Modul Praktikum Perancangan dan Pemrograman Otomasi Proses Industri Menggunakan *Programmable Logic Controller* Dan Program Simulasi Factory I/O

Program Studi Teknik Elektro

Universitas Pelita Harapan

Tangerang

2021

Pembuatan Modul Praktikum Perancangan dan Pemrograman Otomasi Proses Industri Menggunakan Programmable Logic Controller dan Program Simulasi Factory I/O

Universitas Pelita Harapan http://ee.fast.uph.edu

Tangerang, 20 Juli 2021

Disusun oleh
Exsel Predinal Yohanes

Daftar Isi

Panduan Instalasi

I. Tujuan

- Mahasiswa dapat mengenal cara memprogram PLC menggunakan Do-more.
- Mahasiswa dapat melakukan instalasi Do-More dan Factory I/O.
- Mahasiswa dapat mengintegrasikan Do-More dan Factory I/O.

II. Dasar Teori

Programmable Logic Control yang biasa dikenal dengan nama PLC adalah suatu konsep yang dirancang untuk menggantikan suatu rangkaian *relay sequensial* dalam suatu sistem kontrol. Konsep PLC biasanya digunakan untuk memprogram proses otomasi pada industri. Pada modul ini kita akan menggunakan platform PLC Do-More Designer, karena Do-More Designer merupakan suatu platform PLC yang gratis dan memiliki fitur paling lengkap.

Pada modul ini *software* simulasi yang akan digunakan adalah Factory I/O. Factory I/O adalah sebuah simulasi pabrik secara 3D yang dapat membangun dan mensimulasikan sistem industri untuk mempelajari teknologi otomasi. Factory I/O dirancang agar mudah digunakan, hal ini dimaksudkan agar dengan cepat membangun pabrik virtual menggunakan *part-part* industri yang umum seperti halnya sensor dan aktuator.

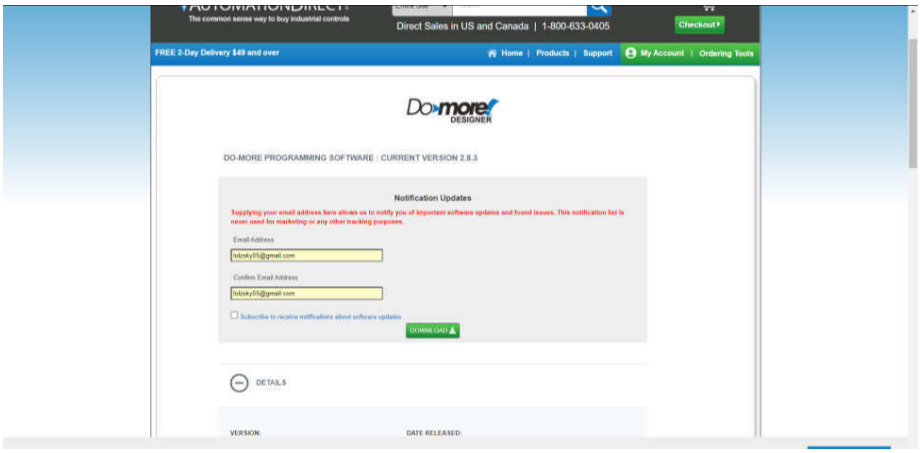
Untuk menghubungkan Do-More pada Factory I/O kita akan menggunakan jaringan Modbus. Modbus adalah metode yang digunakan untuk mentransmisikan informasi melalui jalur serial antar perangkat elektronik. Perangkat yang meminta informasi disebut Modbus Master dalam hal ini adalah Do-More dan perