det ønskelig at man videreutvikler Conveyor-emulatoren i fremtiden slik at dette blir mulig.

Ved animasjonen av Handling-modulen vil det ikke være mulig å plassere en puck, som vil si at for å teste programmet sitt med Handling-emulatoren må man overvåke sekvensen som stemplene beveger seg i. Det vil være vanskelig å utvikle programkode for emulator som vet nøyaktig posisjon til en puck i emulatoren, ettersom det ikke finnes en variabel for puckens posisjon i programmet som styrer modulen (skrevet av studentene). I tillegg vil dette programmet kunne bli utviklet på flere vis, som gjør det enda vanskeligere å animere en puck til rett posisjon til enhver tid i hver enkelt sekvens av prosessen. For å gjøre det enklere å overvåke prosessen uten puck har det blitt lagt til et lys på vakuumgriperen 5A, som indikerer om vakuum er på eller av. I fremtiden bør denne emulatoren forbedres slik at man kan bevege en puck i animasjonen, men dette vil sannsynligvis være omfattende arbeid og åpne for nye fallgruver. Se bilde av Handling-skjermen under, uten puck.



Figur 25 - Handling emulator uten puck og med lys på griper

I tillegg til de ovennevnte utviklingene vil det også være fallgruver i emulatorene, ettersom enhver student vil produsere et unikt program som vil bruke innganger og utganger forskjellig. Dette er viktig å ha i bakhodet når man tester labøvelsene ved hjelp av emulatorene. Det er ikke sikkert at programmet som har blitt utviklet av studentene vil fungere på de fysiske modulene, selv om de fungerer på emulatorene og vice versa. Derfor er det viktig at så mange studenter som mulig får prøve seg på de fysiske modulene, da dette trolig vil være givende og ikke minst oppleves virkelighetsnært og realistisk for studentene.

7. Konklusjon

Det har gjennom prosjektet blitt produsert seks labøvelser med tilhørende løsningsforslag, i sitt respektive programmeringsspråk. Disse labøvelsene tar i bruk tre forskjellige læringsmoduler fra Festo Didactic som skal etterligne prosesser fra industrien, og blir brukt sammen med Siemens PLS og HMI-panel. I tillegg har det blitt utviklet emulatorer i TIA Portal med animasjoner, slik at alle studenter skal ha mulighet til å gjennomføre labøvelsene, selv om de ikke har mulighet til å møte opp på labben.

Det har stått høyt i fokus å overholde kravspesifikasjonene satt av veilederne, men det har i løpet av prosjektet blitt foretatt endringer i kravene og ønskene. Disse endringene har blitt godkjent av veilederne. Blant annet har det kun blitt utformet to labøvelser for programmeringsspråket GRAPH, etter samtale med veiledere, ettersom undervisning i dette språket skal fases ut. Det har blitt vektlagt at labøvelsene skal være enkle å tolke, binde sammen programmeringskunnskap fra andre fag og at det skal være spennende å programmere.

De tre modulene har blitt modifisert og montert slik at det skal være mulig å bruke de i et samlet system. Dette var et aspekt som viste seg å være mer utfordrende enn først antatt, da MecLab-pakken hadde store uoverensstemmelser mellom tilhørende dokumentasjon og fysisk utstyr. Modulene, samt PLS og HMI-panel, har blitt montert på en kryssfinerplate som kan plasseres på 60x100cm traller og lett flyttes på i en undervisningssammenheng. Både PLS og HMI-panelet er enkle å flytte, om det skulle oppstå ønske eller behov for å bytte disse i fremtiden.

Prosjektet har gitt oss viktig og relevant kompetanse om hvordan man arbeider med en langsiktig og omfattende oppgave. Gruppen har anskaffet erfaringer rundt balansering av kravspesifikasjoner mot eventuelle og nødvendige endringer, slik at sluttresultatet blir som tiltenkt. Videre har vi fått verdifull erfaring med det å jobbe effektivt i team, ved alt fra samlet gruppearbeid på teknisk krevende deler, til fordeling av individuelle arbeidsoppgaver. Gjennom prosjektet har gruppen testet mye teoretisk kompetanse fra studiet i praksis, og tilegnet oss kunnskap på områder vi tidligere ikke hadde erfaring med.

Vår mening er at dette prosjektet, og labøvelsene som har blitt utviklet, vil kunne bidra til god læring for fremtidige studenter om det tas i bruk til undervisning. Det ble også gjennomført en undersøkelse av studentenes erfaringer i faget ELE304 PLS programmering våren 2022. Her fremkommer ønsker om mer praktisk utstyr og mer fokus på programmering i øvelsene. Labøvelsene har også blitt testet av en 2.års student, som ga meget positiv tilbakemelding. Av denne grunn ser vi det hensiktsmessig at det som har blitt produsert i dette prosjektet tas med videre i ELE304-faget, og videreutvikles sammen med emneplanen i årene fremover.

Referanser

[1] HVL “Om Høgskulen på Vestlandet” 20.02.2021, [Internett]. Tilgjengelig: https://www.hvl.no/om/ [Funnet 25.01.2022]

[2] HVL “Emneplan for PLS programmering”, [Internett]. Tilgjengelig: https://www.hvl.no/studier/studieprogram/emne/ele304 [Funnet 25.05.2022]

[3] Festo Didactic “About Festo Didactic”, [Internett]. Tilgjengelig: https://www.festo-didactic.com/int-en/company/ [Funnet 26.05.2022]

[4] Festo Didactic “Meclab Mechatronics training system”, [Internett]. Tilgjengelig: https://www.festo.com/us/en/e/technical-education/learning-systems/stem/meclab-r-id_32631/ [Funnet 12.05.2022]

[5] Festo Didactic “Stacking Magazine station”, [Internett]. Tilgjengelig: https://www.festo-didactic.com/int-en/learning-systems/technology-for-schools/meclab/complete-package-and-stations/stacking-magazine-station.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC4xMTE5LjU0MDk [Funnet 16.04.2022]

[6] Festo Didactic “Conveyor station”, [Internett]. Tilgjengelig: https://www.festo-didactic.com/int-en/learning-systems/technology-for-schools/meclab/complete-package-and-stations/conveyor-station.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC4xMTE5LjU0MTA [Funnet 16.04.2022]

[7] Festo Didactic “Handling station”, [Internett]. Tilgjengelig: https://www.festo-didactic.com/int-en/learning-systems/technology-for-schools/meclab/complete-package-and-stations/handling-station.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC4xMTE5LjU0MTE [Funnet 16.04.2022]

[8] Festo Didactic “EduTrainer Universal”, [Internett]. Tilgjengelig: https://www.festo-didactic.com/int-en/learning-systems/training-packages/automation-technology-plc/edutrainer-universal-for-maximum-flexibility.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC4xMjMxLjc2Mjk [Funnet 13.05.2022]

Figurliste

Appendiks A - Forkortelser og ordforklaringer

Ord	**Forklaring**
AI	Analog input - Modul som tillater analoge spenningssignal inn på PLS. Modell som brukes: “AI 8xU/I/RTD/TC ST”
AQ	Analog output - Modul som tillater analoge spenningssignal ut fra PLS. Modell som brukes: “AQ 4xU/I ST”
DI	Digital input - Modul som tillater digitale spenningssignal inn på PLS. Modell som brukes: “DI 32x24VDC HF”
DQ	Digital output - Modul som tillater digitale spenningssignal ut fra på PLS. Modell som brukes: “DQ 32x24VDC/0.5A ST”
Funksjonsblokk	Kodeblokker som lagrer sine inngang- og utgangsverdier i instansdatablokk, slik at verdiene er tilgjengelige selv etter funksjonblokken har blitt utført
GDPR	General Data Protection Regulation - En lov om behandling av personopplysninger
GRAPH	Grafisk programmeringsspråk som brukes for å programmere PLS
HMI	Human-Machine Interface - Operatørpanel, brukergrensesnitt for menneske-maskin-interaksjon
HVL	Høgskulen på Vestlandet
IoT	Internet of Things - Samling av elektroniske enheter som kan dele informasjon med hverandre
Instansdatablokk	Lagrer statisk data lokalt fra funksjonsblokk frem til den kjøres igjen. Henger alltid sammen med en funksjonsblokk
Multi-Pin tilkobling	Interface for tilkobling av alle aktuatorer og sensorer til PLS.
Patentbånd	Et hullbånd i galvanisert metall brukt for å feste ting
PLS	Programmerbar Logisk Styring - Industriell datamaskin brukt til automatisering
SCL	Structured Control Language - Høyklasse programmeringsspråk lignende Pascal
Solenoid	En sylindrisk spole av isolert ledningstråd som brukes som magnet når den fører elektrisk strøm
Sub D	En type elektrisk kontakt som brukes til datakabler
Syslink	En port-type som Festo bruker, samsvarer med IEEE 488 grensesnitt
TIA Portal	Totally Integrated Automation - Programvare til programmering av Siemens PLS

Appendiks B - Prosjektledelse og styring

B.1 Prosjektorganisering

Prosjektleder: Vegard Valen Lindtveit

Ansvarsområder har blitt delegert til gruppemedlemmer for å spre fokuset, men mesteparten av arbeidet har vi samarbeidet på der det har vist seg hensiktsmessig.

Inndeling:

- Vegard Valen Lindtveit: Montering og modifikasjon av moduler, PLS programmering, emulator-kode, emulator-animasjoner, utforming av oppgaver, instruks til labingeniør, rapportskriving
- Fredrik Hundstad: Montering og modifikasjon av moduler, PLS programmering, emulator-kode, utforming av oppgaver, rapportskriving
- Patrick Rindarøy: Montering og modifikasjon av moduler, HMI-design, test av oppgaver, instruks til labingeniør, rapportskriving

B.2 Fremdriftsplan

Vi har produsert timeliste som beskriver hva vi gjorde til hver enkelt dag og hvor mange timer som ble brukt på den spesifikke arbeidsoppgaven.

Fremdriftsplanen vår baserer seg på en Gantt-skjema, hvor det vises hvor lang tid som bør brukes og når forskjellige arbeidsoppgaver skal gjennomføres. Denne har blitt overholdt i stor grad, men har til tider havnet litt bakpå. Se 6.1 Fremdrift og Figur 19 - Gantt-plan.

B.3 Risikoliste

Risiko	Sannsynlighet	Alvorlighet	Konsekvens	Tiltak
Sykdom og/eller skade	Høy	Middels	Hindrer fremdrift i prosjektet. Skjevfordeling av arbeidsmengde.	Hjemmekontor.
Konflikt innad i prosjektgruppen	Lav	Middels	Fører til dårligere samarbeid og samkjøring.	Snakke ut om eventuelle problemer og konflikter.
Konflikt med arbeidsgiver	Lav	Høy	Resultatet blir dårligere enn antatt. Lavere karakter/utbytte.	Jevnlige møter med veiledere og arbeidsgiver for å forhindre konflikter.
Problemer med komponenter/elektronikk i moduler	Middels	Høy	Tid brukes på feilsøking som kunne vært brukt andre steder. Kan fordyre prosjektet ved totalsvikt.	Skrive robust kode for å forhindre feil/skader. Jevnlig testing av moduler.

Appendiks C - Labøvelser

Øvrige bilder og figurer i labøvelsene er hentet fra “Festo dokumentasjon”, tatt selv eller er skjermdump fra egne prosjekt i TIA Portal. Labøvelsene ligger også vedlagt som .docx-fil.

C.1 Lab 1 - MecLab Forarbeid

Fag:	*ELE304 PLS*
Laboratorieøving:	*MecLab Forarbeid*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Lab 1 – MecLab Forarbeid

Gruppenummer:
Studenter:
Godkjent:

Formål:

- Bli kjent med TIA-portal
- Lære å lage et enkelt program i TIA-portal
- Få til kommunikasjon mellom HMI og PLS
- Lære hvorfor vi bruker minneceller

Gjennomføring:

- Laste ned og starte *Mal - MecLab*
- Lage enkle funksjonsblokker (FB) og kjøre disse i Main(OB)
- Styre innganger og utganger med HMI

Utstyr:

- PC med TIA Portal og RJ45-port (evt. Adapter)
- Siemens S7 - 1500 PLS
- Siemens HMI (TP700 Comfort)

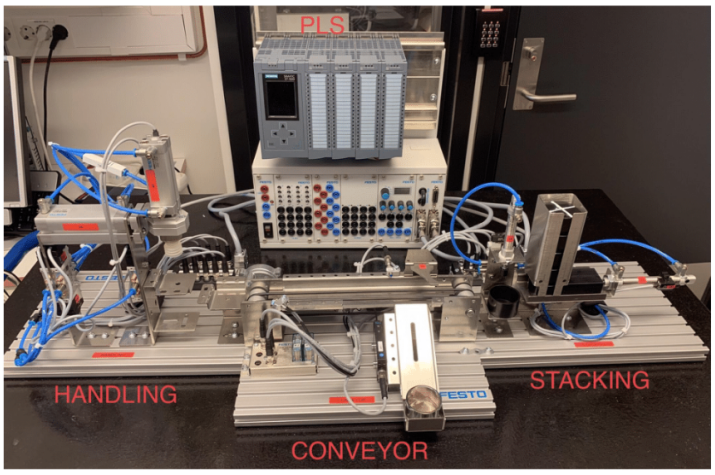
Fag:	*ELE304 PLS*
Laboratorieøving:	*MecLab Forarbeid*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Innføringsinfo:

Dette er en innførende laboppgave hvor dere skal bruke templaten "Mal - MecLab" og bli kjent med TIA-portal. "Mal - MecLab" er et TIA-prosjekt som inneholder en PLS og HMI-skjerm tilsvarende de som skal brukes i praksis. Malen inneholder Tag-liste med alle inngangene og utgangene som trengs for å løse alle labøvelsene, samt en ferdig programmert HMI-skjerm som skal brukes til styring av disse fire labøvelsene. Kommunikasjon mellom HMI og PLS er også konfigurert her, så dere behøver ikke å foreta noen endringer.
Det er veldig viktig å ikke endre på noe av HMI-oppsettet i TIA-prosjektet deres etter dere har lastet ned malen, da den minste endring her vil føre til at dere ikke kan kan kommunisere mellom deres program og koden som er lastet inn på den fysiske HMIen. Om dere skulle være så uheldige å endre på noe i HMI-delen av TIA-prosjektet kan dere laste ned malen på nytt.

Labøvelsene dere skal gjennomføre vil bruke Festo MecLab, som er et sett av tre forskjellige moduler som skal etterligne prosesser fra produksjonsindustrien. Disse tre modulene, kalt Stacking, Conveyor og Handling, skal arbeides med i denne rekkefølgen. Denne første labøvelsen vil legge grunnlaget for videre programmering av disse modulene og kontrollere at kommunikasjonen mellom HMI og PLS fungerer som ønsket. Les gjerne mer om Festo MecLab her.



Fag:	*ELE304 PLS*
Laboratorieøving:	*MecLab Forarbeid*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 1, Kommunikasjonstest mellom HMI og PLS

Før dere begynner på selve oppgaven må dere laste ned «*Mal – MecLab*» og åpne dette prosjektet i TIA-portal. Her ligger som sagt riktig PLS, HMI og alle Tags for innganger og utganger.

Dere skal i denne oppgaven lage et enkelt program for å teste alle knapper og lys i HMI og at kommunikasjonen mot PLS fungerer som den skal. Hver *Start*/*Stop*-knapp i et segment av HMIen skal skru av/på sitt respektive *Run*/*StopLight* og *Reverse* skal aktivere *INDLight*. Handling-segmentet av HMIen har tre start-knapper og en Reset-knapp. Her skal alle startknappene aktivere *RunLights* og *Reset*-knappen skal deaktivere *RunLights* og aktivere *StopLight*.
Lysene skal holde seg påslått selv etter at man har sluppet knappen. Dette kalles en holdefunksjon.

Både *Run*- og *StopLights* kan ikke lyse samtidig i en modul og *StopLights* skal begynne å lyse i det dere starter programmet. Nødstoppen *E_Stop* skal slå av *RunLights* i alle modulene og *INDLight*, i tillegg skal alle *StopLights* slås på.

Tips: Det er lurt å teste koden underveis slik at det blir mindre kode å feilsøke hvis det oppstår problemer. Dette gjelder som regel alltid når man programmerer PLS.

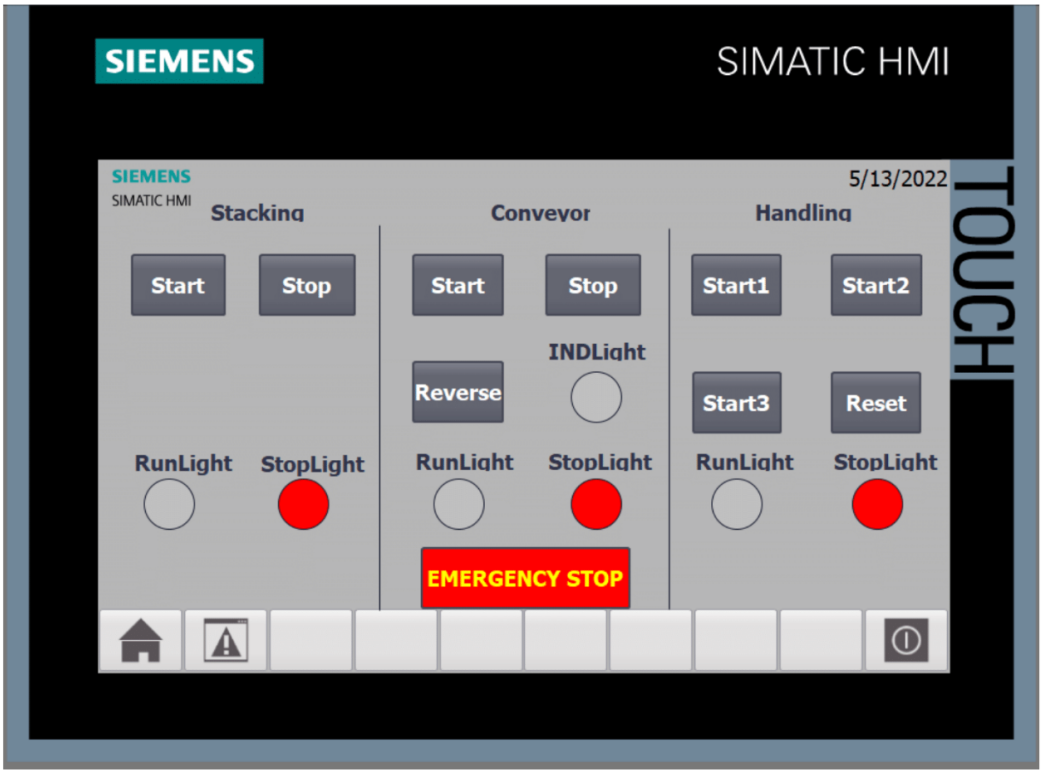
1. Opprett en funksjonsblokk (FB) med navnet *fbHMITest* i språket SCL.
2. Begynn å skrive kode for Stacking-segmentet av HMIen, her bør man bruke If/Else-statements.
3. Skriv deretter kode for knappene og lysene i de to andre modulene.
4. Etter dere har skrevet ferdig koden oppretter dere en instansdatablokk (IDB) med navnet *idbHMITest* av typen som tilsvarer *fbHMITest*.
5. Initier *idbHMITest* i Main[OB1]. OBS! Sjekk at Main er i riktig programmeringsspråk!
6. Test programmet mot HMI.
   PS. Dere kan teste både med fysisk PLS og HMI eller simulere disse i TIA Portal.

Test at programmet deres fungerer.

Fag:	*ELE304 PLS*
Laboratorieøving:	*MecLab Forarbeid*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Se bilde av HMI-skjermen under:



Fag:	*ELE304 PLS*
Laboratorieøving:	*MecLab Forarbeid*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 2, Skriving og testing av programkode

Dere skal her lage et enkelt program for å styre lysene i Stacking-segmentet av HMIen (venstre segment av HMI). Dere kan tenke at dette programmet kan brukes til å styre en motor AV/PÅ. Når dere holder inne startknappen i 1 sekund skal motoren starte og *RunLight* slås på. Hvis dere trykker på *Stop*-knappen skal motoren stoppe og *StopLight* slås på.

Når man skal skrive programkode der man bruker inngangs- og utgangssignaler skal man ikke programmere direkte med disse, men heller opprette variabler (minneceller i PLSen) og bruke disse i programkode og sette disse til riktig inngang/utgang på rett sted i programmet. Dette betyr at dere må opprette statiske/temporære variabler for alle innganger og utganger fra HMIen i funksjonsblokken deres. Inputs (dvs. knapper) initieres i starten av FBen og outputs (dvs. lys) skriver man til på slutten av koden.

Til tidsfunksjonen på startknappen kan det være en god ide å bruke en TON-blokk som dere finner i *Timer operations* under *Basic Instructions*. TON (Time-On Delay) brukes til å «forsinke» et signal ut med en oppgitt tid etter signal blir mottatt på inngangssiden. På den måten kan man utføre funksjonen der man må holde inne startknappen i 1 sekund før utsignalet blir høyt. Man behøver kun å initiere blokken ett sted i koden for å kunne bruke den på forskjellige steder og man behøver kun å oppgi IN og PT (ønsket forsinkelsestid) til deres bruk. Slik ser TON-blokken ut i TIA Portal i SCL:

```
"idbTON".TON(IN:=_bool_in_,
             PT:=_time_in_,
             Q=>_bool_out_,
             ET=>_time_out_);
```

1. Opprett en FB med navnet *fbLogicTimer* i programmeringsspråket SCL.
2. Opprett statiske/temporære variabler for de to knappene og de to lysene. Husk gode variabelnavn!
3. Dere skal nå koble deres statiske/temporære variabler for knappene og lysene mot tilhørende HMI-tags.
   Hint:
   //Initiate temporaries/statics:
   «*Start* statisk/temp» := «*Stacking_Start* tag»
   «*Stop* statisk/temp» := «*Stacking_Stop* tag»
   //Write to outputs
   «*Stacking_RunLight* tag» := «*RunLight* statisk/temp»
   «*Stacking_StopLight* tag» := «*StopLight* statisk/temp»
4. Skriv kode slik at *RunLight* blir *true* (og holder verdien) når dere har holdt inne *Start* i 1 sekund.
5. Når dere trykker på *Stop* skal *RunLight* slås av og *StopLight* slås på og forbli påskrudd selv etter dere har sluppet knappen. *StopLight* skal alltid være aktivert når *RunLight* er deaktivert!
6. Opprett en IDB av *fbLogicTimer* og initier den i Main[OB1] .
7. Test at programmet fungerer som det skal, enten med fysisk PLS og HMI, eller simulert i TIA Portal.

Fag:	*ELE304 PLS*
Laboratorieøving:	*MecLab Forarbeid*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 3, Teorispørsmål

1. Hvorfor skal man alltid bruke interne variabler når man skriver kode i TIA-portal.
2. Hva er hensikten med TAG-table i PLS og HMI?
3. Hva er forskjellen på TON- og TOF-delay?

Svar:

Fag:	*ELE304 PLS*
Laboratorieøving:	*MecLab Forarbeid*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Levering av labøvelsen:

Opprett en mappe der dere legger besvarelse på teorispørsmålene og en zippet mappe med TIA-prosjektet deres.

C.2 Lab 2 - Stacking Modul

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Lab 2 – Stacking Modul

Gruppenummer:
Studenter:
Godkjent:

Formål:

- Lære å arbeide med fysisk utstyr
- Bruke inngang- og utgangssignaler fra fysisk PLS i programmering
- Bruke Case-strukturer til å utføre sekvensiell kode

Gjennomføring:

- Opprette og programmere nødvendige blokker, og initiere disse i Main
- Teste med fysisk modul, PLS og HMI

Utstyr:

- PC med TIA Portal og RJ45-port (evt. Adapter)
- Siemens S7 - 1500 PLS
- Siemens HMI (TP700 Comfort)
- Festo Didactic Stacking Module

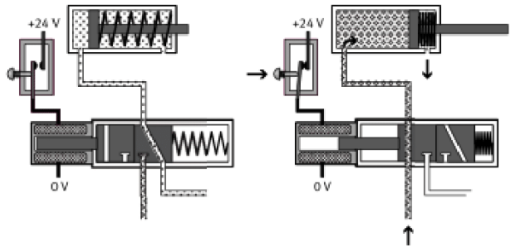
Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Innføringsinfo:

I denne labøvelsen skal dere skrive kode for å styre prosessen til Stacking-modulen fra MecLab. Denne modulen har som virkemåte å presse to pucker sammen, en prosess som utføres ved hjelp av to pneumatiske (luftstyrte) stempel. Disse stemplene styres av solenoidstyrte pneumatiske retningsventiler, som styrer lufttrykket til forskjellig side av stempelet ved hjelp av et elektrisk signal fra PLSen. Disse må ha lufttrykk inn på luftinngangen og signalet inn på solenoidene styrer hvilken retning lufttrykket flyttes. Slike solenoidstyrte ventiler kan ha mange forskjellige konfigurasjoner, men de dere skal bruke er relativt simple.

Se bilde under som viser hvordan *2M1* styrer *2A*.



Hentet fra Festo "563063_Teaching_With_MecLab"

Til å styre stempel *1A* bruker dere en solenoidventil med to elektriske innganger, der signal inn på *1M1* vil dytte stempelet til ytre posisjon og *1M2* vil trekke stempelet til indre posisjon. Hvis både *1M1* og *1M2* mottar høyt signal samtidig vil ikke stempelet bevege seg, men beholde nåværende posisjon. Stempel *2A* fungerer litt annerledes da det er fjærbelastet og vil automatisk trekke seg tilbake til indre posisjon når det ikke kommer signal inn på *2M1*.

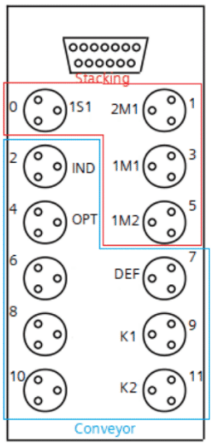
I koden dere skal utvikle i denne labøvelsen skal *1A* dytte en toppuck (med hvitt kryss) oppå en bunnpuck som ligger på en plattform. Puckene skal klemmes sammen ved hjelp av stempel *2A*. Stempel *1A* har også en sensor, *1S1*, som gir digital tilbakemelding (høyt signal) når stempelet er i bakre posisjon. Denne sensoren vil være smart å bruke for å sjekke at stempelet er i rett posisjon til å utføre en sekvens.

Fag: *ELE304 PLS*
Laboratorieøving: *Lab 2 – Stacking Modul*
Laboratorielærer: *[Navn her]*

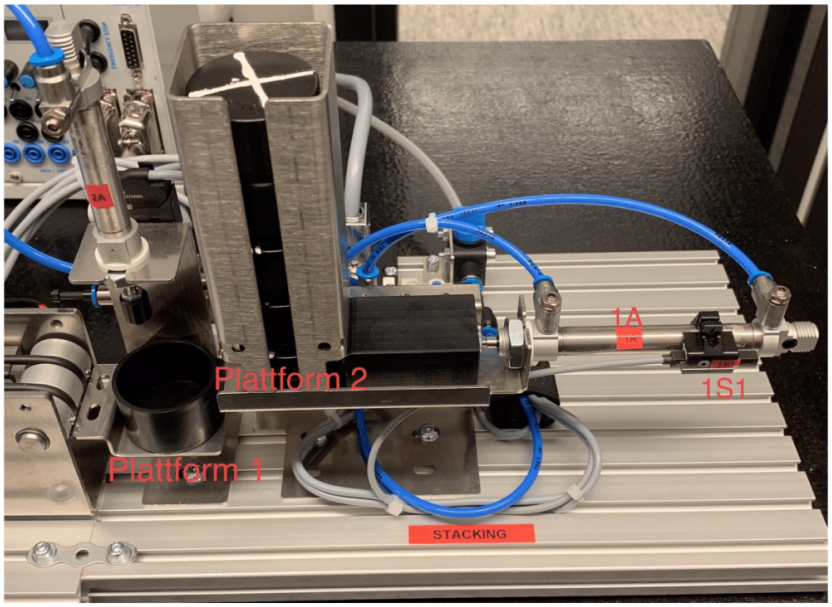
Høgskulen på Vestlandet

Innganger og utganger til Stacking-modulen (ta i bruk "Tag" i koden):

PLS-Inputs	Multipin-Inputs	Komponent	Tag
I0.0	0	Sensor 1A, indre posisjon	1S1



PLS-Outputs	Multipin-Outputs	Komponent	Tag
Q0.0	1	Solenoid 2A, ytre posisjon	2M1
Q0.1	3	Solenoid 1A, ytre posisjon	1M1
Q0.2	5	Solenoid 1A, indre posisjon	1M2



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 1, fbStackingModule

Fortsett i prosjektet fra Lab 1 – MecLab Forarbeid, slik at dere fremdeles har riktig PLS og HMI. Dere skal i denne oppgaven ta i bruk Stacking-segmentet av HMIen. Startposisjon til stemplene vil være når *1A* og *2A* er i bakre posisjon. Da må ventilene ha *1M1=false*, *1M2=true* og *2M1=false*. Ved denne posisjonen vil også *1S1* være *true*.

Prosessen som skal utføres, skal skje sekvensielt. Når man skal utvikle slik kode kan det være vanskelig å bruke *If-Else*-strukturer, da er Case-strukturer et meget godt alternativ. Dette er en måte å kjøre koden i en ønsket rekkefølge avhengig av en INT variabel som bestemmer hvilken case som skal kjøre ved en gitt INT-verdi(0, 1, 2, 3….).

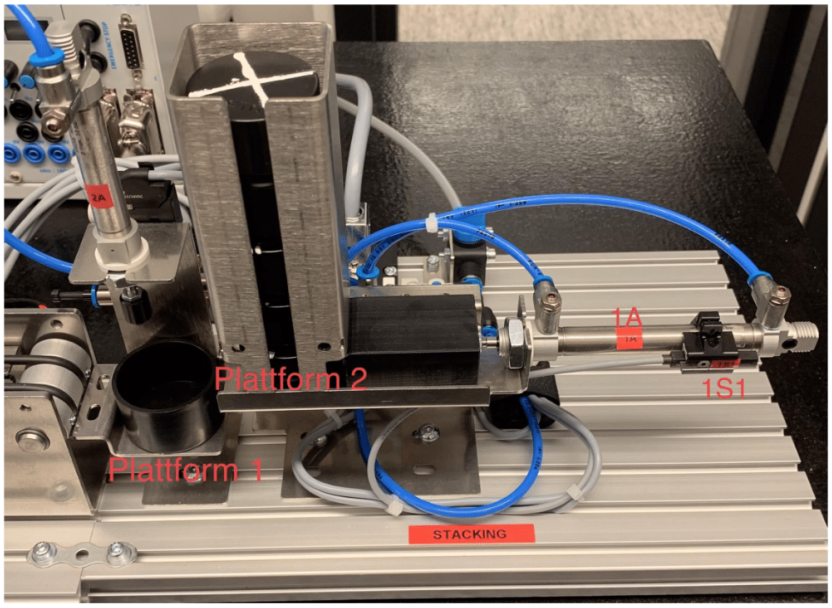
1. Opprett en funksjonsblokk *fbStackingModule* og tilhørende statiske/temporære variabler for innganger/utganger fra HMI og PLS.
2. Skriv kode slik at når man trykker på startknappen (behøver ikke å holde den inne), og sensor *1S1* er *true,* vil følgende sekvens utføres:
   Stempel *1A* skyves ut (dvs. *1M1* blir *true*) -> etter 2s trekkes stempel *1A* tilbake (*1M2* blir *true*) -> stempel *2A* skyves ut (*2M1* blir *true*) -> etter 2 sekund trekkes stempel *2A* tilbake (*2M1* blir *false*). Her vil da toppucken dyttes på bunnpucken for så å presses sammen. Sekvensen skal kun kjøres en gang når startknappen trykkes på.
   Tips: Det kan være smart å bruke sensor *1S1* til å sjekke at stempel *1A* er i ønsket posisjon og bruke TON/TOF-blokker til tidsfunksjonene.
3. *Stop* skal stanse sekvensen til enhver tid og setter stemplene til startposisjon.
4. *RunLight* skal lyse så lenge sekvensen kjører og *StopLight* skal alltid lyse når sekvensen ikke kjører.
5. Opprett datablokk og initier i Main.

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Testing av programkode med fysiske moduler:

Når dere skal teste modulen legger dere en bunnpuck på plattform 1, med bunnen ned, og en toppuck legges på plattform 2 med krysset opp (Se bildet under). Bruk HMI-panelet til å starte prosessen og test om sekvensen utføres som spesifisert i oppgaven.



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

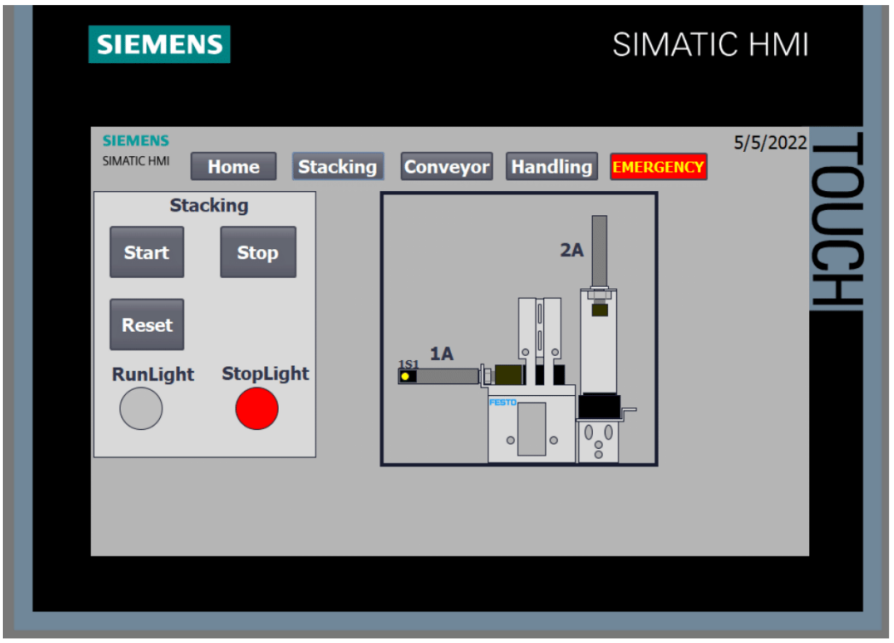
Høgskulen på Vestlandet

Testing av programkode med emulatorprogram i TIA-portal:

Det vil også være mulig å teste programkoden deres med en emulator om dere ikke har tilgang på de fysiske Meclab-modulene. Denne emulatoren utfører koden deres på tilnærmet samme måte som en fysisk modul. Om dere skal bruke emulatoren må dere laste over FBene deres over til "Mal - MecLab (Emulator)" og gjenopprette eventuelle TON/TOF-blokker om dere har brukt "Single Instance"-blokker.

I startposisjon vil toppucken være plassert som vist på bildet under og etter vellykket gjennomføring av koden vil puckene ha blitt presset sammen. For å flytte toppucken tilbake til startposisjon igjen trykker dere på "Reset"-knappen.

Slik ser emulator-HMIen ut:



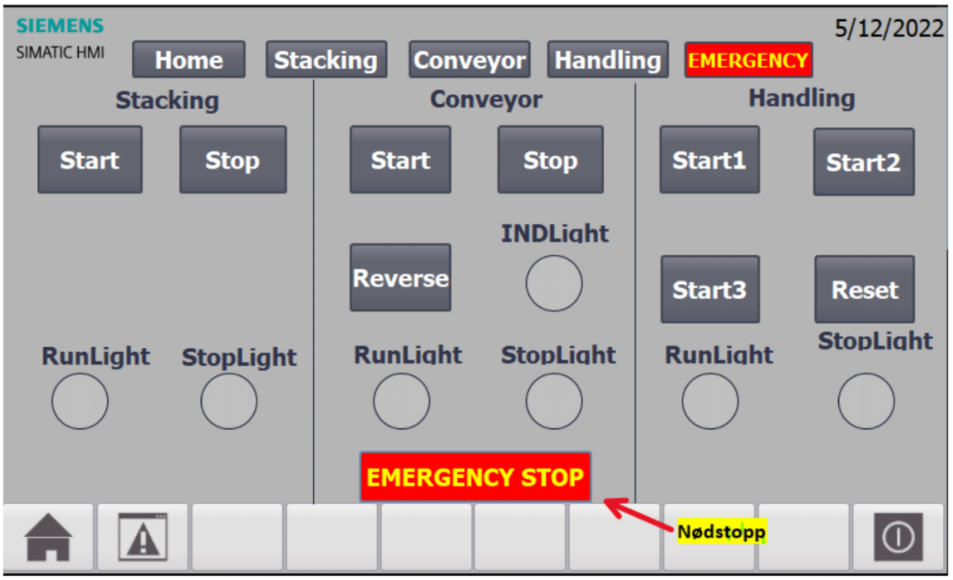
Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 2, Nødstopp

Nødstoppknappen er en vesentlig del av de fleste prosesser og ofte er nødstoppkoden meget omfattende. Det finnes utrolig mange forskjellige standarder i arbeidslivet på dette og det vil ikke gås noe dypere inn i forskjellige løsninger. I labøvelsene fremover behøver dere ikke å skrive spesielt omfattende nødstoppkode, men vi vil gjerne at dere tenker gjennom hva som skal skje når dere aktiverer nødstoppen. Skal hele prosessen stoppe? Bør man plassere nødstoppkoden i Main eller i FB? Skal man stoppe signalet til alle ventiler eller flytte stemplene til en spesiell posisjon? Her det mange løsninger. Ha dette i bakhodet når dere programmerer nødstoppfunksjon på de gjenstående labøvelsene.

1. Reflekter rundt hva som kan være en god løsning for hva nødstoppen skal gjøre i denne modulen.
2. Implementer den løsningen dere kom frem til i programkoden.
3. Test om nødstoppkoden virker som ønsket.



Nødstoppen er inkorporert i HMIen. Det er ingen fysisk nødstoppknapp.

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 3, Teorispørsmål

1. Hva er fordeler og ulemper med å bruke Case-struktur over If/Else-statements?
2. Hvordan fungerer solenoid-ventiler?
3. Hva skjer hvis du gir høyt signal til begge sider av en retningsventil (1M1 og 1M2)?

Svar:

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Levering av labøvelsen:

Opprett en mappe der dere legger besvarelse på teorispørsmålene og en zippet mappe med TIA-prosjektet deres.

C.3 Lab 3 - Conveyor Module

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 3 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Lab 3 – Conveyor Modul

Gruppenummer:
Studenter:
Godkjent:

Formål:

- Lære å arbeide med fysisk utstyr
- Bruke inngang- og utgangssignaler fra fysisk PLS i programmering
- Styre motor vha. releer

Gjennomføring:

- Lage og kode nødvendige blokker, og initialisere disse i main.
- Teste med fysisk modul, PLS og HMI.

Utstyr:

- PC med TIA Portal og RJ45-port (evt. Adapter)
- Siemens S7 - 1500 PLS
- Siemens HMI (TP700 Comfort)
- Festo Didactic Conveyor Module
- Festo Didactic Stacking Module

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 3 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

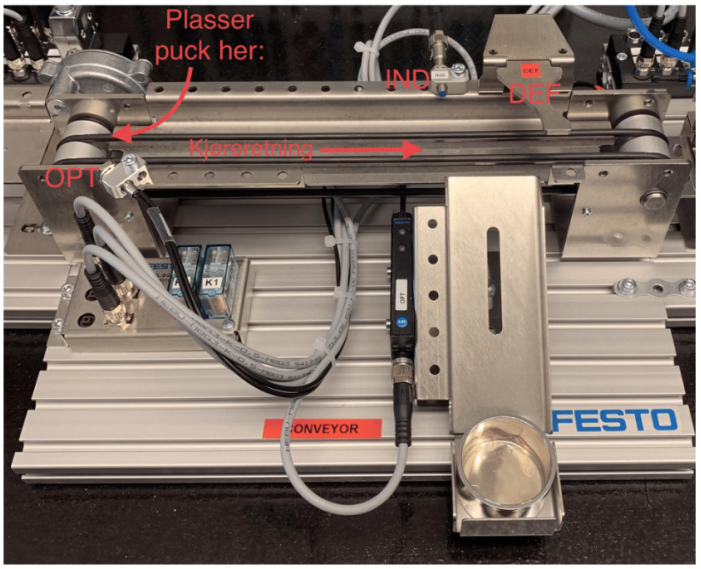
Høgskulen på Vestlandet

Innføringsinfo:

I denne laboppgaven skal dere kjøre pucker på et samlebånd og sortere svarte og sølvfargede pucker fra hverandre. Modulen har to inngangssignaler til PLSen, en optisk sensor (*OPT*) som gir tilbakemelding når et objekt oppdages ved starten av samlebåndet, og en induktiv sensor (*IND*) midt på samlebåndet som gir tilbakemelding ved deteksjon av en sølvfarget puck siden disse har et metallisk belegg. Det er også tre separate element på utgangssiden; en solenoidstyrt deflektor (*DEF*) som vil dytte pucker av samlebåndet, et rele til å starte motoren og et rele til å reversere motorretningen.

Når man styrer en motor, i dette tilfellet til samlebåndet, med en PLS må man bruke releer. Man bruker disse ettersom strømmen PLSen kan supplere fra sine utganger ofte er for lav til å drive en motor. Releet fungerer som en bryter der man bruker et lite signal (fra PLS) til å aktivere en solenoid, som igjen kobler sammen inngangen og utgangen på releet. På denne måten styres en større strøm som driver selve motoren. *Conveyor*-modulen har to releer, *K1* som styrer motoren av/på og *K2* som vil kjøre motoren i revers (*K2* vil kun kjøre samlebåndet i revers hvis *K1* også er aktiv).

Samlebåndet kjører de svarte klossene videre til stacking-modulen. I oppgave 2 skal dere binde sammen disse to modulene og teste at koden fra forrige labøvelse samkjører med koden dere vil skrive i denne labøvelsen.



Bilde av Conveyor-modulen

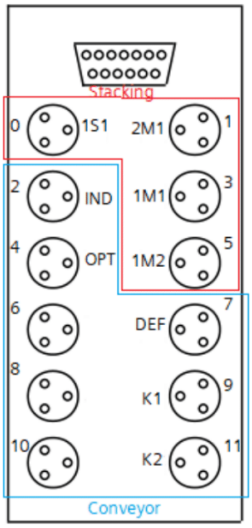
Fag: *ELE304 PLS*
Laboratorieøving: *Lab 3 – Conveyor Modul*
Laboratorielærer: *[Navn her]*

Høgskulen på Vestlandet

Innganger og utganger til Conveyor-modulen (ta i bruk "Tag" i koden):

PLS-Inputs	Multipin-Inputs	Komponent	Tag
I0.1	2	Induktiv sensor	IND
I0.2	4	Optisk sensor	OPT

PLS-Outputs	Multipin-Outputs	Komponent	Tag
Q0.3	7	Deflektor	DEF
Q0.4	9	MotorStart	K1
Q0.5	11	Endre Motorretning	K2



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 3 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

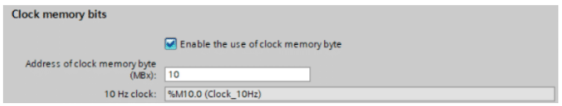
Oppgave 1, fbConveyorModule

Fortsett i prosjektet fra Lab 2 – Stacking Modul. Dere skal i denne oppgaven ta i bruk «Conveyor»-segmentet av HMIen.

Det skal legges pucker av to forskjellige materialer på båndet, der de sølvfargede kan oppdages av en induktiv sensor (*IND*) og de svarte ikke kan. Slik skal man sortere vekk de sølvfargede puckene og la de svarte puckene passere forbi en deflektor. Det skal i tillegg lages ekstra kode for å kjøre samlebåndet i revers, selv om dette ikke har noen reell hensikt i denne modulen.

1. Opprett en funksjonsblokk *fbConveyorModule* og tilhørende statiske/temporære variabler for innganger/utganger fra HMI og PLS.
2. Skriv kode slik at samlebåndet begynner å kjøre (*K1* blir *true*) når man trykker på startknappen og sensor *OPT* er *true*. Hvis *OPT* ikke er *true* skal ikke samlebåndet starte.
3. RunLight skal være aktivert når *K1* er *true* og *StopLight* skal alltid være aktivert når *K1* er *false*.
4. Nødstoppen og stoppknappen skal stanse samlebåndet, slå av *RunLight* og skru på *StopLight*.
5. De sølvfargede puckene skal sorteres vekk ved bruk av sensoren *IND* og deflektoren *DEF,* de svarte puckene skal kjøres videre til stacking-modulen. Deflektoren skal holdes nede i 1 sekund etter at *IND* har blitt aktivert.
6. Når sensoren *IND* har høyt signal skal *INDLight* være aktivert.
7. Når man holder inne *Reverse*-knappen skal motoren kjøre i revers (*K2* blir *true*). Når samlebåndet kjøres i revers, skal også *RunLight* blinke 1 gang i sekundet (1Hz). *K2* kan kun kjøres samtidig som *K1* for ønsket effekt.
   OBS! Det kan hende at blinkingen ender med at *RunLight* er av, i så fall må lyset skrus på igjen automatisk når dere trykker på neste startknapp.
8. Opprett datablokk og initier i Main.

Tips:
For å få til blinkefunksjonen på *RunLight* er det en god ide å bruke *Clock Memory Bits* i PLSen. Disse er *MemoryBits* som skrus av/på med intervaller som samsvarer med frekvensen som står oppgitt i navnet på Clock Bit'en (Eks. *Clock_10Hz* skrus true/false 10 ganger per sekund).



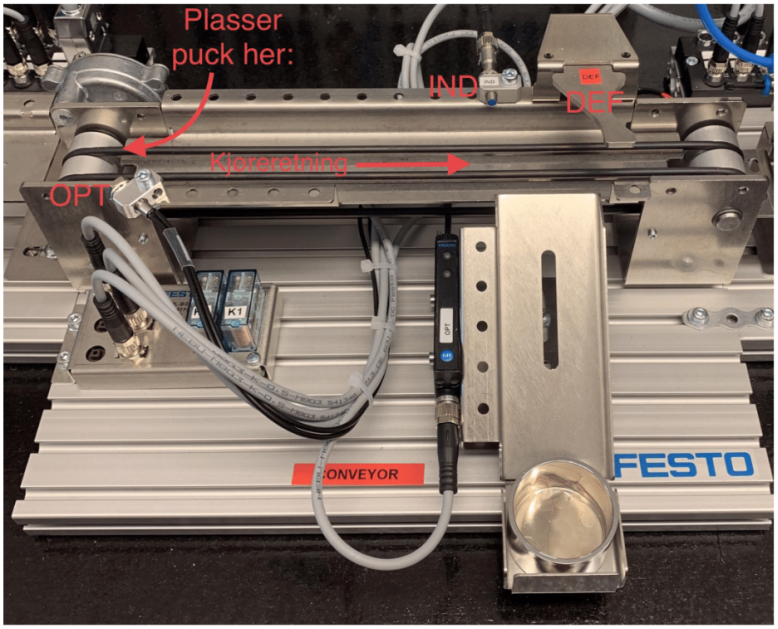
Clock memory bits må aktiveres ved å trykke på selve PLSen under *Device configuration,* og dere må sette *Address of clock memory byte* til eksempelvis 10, slik at ikke klokkebitene allokeres til en minnebit som allerede brukes. Når dette er gjort blir klokkebitene automatisk lagt til i *Tag table* (husk å kompilere PLSen) og kan tas i bruk direkte med navn i koden.

Fag: *ELE304 PLS*
Laboratorieøving: *Lab 3 – Conveyor Modul*
Laboratorielærer: *[Navn her]*

Høgskulen på Vestlandet

Testing av programkode med fysiske moduler:

Når dere skal teste modulen legger dere en puck ved starten av samlebåndet (som vist på bildet under), slik at den blir observert av OPT. Husk å teste programmet med både svarte og sølvfargede pucker, og sjekk at alle lys og knapper fungerer som de skal.



Fag: *ELE304 PLS*
Laboratorieøving: *Lab 3 – Conveyor Modul*
Laboratorielærer: *[Navn her]*

Høgskulen på Vestlandet

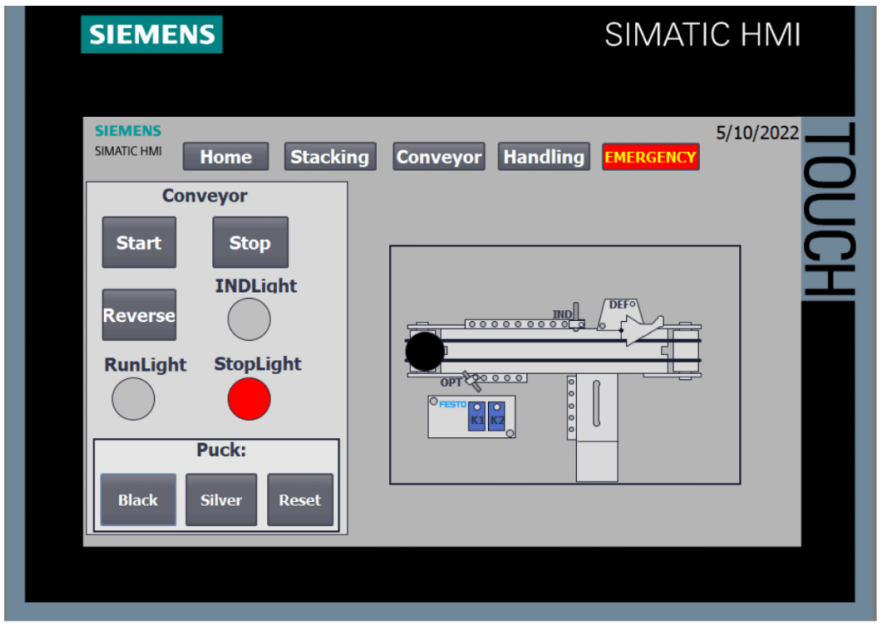
Testing av programkode med emulatorprogram i TIA-portal:

Det vil også være mulig å teste programkoden deres med en emulator om dere ikke har tilgang på de fysiske Meclab-modulene. Denne emulatoren utfører koden deres på tilnærmet samme måte som i virkeligheten. Om dere skal bruke emulatoren må dere laste over FBene deres over til "Mal - MecLab (Emulator)" og gjenopprette eventuelle TON/TOF-blokker om dere har brukt "Single Instance"-blokker.

Det er lagt til knapper på emulator-HMIen for å bestemme hvilken type puck som blir "lagt på samlebåndet". Man må trykke på "Silver" eller "Black" for at en animert puck av den ønskede fargen/typen legges på båndet og OPT blir true. Ved fungerende programkode vil de sølvfargede puckene sorteres ut på rampen, og de svarte puckene vil kjøres ut av samlebåndet og "forsvinne". Trykk på "Reset"-knappen for å fjerne pucker i fra samlebåndet/rampen. Det er kun mulig å simulere èn puck av gangen med emulatoren. IND vil kun settes til true når en animert sølvfarget puck passerer posisjonen til IND-sensoren, og OPT er kun true når en puck er posisjonert i begynnelsen av samlebåndet.

Koden deres skal fungere på samme måte med denne emulatoren som med den fysiske Conveyor-modulen.

Slik ser emulator-HMIen ut:



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 3 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 2, Samkjøre med *Stacking Modul*

Dere skal i denne oppgaven prøve å kjøre både *Stacking*-modulen og *Conveyor*-modulen sammen.
Dette gjør dere enkelt ved å initiere instansdatablokkene for begge programmene i Main. Dere behøver ikke å endre på noe av koden i FBene.

Det er dessverre ikke mulig å utføre denne oppgaven ved bruk av emulator. Om ønskelig kan dere teste begge programmene samtidig med emulatoren, men er ikke mulig å kjøre en puck fra en modul til en annen.

Fag: *ELE304 PLS*
Laboratorieøving: *Lab 3 – Conveyor Modul*
Laboratorielærer: *[Navn her]*

Høgskulen på Vestlandet

Oppgave 3, Teorispørsmål

1. Hvorfor bruker man rele til å styre motorer?
2. Hvordan fungerer en optisk sensor?
3. Hvordan fungerer en induktiv sensor?

Svar:

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 3 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Levering av labøvelsen:

Opprett en mappe der dere legger besvarelse på teorispørsmålene og en zippet mappe med TIA-prosjektet ditt.

C.4 Lab 4 - Handling Modul

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 4 – Handling Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Lab 4 – Handling Modul

Gruppenummer:
Studenter:
Godkjent:

Formål:

- Lære å arbeide med fysisk utstyr
- Lære å utføre mer avansert sekvensiell kode

Gjennomføring:

- Lage og kode nødvendige blokker, og initiere disse i Main.
- Skrive kode med flere Case-strukturer
- Teste med fysisk modul, PLS og HMI.

Utstyr:

- PC med TIA Portal og RJ45-port (evt. Adapter)
- Siemens S7 - 1500 PLS
- Siemens HMI (TP700 Comfort)
- Festo Didactic Handling Module
- Festo Didactic Conveyor Module
- Festo Didactic Stacking Module

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 4 – Handling Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Innføringsinfo:

Oppgaven dere skal utføre i denne labøvelsen vil være å plukke opp en puck med vakuumgriperen for så å løfte den opp fra en plattform og forflytte den til en annen plattform. Denne prosessen skal utføres gjennom tre sekvenser der hver enkelt sekvens skal startes med sin egen startknapp, derav tre startknapper på *Handling*-delen av HMIen. Denne laboppgaven bygger sterkt på prinsippene fra laboppgave 2 (*Stacking*), da det skal styres to stempler sekvensielt. Denne modulen har derimot et større antall sensorer, solenoidstyrte retningsventiler og startknapper som øker vanskelighetsgraden og det trengs mer betenkning.

Stempel *3A* styres av en retningsventil med to elektriske innganger, der 3M1 vil dytte stempelet ut, og 3M2 vil trekke stempelet inn. Det samme gjelder også for stempel *4A*. Det er montert to sensorer på begge stemplene, til sammen fire sensorer, som monitorerer posisjonen til stemplene. Her vil 3S1 og 4S1 bli aktivert når stempel 3A og 4A er i indre posisjon, og 3S2 og 4S2 vil aktiveres når stemplene er i ytre posisjon. Vakuumgriperen 5A er styrt av en retningsventil, 5M1, som vil aktivere vakuumet. Det vil si at når det ikke kommer høyt signal inn på 5M1 vil vakuumet være deaktivert.

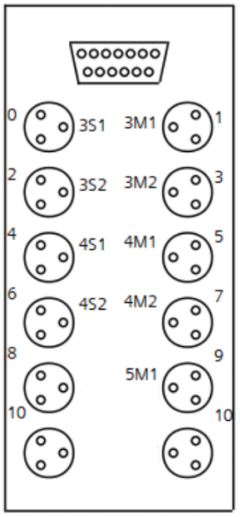
Stempel 3A opererer horisontalt og stempel 4A, som opererer vertikalt, er montert på enden av 3A. På enden av 4A er vakuumgriperen (5A) montert. Det kan være en god ide å studere modulen for å få et overblikk over hvordan disse stemplene er montert og vil fungere sammen. Se bilde på neste side.

Fag: *ELE304 PLS*
Laboratorieøving: *Lab 4 – Handling Modul*
Laboratorielærer: *[Navn her]*

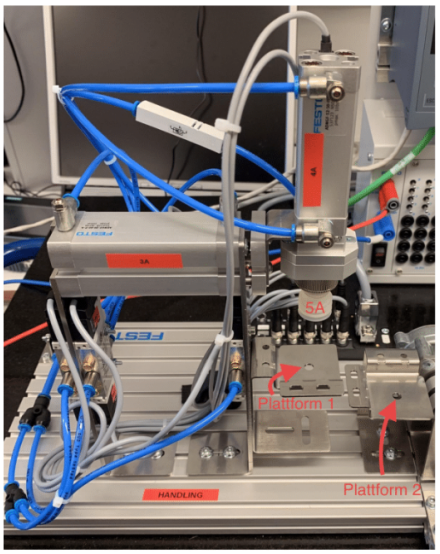
Høgskulen på Vestlandet

Innganger og utganger til Handling-modulen (ta i bruk "Tag" i koden):

PLS-Inputs	Multipin-Inputs	Komponent	Tag
I1.0	0	Sensor 3A, indre posisjon	3S1
I1.1	2	Sensor 3A, ytre posisjon	3S2
I1.2	4	Sensor 4A, indre posisjon	4S1
I1.3	6	Sensor 4A, ytre posisjon	4S2



PLS-Outputs	Multipin-Outputs	Komponent	Tag
Q1.0	1	Solenoid 3A, ytre posisjon	3M1
Q1.1	3	Solenoid 3A, indre posisjon	3M2
Q1.2	5	Solenoid 4A, ytre posisjon	4M1
Q1.3	7	Solenoid 4A, indre posisjon	4M2
Q1.4	9	Solenoid 5A, vakuum på	5M1



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 4 – Handling Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 1, fbHandlingModule

Fortsett i prosjektet fra *Lab 3 – Conveyor Modul*. Dere skal i denne oppgaven ta i bruk «*Handling*»-siden av HMIen.

Denne modulen skal kjøres sekvensielt og med relativt omfattende sekvenser og krav. Alle sekvensene skal kun kunne startes når sensorene *3S1* og *4S1* er aktivert (startposisjon), dvs. når stemplene *3A* og *4A* er i bakre posisjon. Sekvensene skal kun utføres i «riktig» rekkefølge (*Start1 -> Start2 -> Start3*).
Det anbefales å bruke sensorene til å sjekke at stemplene er i riktig posisjon til å utføre sekvensene på ønsket måte.

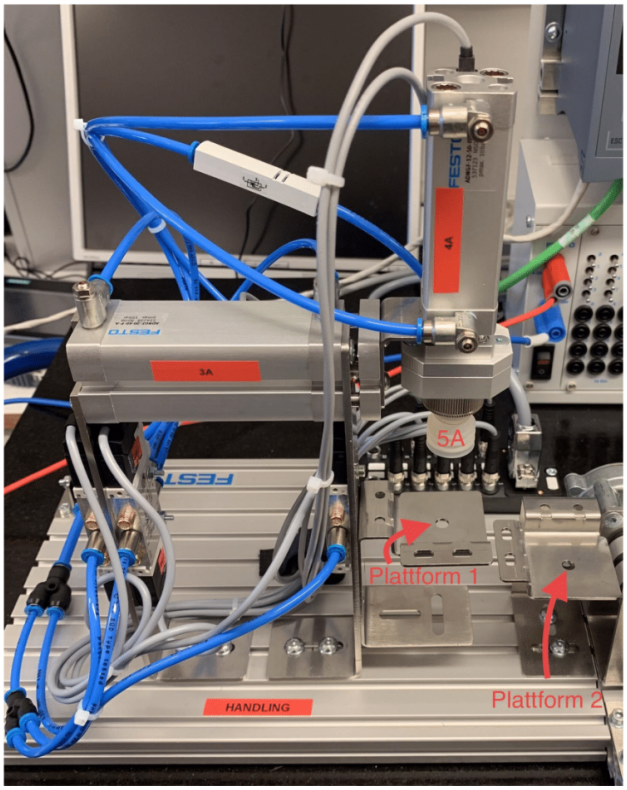
1. Opprett en funksjonsblokk *fbHandlingModule* og tilhørende statiske/temporære variabler for innganger/utganger fra HMI og PLS.
2. Skriv kode slik at når man trykker på *Start1* vil følgende sekvens utføres:
   Stempel *4A* skyves ut (dvs. *4M1* blir *true*) -> etter 1s skal vakuumgriperen aktiveres (*5M1* blir *true*) -> når vakuum er på trekkes stempel *4A* tilbake (*4M2* blir *true*).
   Her plukker man opp pucken fra plattform 1.
3. Skriv kode slik at når man trykker på *Start2* vil følgende sekvens utføres:
   Stempel *3A* skyves ut (*3M1* blir *true*) -> Stempel *4A* skyves ut (*4M1* blir *true*) -> vakuumgriperen deaktiveres (*5M1* blir *false*) når *4A* er i ytre posisjon (*4S2* er *true*) -> stempel *4A* trekkes tilbake -> stempel *3A* trekkes tilbake.
   Her plasseres pucken på plattform 2.
4. Når man trykker på *Start3* vil følgende sekvens utføres:
   Stempel *4A* skyves ut (*4M1* blir *true*) -> Stempel *3A* skyves ut (*3M1* blir *true*) -> stempel *4A* trekkes tilbake -> stempel *3A* trekkes tilbake.
   Her dyttes pucken ut på samlebåndet. OBS! Hvis samlebåndet ikke kjører når pucken dyttes ut på båndet kan det skje at pucken vrir seg.
5. *RunLight* skal lyse permanent når en sekvens kjører og mellom sekvensene skal *RunLight* blinke frem til neste startknapp trykkes inn og ny sekvens utføres. Ved fullført sekvens 3 skal *RunLight* slås av og *StopLight* slås på. *Stoplight* skal være på så lenge det ikke utføres sekvens.
6. *Reset*-knappen skal alltid kunne benyttes, uavhengig av hvor i prosessen man er. Når den aktiveres skal stemplene tilbake til startposisjon, *RunLight* slås av (selv ved blinking), *StopLight* slås på og man må begynne sekvensene fra sekvens 1 igjen. Dette innebærer også å deaktivere vakuumgriperen som vil si at i noen sekvenser vil pucken slippes. Vær obs på dette.
7. Implementer også nødstopp, med ønsket virkemåte.
8. Opprett datablokk og initier i Main.

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 4 – Handling Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Testing av programkode med fysiske moduler:

Når dere skal teste modulen legger dere en puck på plattform 1 med bunnen ned, slik at den kan plukkes opp av vakuumgriperen og flyttes til plattform 2 (se bilde under). Husk å sjekke at alle lys og knapper fungerer som de skal.



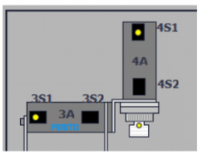
Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 4 – Handling Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

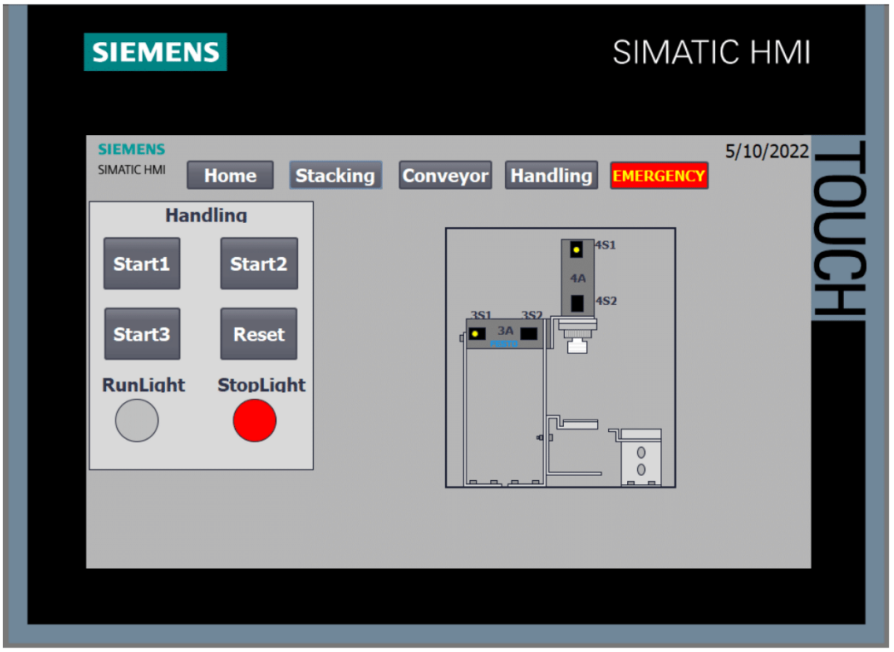
Testing av programkode med emulatorprogram i TIA-portal:

Det vil også være mulig å teste programkoden deres med en emulator om dere ikke har tilgang på de fysiske Meclab-modulene. Denne emulatoren utfører koden deres på tilnærmet samme måte som i virkeligheten. Om dere skal bruke emulatoren må dere laste over FBene deres over til "Mal - MecLab (Emulator)" og gjenopprette eventuelle TON/TOF-blokker om dere har brukt "Single Instance"-blokker.

Det er ikke animert pucker for denne emulatoren, ettersom det ikke er mulig å vite posisjon til puck ut ifra deres kode, så her må dere monitorere at bevegelsen til stemplene er riktig i henhold til sekvensene som ønskes i oppgaven. Det er lagt til lys på alle posisjonssensorene på stemplene og et lys på vakuumgriperen for å monitorere om den er aktivert eller ikke. Se bilde under.



Koden deres skal fungere på samme måte med denne emulatoren som med den fysiske Handling-modulen. Slik ser emulator-HMIen ut:



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 4 – Handling Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 2, Samkjøre med *Stacking*-og *Conveyor Modul*

Her skal *Handling*, *Conveyor* og *Stacking*-modulene kjøres sammen.
Dere skal også fjerne aktivering av samlebåndet ved startknapp i *Conveyor*-programmet deres, og i stedet kun bruke *OPT* som startsignal for samlebåndet. Utenom dette behøver dere ikke å endre på noe av koden i FBene deres.

Selv om modulene samkjøres må dere fremdeles aktivere Handling- og Stacking-modulen med knappene på HMI-skjermen.

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 4 – Handling Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 3, Teorispørsmål

1. Det er laget relativt strenge krav for kjøring av stempler samtidig i denne oppgaven. Hva er konsekvensene av at stemplene kan kjøres samtidig og/eller i feil rekkefølge?
2. Dere har nå samkjørt alle 3 modulene slik at de kjører samtidig. Hva er fordeler/ulemper ved å ha koden til hver modul i hver sin funksjonsblokk, kontra å ha alt i samme funksjonsblokk?

Svar:

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 4 – Handling Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Levering av labøvelsen:

Opprett en mappe der dere legger besvarelse på teorispørsmålene og en zippet mappe med TIA-prosjektet deres.

C.5 Lab 1 GRAPH - Stacking Modul

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 1 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Lab 1 – Stacking Modul

Gruppenummer:
Studenter:
Godkjent:

Formål:

- Lære å arbeide med fysisk utstyr
- Bruke inngang- og utgangssignaler fra fysisk PLS i programmering

Gjennomføring:

- Opprette et fungerende program for å styre fysisk utstyr
- Teste med fysisk modul, PLS og HMI

Utstyr:

- PC med TIA Portal og RJ45-port (evt. Adapter)
- Siemens S7 - 1500 PLS
- Siemens HMI (TP700 Comfort)
- Festo Didactic Stacking Module

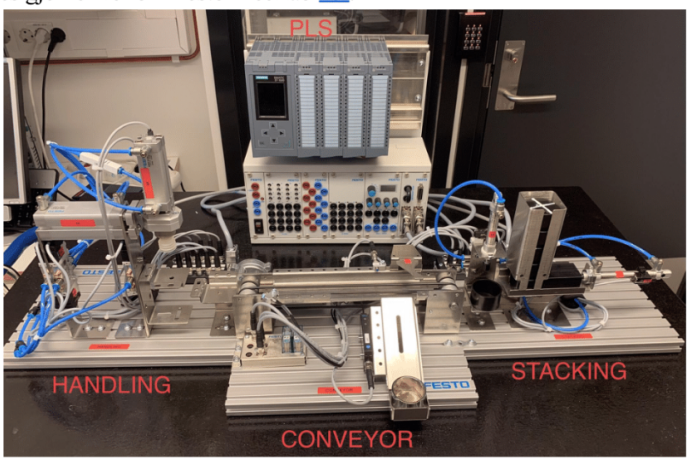
Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 1 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Innføringsinfo:

I denne labøvelsen skal dere bruke templaten "Mal - MecLab" og bli kjent med TIA-portal.
"Mal - MecLab" er et TIA-prosjekt som inneholder en PLS og HMI-skjerm tilsvarende de som skal brukes i praksis. Malen inneholder Tag-liste med alle inngangene og utgangene som trengs for å løse alle labøvelsene, samt en ferdig programmert HMI-skjerm som skal brukes til styring av disse 2 labøvelsene. Kommunikasjon mellom HMI og PLS er også konfigurert her, så dere behøver ikke å foreta noen endringer. Det er veldig viktig å ikke endre på noe av HMI-oppsettet i TIA-prosjektet deres etter dere har lastet ned malen, da den minste endring her vil føre til at dere ikke kan kan kommunisere mellom deres program og koden som er lastet inn på den fysiske HMIen. Om dere skulle være så uheldige å endre på noe i HMI-delen av TIA-prosjektet kan dere laste ned malen på nytt.

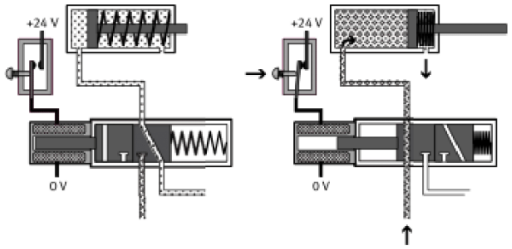
Labøvelsene dere skal gjennomføre vil bruke Festo MecLab, et sett på tre forskjellige moduler, Stacking, Conveyor og Handling, som skal etterligne prosesser fra produksjonsindustrien. Dere skal kun bruke Conveyor- og Stacking-modulene i disse to laboppgavene.
Se bilde under og les gjerne mer om Festo MecLab her.



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 1 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

I denne labøvelsen skal dere skrive kode for å styre prosessen til Stacking-modulen av MecLab. Denne modulen har som formål å presse to pucker sammen, en prosess som utføres ved hjelp av to pneumatiske (luftstyrte) stempel. Disse stemplene styres av solenoidstyrte pneumatiske retningsventiler, som styrer lufttrykket til forskjellig side av stempelet ved hjelp av et elektrisk signal fra PLSen. Disse må ha lufttrykk inn på luftinngangen og signalet inn på solenoidene styrer hvilken retning lufttrykket flyttes. Slike solenoidstyrte ventiler kan ha mange forskjellige konfigurasjoner, men de dere skal bruke er relativt simple. Se bilde under som viser hvordan *1M1* styrer *2A*.



Hentet fra Festo "563063_Teaching_With_MecLab"

Til å styre stempel *1A* brukes det en solenoidventil med to elektriske innganger, der signal inn på *1M1* vil dytte stempelet til ytre posisjon og *1M2* vil trekke stempelet til indre posisjon. Hvis både *1M1* og *1M2* mottar høyt signal samtidig vil ikke stempelet bevege seg, men beholde nåværende posisjon. Stempel *2A* fungerer litt annerledes da det er fjærbelastet og automatisk vil trekke seg tilbake til indre posisjon når det ikke kommer signal inn på *2M1*.

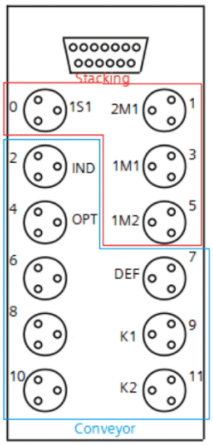
I koden dere skal utvikle i denne labøvelsen skal *1A* dytte en toppuck (med hvitt kryss) oppå en bunnpuck som ligger på en plattform. Puckene skal klemmes sammen ved hjelp av stempel *2A*. Stempel *1A* har også en sensor, *1S1*, som gir digital tilbakemelding (høyt signal) når stempelet er i bakre posisjon. Denne sensoren vil være smart å bruke for å sjekke at stempelet er i rett posisjon til å utføre en sekvens.

Fag: *ELE304 PLS*
Laboratorieøving: *Lab 1 – Stacking Modul*
Laboratorielærer: *[Navn her]*

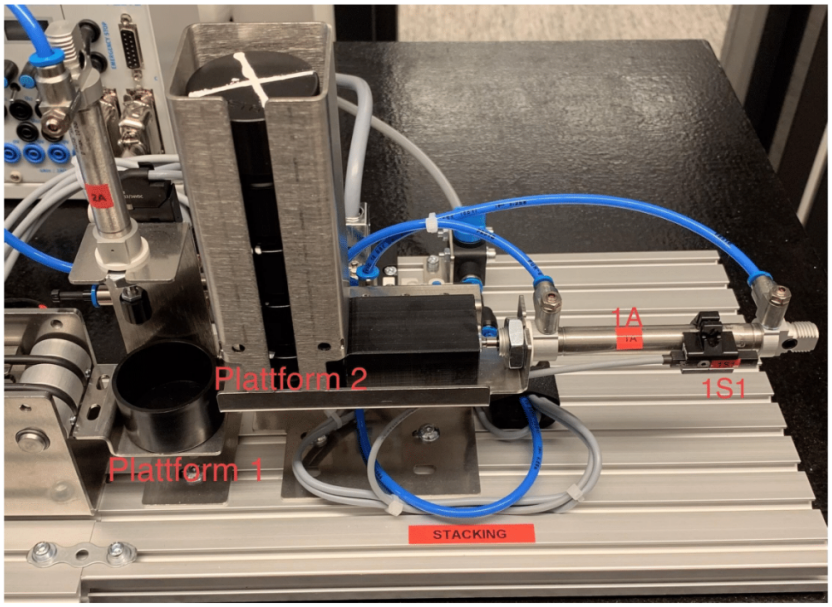
Høgskulen på Vestlandet

Innganger og utganger til Stacking-modulen (ta i bruk "Tag" i koden):

PLS-Inputs	Multipin-Inputs	Komponent	Tag
I0.0	0	Sensor 1A, indre posisjon	1S1



PLS-Outputs	Multipin-Outputs	Komponent	Tag
Q0.0	1	Solenoid 2A, ytre posisjon	2M1
Q0.1	3	Solenoid 1A, ytre posisjon	1M1
Q0.2	5	Solenoid 1A, indre posisjon	1M2



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 1 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 1, fbStackingModule

Dere skal i denne oppgaven ta i bruk Stacking-segmentet av HMIen. Startposisjon til stemplene vil være når *1A* og *2A* er i bakre posisjon. Da må ventilene ha *1M1=false*, *1M2=true* og *2M1=false*. Ved denne posisjonen vil også *1S1* være *true*.

1. Opprett en funksjonsblokk *fbStackingModule.*
2. Programmer blokken til å utføre prosessen beskrevet under når man trykker på startknappen og sensor *1S1* er *true*.
   Stempel *1A* skyves ut (dvs. *1M1* blir *true*) -> etter 2s trekkes stempel *1A* tilbake (*1M2* blir *true*) -> stempel *2A* skyves ut (*2M1* blir *true*) -> etter 2 sekund trekkes stempel *2A* tilbake (*2M1* blir *false*). Her vil da toppucken dyttes på bunnpucken, for så å presses sammen. Sekvensen skal kun kjøres en gang når startknappen på.
   Tips: Det kan være smart å bruke sensor *1S1* til å sjekke at stempel *1A* er i ønsket posisjon.
3. *Stop* skal stanse sekvensen til enhver tid og setter stemplene til startposisjon.
4. *RunLight* skal lyse så lenge sekvensen kjører og *StopLight* skal alltid lyse når sekvensen ikke kjører.
5. Initier i Main.

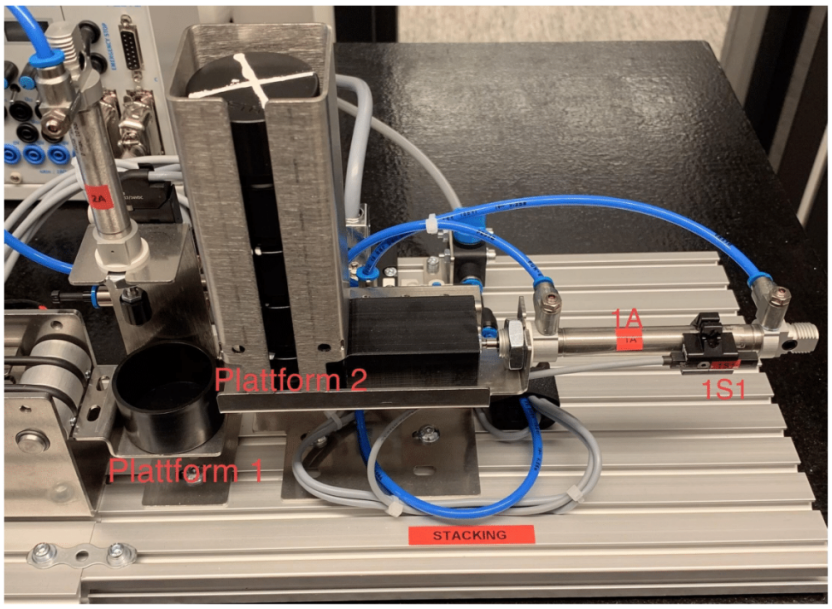
Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 1 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Testing av programkode med fysiske moduler:

Når dere skal teste programmet deres legger dere en bunnpuck på plattform 1, med bunnen ned, og en toppuck legges på plattform 2 med krysset opp. Se bilder under:

Bruk HMI-panelet til å starte prosessen og test om sekvensen utføres som spesifisert i oppgaven.



Fag: *ELE304 PLS*
Laboratorieøving: *Lab 1 – Stacking Modul*
Laboratorielærer: *[Navn her]*

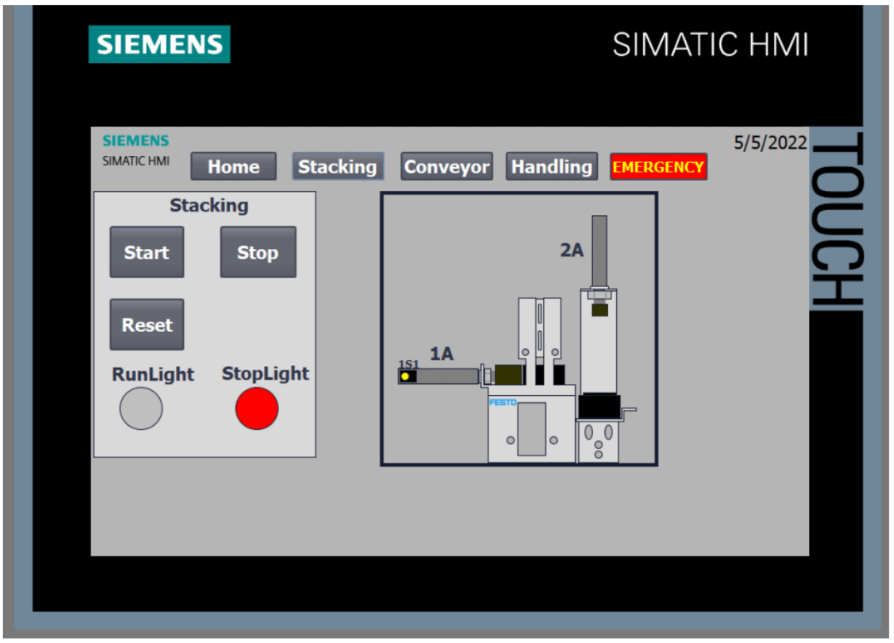
Høgskulen på Vestlandet

Testing av programkode med emulatorprogram i TIA-portal:

Det vil også være mulig å teste programkoden deres med en emulator om dere ikke har tilgang på de fysiske Meclab-modulene. Denne emulatoren utfører koden deres på tilnærmet samme måte som i virkeligheten. Om dere skal bruke emulatoren må dere laste over FBene deres over til "Mal - MecLab (Emulator)" og gjenopprette eventuelle TON/TOF-blokker om dere har brukt "Single Instance"-blokker.

I startposisjon vil toppucken være plassert som vist på bildet under og etter vellykket gjennomføring av koden vil puckene ha blitt presset sammen. For å flytte toppucken tilbake til startposisjon igjen trykker dere på "Reset"-knappen.

Slik ser emulator-HMIen ut:



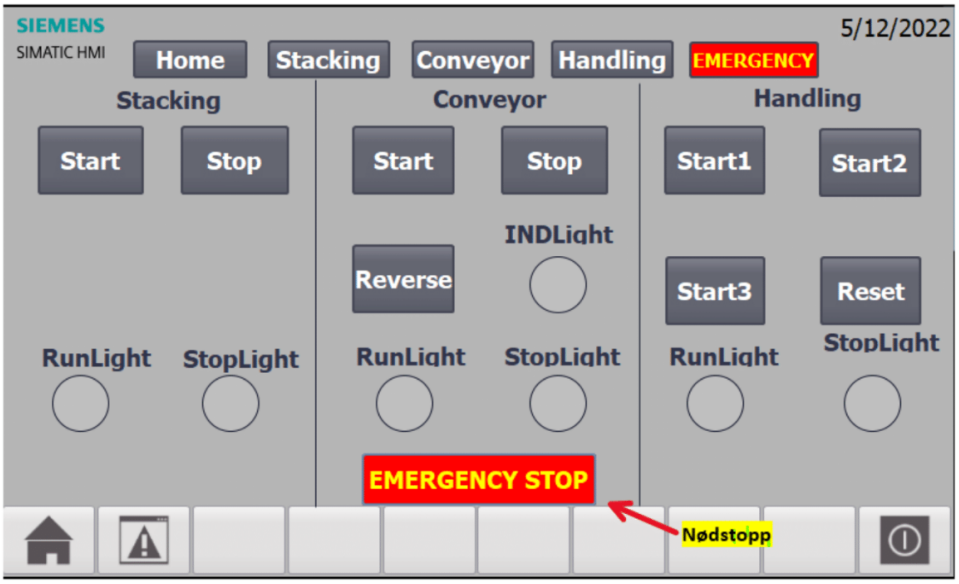
Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 1 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 2, Nødstopp

Nødstoppknappen er en vesentlig del av de fleste prosesser og ofte er nødstoppkoden meget omfattende. Det finnes utrolig mange forskjellige standarder i arbeidslivet på dette og det vil ikke gås noe dypere inn i forskjellige løsninger. I labøvelsene fremover behøver dere ikke å skrive spesielt omfattende nødstoppkode, men vi vil gjerne at dere tenker gjennom hva som skal skje når dere aktiverer nødstoppen. Skal hele prosessen stoppe? Bør man plassere nødstoppkoden i Main eller i FB? Skal man stoppe signalet til alle ventiler eller flytte stemplene til en spesiell posisjon? Her det mange løsninger. Ha dette i bakhodet når dere programmerer nødstoppfunksjon på de gjenstående labøvelsene.

1. Reflekter rundt hva som kan være en god løsning for hva nødstoppen skal gjøre i denne modulen.
2. Implementer den løsningen dere kom frem til i programkoden.
3. Test om nødstoppkoden virker som ønsket.



Nødstoppen er inkorporert i HMIen. Det er ingen fysisk nødstoppknapp.

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 1 – Stacking Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Levering av labøvelsen:

Opprett en mappe der dere legger besvarelse på teorispørsmålene og en zippet mappe med TIA-prosjektet deres.

C.6 Lab 2 GRAPH - Conveyor Modul

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Lab 2 – Conveyor Modul

Gruppenummer:
Studenter:
Godkjent:

Formål:

- Lære å arbeide med fysisk utstyr
- Bruke inngang- og utgangssignaler fra fysisk PLS i programmering
- Styre motor vha. releer

Gjennomføring:

- Opprette et fungerende program for å styre fysiske element
- Teste med fysisk modul, PLS og HMI.

Utstyr:

- PC med TIA Portal og RJ45-port (evt. Adapter)
- Siemens S7 - 1500 PLS
- Siemens HMI (TP700 Comfort)
- Festo Didactic Conveyor Module
- Festo Didactic Stacking Module

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

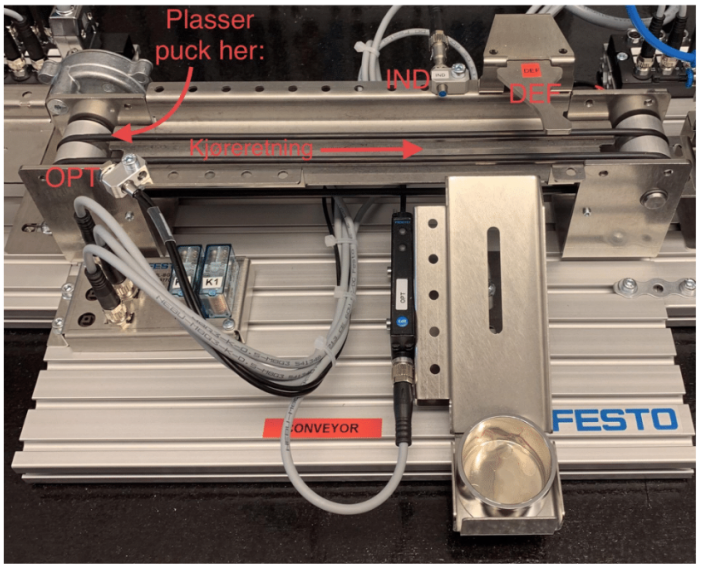
Høgskulen på Vestlandet

Innføringsinfo:

I denne laboppgaven skal dere kjøre pucker på et samlebånd og sortere svarte og sølvfargede pucker fra hverandre. Modulen har to inngangssignaler til PLSen, en optisk sensor (*OPT*) som gir tilbakemelding når et objekt oppdages ved starten av samlebåndet, og en induktiv sensor (*IND*) midt på samlebåndet som gir tilbakemelding ved deteksjon av en sølvfarget puck siden disse har et metallisk belegg. Det er også tre separate element på utgangssiden; en solenoidstyrt deflektor (*DEF*) som vil dytte pucker av samlebåndet, et rele til å kjøre motoren og et rele til å reversere motorretningen.

Når man styrer en motor, i dette tilfellet til samlebåndet, med en PLS bør man bruke releer. Man bruker disse ettersom strømmen PLSen kan supplere fra sine utganger ofte er for lav til å drive en motor. Releet fungerer som en bryter der man bruker et lite signal (fra PLS) til å aktivere en solenoid som igjen kobler sammen inngangen og utgangen på releet. På denne måten styres en større strøm som driver selve motoren. *Conveyor*-modulen har to releer, *K1* som styrer motoren av/på og *K2* som vil kjøre motoren i revers (*K2* vil kun kjøre samlebåndet i revers hvis *K1* også er aktiv).

Samlebåndet kjører de svarte klossene videre til stacking-modulen. I oppgave 2 i denne labøvelsen skal dere binde sammen disse to modulene og teste at koden fra forrige labøvelse samkjører med koden dere vil skrive i denne labøvelsen.



Bilde av Conveyor-modulen

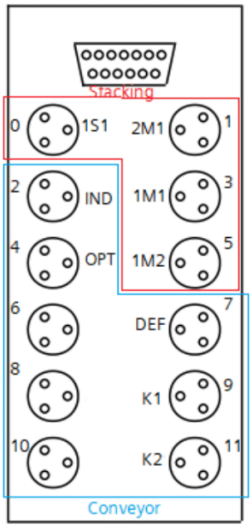
Fag: *ELE304 PLS*
Laboratorieøving: *Lab 2 – Conveyor Modul*
Laboratorielærer: *[Navn her]*

Høgskulen på Vestlandet

Innganger og utganger til Conveyor-modulen (ta i bruk "Tag" i koden):

PLS-Inputs	Multipin-Inputs	Komponent	Tag
I0.1	2	Induktiv sensor	IND
I0.2	4	Optisk sensor	OPT

PLS-Outputs	Multipin-Outputs	Komponent	Tag
Q0.3	7	Deflektor	DEF
Q0.4	9	MotorStart	K1
Q0.5	11	Endre Motorretning	K2



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

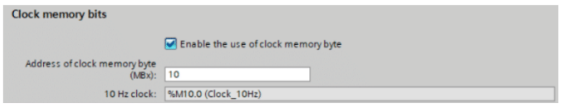
Oppgave 1, fbConveyorModule

Fortsett i prosjektet fra Lab 2 – Stacking Modul. Dere skal i denne oppgaven ta i bruk «Conveyor»-segmentet av HMIen.

Det skal legges pucker av to forskjellige materialer på båndet, der de sølvfargede kan oppdages av en induktiv sensor (*IND*) og de svarte ikke kan. Slik skal man sortere vekk de sølvfargede puckene og la de svarte puckene passere forbi en deflektor. Det må også tas hensyn til at samlebåndet skal kunne kjøres i revers, selv om dette ikke har noen reell hensikt i denne modulen.

1. Opprett en funksjonsblokk *fbConveyorModule.*
2. Programmer blokken slik at samlebåndet begynner å kjøre (*K1* blir *true*) når man trykker på startknappen og sensor *OPT* er *true*. Hvis *OPT* ikke er *true* skal ikke samlebåndet starte.
3. RunLight skal være aktivert når *K1* er *true* og *StopLight* skal alltid være aktivert når *K1* er *false*.
4. Nødstoppen og stoppknappen skal stanse samlebåndet, slå av *RunLight* og skru på *StopLight*.
5. De sølvfargede puckene skal sorteres vekk ved bruk av sensoren *IND* og deflektoren *DEF,* de svarte puckene skal kjøres videre til stacking-modul. Deflektoren skal holdes nede i 1 sekund etter at *IND* har blitt aktivert.
6. Når sensoren *IND* har høyt signal skal *INDLight* være aktivert.
7. Når man holder inne *Reverse*-knappen skal motoren kjøre i revers (*K2* blir *true*). Når samlebåndet kjøres i revers, skal også *RunLight* blinke 1 gang i sekundet (1Hz). *K2* kan kun kjøres samtidig som *K1* for ønsket effekt.
   OBS! Det kan hende at blinkingen ender med at *RunLight* er av, i så fall må lyset skrus på igjen automatisk når dere trykker på neste startknapp.
8. Initier i Main.

Tips:
For å få til blinkefunksjonen på *RunLight* er det en god ide å bruke *Clock Memory Bits* i PLSen. Disse er *MemoryBits* som skrus av/på med intervaller som samsvarer med frekvensen som står oppgitt i navnet på Clock Bit'en (Eks. *Clock_10Hz* skrus true/false 10 ganger per sekund).



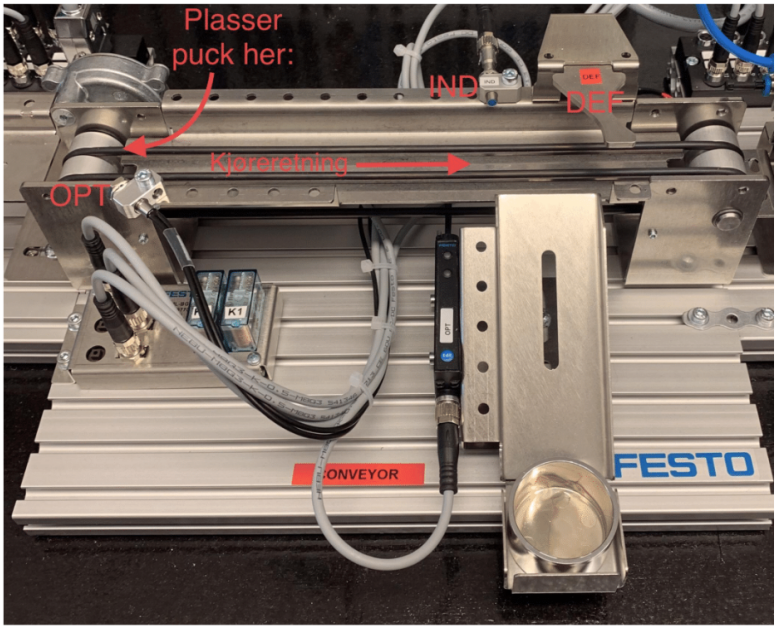
Clock memory bits må aktiveres ved å trykke på selve PLSen under *Device configuration,* og dere må sette *Address of clock memory byte* til eksempelvis 10, slik at ikke klokkebitene allokeres til en minnebit som allerede brukes. Når dette er gjort blir klokkebitene automatisk lagt til i *Tag table* (husk å kompilere PLSen) og kan tas i bruk direkte med navn i koden.

Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Testing av programkode med fysiske moduler:

Når dere skal teste modulen legger dere en puck ved starten av samlebåndet (som vist på bildet under), slik at den blir observert av OPT. Husk å teste programmet med både svarte og sølvfargede pucker og sjekk at alle lys og knapper fungerer som de skal.



Fag: *ELE304 PLS*
Laboratorieøving: *Lab 2 – Conveyor Modul*
Laboratorielærer: *[Navn her]*

Høgskulen på Vestlandet

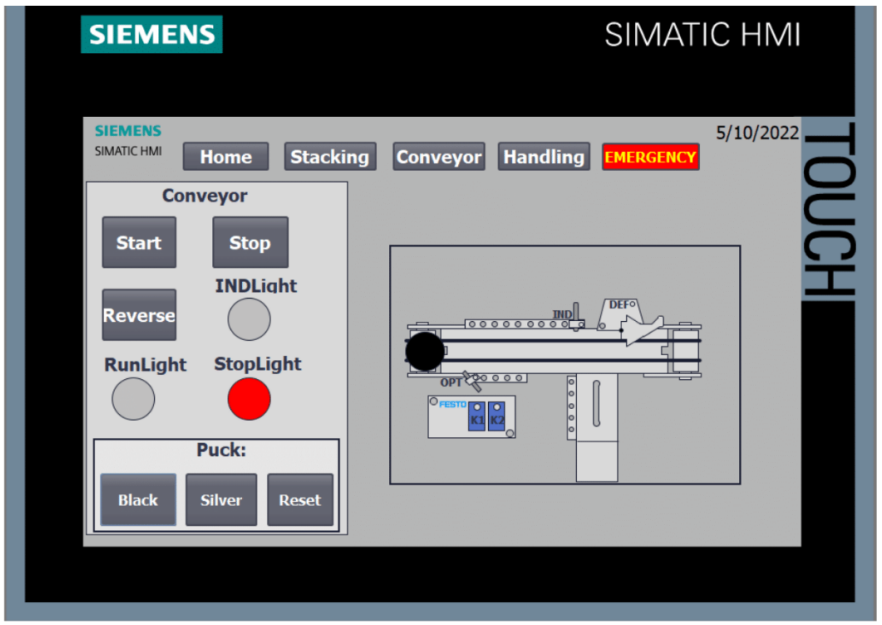
Testing av programkode med emulatorprogram i TIA-portal:

Det vil også være mulig å teste programkoden deres med en emulator om dere ikke har tilgang på de fysiske Meclab-modulene. Denne emulatoren utfører koden deres på tilnærmet samme måte som i virkeligheten. Om dere skal bruke emulatoren må dere laste over FBene deres over til "Mal - MecLab (Emulator)" og gjenopprette eventuelle TON/TOF-blokker om dere har brukt "Single Instance"-blokker.

Det er lagt til knapper på emulator-HMIen for å bestemme hvilken type puck som blir "lagt på samlebåndet". Man må trykke på "Silver" eller "Black" for at en animert puck av den ønskede fargen/typen legges på båndet og OPT blir true. Ved fungerende programkode vil de sølvfargede puckene sorteres ut på rampen og de svarte puckene vil kjøres ut av samlebåndet og "forsvinne". Trykk på "Reset"-knappen for å fjerne pucker i fra samlebåndet/rampen. Det er kun mulig å simulere èn puck av gangen med emulatoren. IND vil kun settes til true når en animert sølvfarget puck passerer posisjonen til IND-sensoren, og OPT er kun true når en puck er posisjonert i begynnelsen av samlebåndet.

Koden deres skal fungere på samme måte med denne emulatoren som med den fysiske Conveyor-modulen.

Slik ser emulator-HMIen ut:



Fag:	*ELE304 PLS*
Laboratorieøving:	*Lab 2 – Conveyor Modul*
Laboratorielærer:	*[Navn her]*

Høgskulen på Vestlandet

Oppgave 2, Samkjøre med *Stacking Modul*

Dere skal i denne oppgaven prøve å kjøre både *Stacking*-modulen og *Conveyor*-modulen sammen.
Dette gjør dere enkelt ved å dra FBene for begge programmene i Main. Dere behøver ikke å endre på noe av koden i FBene.

Det er dessverre ikke mulig å utføre denne oppgaven ved bruk av emulator. Om ønskelig kan dere teste begge programmene samtidig med emulatoren, men det er ikke mulig å kjøre en puck fra en modul til en annen.

Levering av labøvelsen:

Opprett en mappe der dere legger besvarelse på teorispørsmålene og en zippet mappe med TIA-prosjektet ditt.

Appendiks D - Instruks til labingeniør

Øvrige bilder i instruksen er tatt selv eller er skjermdump fra egne prosjekt i TIA Portal. Instruksen ligger også vedlagt som .docx-fil.

Instruks til labingeniør

Forarbeid:

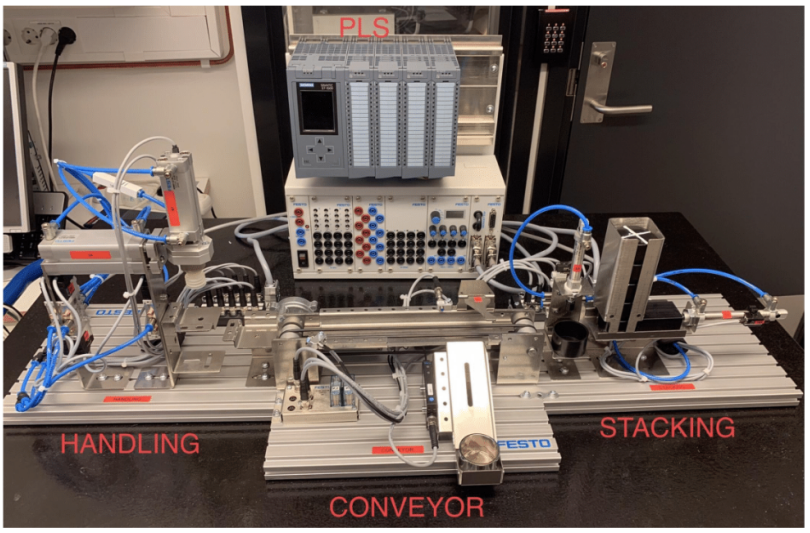
I det en labøvelse skal gjennomføres må ansvarlig labingeniør gjøre følgende forarbeid før studentene kan operere de fysiske MecLab-modulene.

- Kompressor må skrus på og stilles inn på minimum 3 Bar.
- Alle lukkeventiler på modulene må åpnes.
- Strøm til PLS og HMI må skrus på med bryter
- Tilse at PC som skal kobles mot PLS har RJ45-port, eller USB-C ved bruk av adapter

Når det kommer til opprettelse av studentgrupper som skal gjennomføre labøvelsene sammen foreslår vi et antall på 2-3 studenter. Dette er også anbefalingen fra Festo, som står skrevet i "563063_Teaching_with_MecLab" som følger med i dokumentasjonen til MecLab (ligger også i mappen "Vedlegg til Labingeniør".

Oppkoblingen mellom modulene og PLS skal være fullført, slik at det ikke behøves å gjøre noe fysisk oppkobling i det man skal operere modulene. Om man ønsker å bytte PLS, anbefales det å bruke Festo sitt system "EduTrainer Universal" sammen med en Siemens S7-1500, som er det som brukes til modulene i dag. Den eneste koblingen mellom Festo modulene og PLSen er to datakabler med SYSLINK-port i ene enden og sub D-port i den andre. Disse går fra port A->Multi-pin tilkobling montert på Stacking-modul og fra port B-> Multi-pin tilkobling montert på Handling-modul.

Se bilde av MecLab-modulene under.



Kjente problemer:

Vi har kjennskap til et visst antall problemer studentene kan møte på ved opplasting til PLS og idriftsettelse. De er presentert i listeform, med mulig løsning på problemet, under:

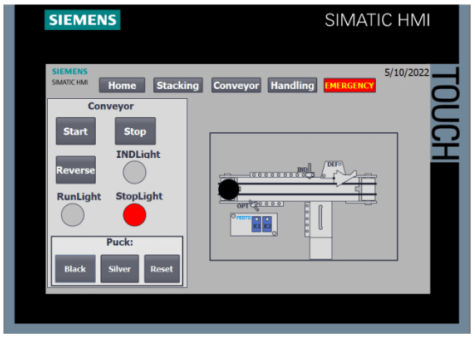
- Finner ikke riktig IP-adresse/IP-adressen er opptatt ved tilkobling til PLS
  - Her er det bare å trykke *OK* så blir IP-adresse tildelt
- Pop-up vindu “Too many tags” i HMI ved oppstart
  - Skjer ettersom man ikke har riktig lisens
    Pop-up vinduet forsvinner etter et par sekunder
- Stempler beveger seg ikke selv om styringsventilene er i riktig posisjon
  - For lavt trykk på kompressoren (Minimum 3 bar).
  - Stengeventilene til modulene ikke er åpnet.
  - Luftslange har løsnet ved ventil/t-kobling
- Ingen forbindelse mellom PLS og HMI (ingenting skjer når man trykker på knapper på HMI)
  - Kontrollere om det er riktig “connection” mellom tags i HMI og PLS under “HMI tags”
  - Laste opp konfigurasjonen fra “Mal - MecLab” til PLS og HMI på nytt

Disse punktene må derfor kontroll-sjekkes før/under hver lab for å forhindre forsinkelser.

Info om medfølgende PLS-og HMI-program:

For at gjennomføring av labøvelsene skal foregå på enklest mulig vis for studentene, suppleres et templat i TIA-portal. Dette templatet, kalt “MecLab - Mal” inneholder riktig PLS med tilhørende IO-moduler og riktig HMI. Virtuell forbindelse mellom disse i TIA-portal skal også være riktig, slik at man ikke behøver å oppdatere noe som helst på HMIen.
Det er også opprettet Tag table for alle innganger og utganger fra PLS til MecLab-modulene, Tags som brukes i hver labøvelse står spesifisert i oppgaveteksten.
Denne malen tenkes brukt ved gjennomføring av labbene med de fysiske modulene.

Om det ikke er mulig å gjennomføre labøvelsene med de fysiske modulene, tar man i bruk “Mal - MecLab (Emulator)”. Dette er et TIA-prosjekt som inneholder det samme som i “Mal - MecLab” i tillegg til programmer for å simulere de fysiske modulene. Det er laget en funksjonsblokk med emulatorkode, samt et HMI-skjermtemplat, til hver modul. Hvert HMI-skjermtemplat inneholder touch-knapper til styring av modulene, driftlys og en animert modul med tilsvarende virkemåte som de fysiske modulene. Det er en liten fare for å møte på bugs ved bruk av emulatorene, da det er mulig å tvinge emulatorene inn i tilstander som ikke er mulig på de fysiske modulene.
Se bilde under for eksempel på animert Conveyor-modul med tilhørende knapper og lys.



Det anbefales på det sterkeste at man legger til rette for å gjennomføre labøvelsene med de fysiske modulene for studentenes læring og utbytte.

Funksjonsblokkene som inneholder emulatorprogrammene er passordbeskyttet. HMI-delen av prosjektet er ikke passordbeskyttet og av denne grunn er det ekstremt viktig at studenter ikke endrer på noe under HMI-delen i TIA-portal, da dette kan føre til at emulatorprogrammene og animasjonene til disse vil slutte å fungere. Om dette skulle skje, anbefales det å laste ned emulatormalen på nytt og overføre studentenes funksjonsblokker til denne malen.

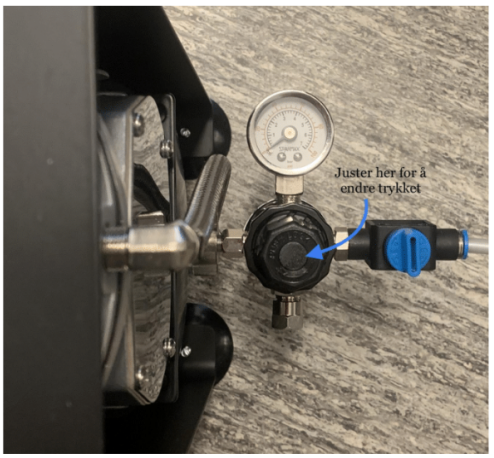
Passord til passordbeskyttede funksjonsblokker:
[redacted]

Kompressor

Sammen med MecLab-pakken ble det levert en luftkompressor fra Festo til å drive pneumatikken til modulene. Denne er laget for å produsere så lite støy at den kan brukes i klasseromssammenheng. Av denne grunn er det ønskelig å bruke denne kompressoren.

Kompressoren kom levert med type-B støpsel og tar 120VAC 60Hz. For å kunne bruke denne kompressoren må man da ta i bruk en spenningsomformer (se bilde under).

I tillegg til å skru på denne omformeren må også kompressoren skrus på med en godt skjult bryter (se bilde under). Husk også å justere barometeret slik at det suppleres minimum 3 Bar, slik at pneumatikken vil fungere som ønsket.



Gjennomføring ved SCL eller GRAPH:

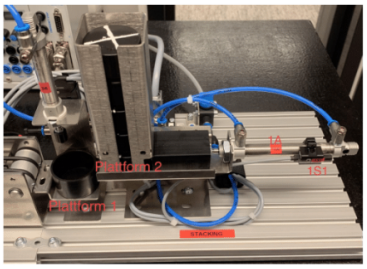
Det har blitt utviklet labøvelser, med løsningsforslag, som kan gjennomføres enten ved SCL eller GRAPH. Ettersom det er langt flere studenter som lærer SCL og opplæring i GRAPH fases ut, er det utviklet færre oppgaver for GRAPH.
Til sammen er det utviklet fire labøvelser for SCL og to for GRAPH. For SCL-øvelsene vil alle tre modulene tas i bruk, i tillegg til en innførende labøvelse. Ved GRAPH vil man kun ta i bruk Stacking- og Conveyor-modulene, da Handling-modulen har meget omfattende kode som vil være krevende og muligens utenfor studentenes kunnskapsområde.

MecLab Forarbeid:

Før studentene begynner å arbeide med de fysiske MecLab-modulene skal de gjennom en labøvelse kalt “MecLab Forarbeid”. I denne øvelsen skal de utvikle et program for å teste at kommunikasjonen mellom PLS og HMI fungerer, og et enkelt program der de skal bruke tidsforsinkelse og holdefunksjon. Når studentene har gjennomført denne labøvelsen, vil de ha gjennomgått vesentlige funksjoner som brukes videre i faget, og ha bedre forutsetninger for å løse labøvelse 2 til og med 4.

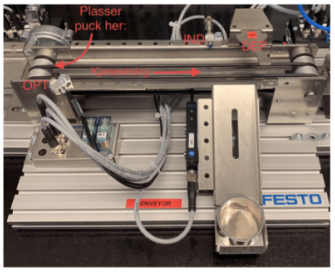
Stacking modul:

Dette er modulen som brukes i labøvelse 2 (labøvelse 1 ved GRAPH-øvelse), den første labøvelsen der MecLab-modulene tas i bruk. Denne modulen har som formål å dytte sammen en toppuck og bunnpuck til en samlet puck, ved hjelp av to stempler.
Bilde av Stacking-modul:



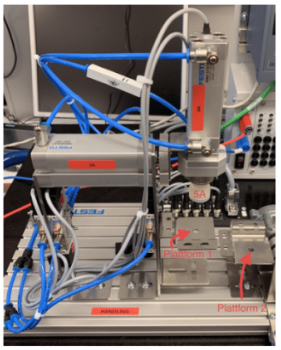
Conveyor modul:

Dette er modulen som brukes i labøvelse 3 (labøvelse 2 ved GRAPH-øvelse). Modulen består av et samlebånd med en optisk og en induktiv proximity-sensor, samt en solenoidstyrt deflektor. Formålet til denne modulen vil være å kjøre svarte og sølvfargede pucker på samlebåndet, for så å sortere vekk de sølvfargede puckene ved å bruke den induktive sensoren og deflektoren.
Bilde av Conveyor-modulen:



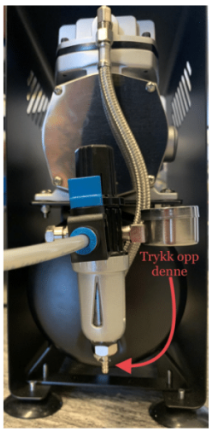
Handling modul:

Dette er modulen som brukes i labøvelse 4. Det er ikke blitt laget labøvelse for GRAPH med denne modulen, da det vil bli et meget omfattende program utenfor GRAPH-studentenes kunnskapsnivå. Modulen er meget lik Stacking-modulen, med to stempler og en vakuumgriper. Handling-modulen har også posisjonsdetektor på ytre og indre posisjon på de to stemplene, som vil tas i bruk i programmet som skal utvikles av studentene. Formålet til denne modulen er å plukke opp en puck, ved hjelp av vakuumgriperen, og plassere den på en annen plattform for så å dytte pucken ut på samlebåndet.
Bilde av Handling-modulen:



Etterarbeid

Når labøvelsene har blitt gjennomført gjenstår det kun å skru av PLS, HMI og kompressor. Alle stengeventiler bør settes i lukket posisjon. Lufttanken på kompressoren må tømmes, noe som gjøres ved å presse opp den lille "tappen" under barometeret (se bilde under).



Det kan også være smart å fjerne programmet som ligger på PLSen, slik at ikke neste gruppe laster dette ned og plagierer det. Dette gjelder også spesielt hvis man har lastet opp "Løsningsforslag MecLab (SCL)" for å gjennomføre demonstrasjon av modulene.

Teorispørsmål - løsningsforslag:

Til hver av de fire SCL-labbene har det blitt laget teorispørsmål, under er løsningsforslag til disse spørsmålene beskrevet. Labingeniør må gjerne endre, legge til eller fjerne teorispørsmål om de foreslåtte spørsmålene ikke stemmer overens med fungerende pensum.
Det er ikke laget teorispørsmål til GRAPH-labøvelsene, da studentene som vil gjennomføre disse kun skal få et innblikk i PLS og da sannsynligvis ikke innehar kompetansen til å løse teorispørsmålene.

Lab 1 - MecLab Forarbeid:

1. **Hvorfor skal man alltid bruke interne variabler når man skriver kode i TIA-portal?**
   - Funksjonsblokker blir mer generelle og overførbare, lettere å feilsøke når man ikke programmerer direkte med innganger og utganger på PLSen.

2. **Hva er hensikten med TAG-table i PLS og HMI?**
   - TAG-table brukes for å navngi og deklarere innganger, utganger og minneceller. Verdiene deklarert i PLSen sitt tag table er globale, det vil si de kan nås alle steder i PLSen.

3. **Hva er forskjellen på TON- og TOF delay?**
   - TON-delay (time-on), forsinker et signal inn med en ønsket tid. Det vil si at hvis man trykker på en knapp og dette signalet går inn i en TON-blokk, forsinkes signalet ut. TOF-delay vil beholde utgangssignalet høyt i ønsket tid etter innsignalet har gått lavt. Det vil si at om man slår av et lys og dette signalet går inn i en TOF-blokk, vil signalet holdes høyt til den ønskede tiden har passert og signalet blir lavt ut.

Lab 2 - Stacking Modul:

1. **Hva er fordeler/ulemper med å bruke Case-struktur kontra If/Else-statements?** - Når man skal utføre sekvensiell programkode vil det bli meget tungvint å utføre mange sekvenser i rekkefølge med If/Else. Case-struktur benytter en integer variabel som representerer forskjellige caser med kode som skal utføres. Det vil si at ved omfattende sekvensiell programkode kan det hoppes mellom caser etter oppfylte krav eller gjennomført kode før neste sekvens skal gjennomføres. Dersom en If/Else-løsning skal gjøre det samme kreves det mange statements, og blir som nevnt mer tungvint. Case vil derfor fungere raskere ved sekvensiell kode dersom casene blir tildelt verdier på en ryddig og god måte, mens If/Else fungerer bedre for det som resulterer i boolske tilstander.

2. **Hvordan fungerer solenoid-ventiler? -** En solenoid er en spole av isolert ledningstråd brukt som elektromagnet. En pneumatisk solenoidstyrt retningsventil styres av spenningen på solenoiden; når solenoiden får tilført spenning endrer den posisjonen på ventilen. Luften som suppleres på inngangssiden til ventilen vil da styres i den retningen som ventilens posisjon tilsier (dvs. stempel ut eller stempel inn).

3. **Hva skjer hvis du gir høyt signal til begge sider av en retningsventil (1M1 og 1M2)?**
   - Hvis man supplerer spenning på begge sider av en 2-sidet retningsventil vil den forbli i den posisjonen den var i først. Det vil si at stempelet som ventilen styrer, ikke vil endre posisjon.

Lab 3 - Conveyor Modul:

1. **Hvorfor bruker man rele til å styre motorer?**
   - Man bruker de ettersom strømmen PLSen kan supplere fra sine utganger er for lav til å drive en motor. Releet fungerer som en bryter der man bruker et lite signal (fra PLS) til å aktivere en solenoid som igjen kobler sammen inngangen og utgangen på releet og styrer en større strøm (hovedstrøm) til motoren.

2. **Hvordan fungerer en optisk sensor?**
   - Optiske proximity sensorer har en sender og en mottaker, og bruker rødt eller infrarødt lys i samarbeid med elektriske komponenter og moduler til å detektere objekt i mellom sender og mottaker.

3. **Hvordan fungerer en induktiv sensor?**
   - Induktive sensorer oppretter et elektromagnetisk felt foran sensoren ved hjelp av en oscillator. Når et objekt av et elektrisk ledende materiale passerer feltet blir et elektrisk signal sendt ut.

Lab 4 - Handling Modul:

1. **Vi har laget relativt strenge krav for kjøring av stempler samtidig i denne oppgaven. Hva er konsekvensene av at stemplene kan kjøres samtidig og/eller i feil rekkefølge?**
   - Hvis man kjører to stempler samtidig på Handling-modulen risikerer man å rive i stykker vakuumgriperen da den ene plattformen er plassert så høyt at griperen tar borti den når stempel 4A er i ytre posisjon.

2. **Du har nå samkjørt alle 3 modulene slik at de kjører samtidig. Hva er fordeler/ulemper ved å ha koden til hver modul i hver sin funksjonsblokk kontra å ha alt i samme funksjonsblokk?**
   - Det er mye lettere å feilsøke når man deler opp de forskjellige programmene i forskjellige funksjonsblokker. Man kunne også delt opp koden inni

funksjonsblokkene mer ved å bruke egne funksjoner. Dette er også meget god programmeringsskikk.

Vedlegg til Labingeniør:

Tilhørende denne instruksen følger også en mappe kalt “Vedlegg til Labingeniør” som labingeniør skal ha tilgang til ved gjennomføring av labøvelsene. Denne mappen inneholder:

- “Lab 1 - MecLab Forarbeid” - Word-fil av labøvelse 1 (SCL)
- “Lab 2 - Stacking Module” - Word-fil av labøvelse 2 (SCL)
- “Lab 3 - Conveyor Module” - Word-fil av labøvelse 3 (SCL)
- “Lab 4 - Handling Module” - Word-fil av labøvelse 4 (SCL
- “Lab 1 GRAPH - Stacking Module” - Word-fil av labøvelse 1 (GRAPH)
- “Lab 2 GRAPH - Conveyor Module” - Word-fil av labøvelse 2 (GRAPH)
- “Mal - MecLab” - TIA-prosjekt
- “Mal - Meclab (Emulator)” - TIA-prosjekt med emulatorene
- “Løsningsforslag MecLab (SCL)” - TIA-prosjekt med løsningsforslag til labøvelsene med SCL
- “Løsningsforslag MecLab (GRAPH)” - TIA-prosjekt med løsningsforslag til labøvelsene med GRAPH
- “Løsningsforslag MecLab (SCL) - Emulator” - TIA-prosjekt med løsningsforslag til labøvelsene med SCL, med fullt fungerende emulatorer
- “Festo dokumentasjon” (Mappe med dokumentasjonen fra CD-rom som fulgte med MecLab)
- “Video - MecLab demonstrasjon” - Video av MecLab med fullt fungerende kode

OBS! Alle TIA-prosjektene er i zippede mapper, disse må da unzippes før de kan tas i bruk.

Appendiks E - Andre filer

Vedlagt ligger zippet mappe med følgende element:

Labøvelser og Instruks:

- Instruks til labingeniør.docx
- Lab 1 - MecLab Forarbeid.docx
- Lab 2 - Stacking Modul.docx
- Lab 3 - Conveyor Modul.docx
- Lab 4 - Handling Modul.docx
- Lab 1 GRAPH - Stacking Modul.docx
- Lab 2 GRAPH - Conveyor Modul.docx

Programmer i TIA Portal (prosjekt i TIA Portal i zippete mapper):

- Mal - MecLab.zip
- Mal - MecLab (Emulator).zip
- Løsningsforslag MecLab (SCL).zip
- Løsningsforslag MecLab (SCL) - Emulator.zip
- Løsningsforslag MecLab (GRAPH).zip

Øvrige dokumenter:

- Møtereferat.pdf
- Timelister.pdf
- Undersøkelse ELE304.pdf

Vedlegg Labingeniør:

- Lab 1 - MecLab Forarbeid.docx
- Lab 2 - Stacking Modul.docx
- Lab 3 - Conveyor Modul.docx
- Lab 4 - Handling Modul.docx
- Lab 1 GRAPH - Stacking Modul.docx
- Lab 2 GRAPH - Conveyor Modul.docx
- Mal - MecLab.zip
- Mal - Meclab (Emulator).zip
- Løsningsforslag MecLab (SCL).zip
- Løsningsforslag MecLab (SCL) - Emulator.zip
- Løsningsforslag MecLab (GRAPH).zip
- Filmappe: Festo dokumentasjon
- Video - MecLab demonstrasjon.mp4

Elementene i filmappen “Festo dokumentasjon” fulgte med MecLab-pakken på CD-rom. Flere av dokumentene her, blant annet “563063_Teaching_With_MecLab”, er kun tillatt å bruke i undervisningssammenheng og kan ikke publiseres offentlig. Av denne grunn brukes ikke “563063_Teaching_With_MecLab” som referanse.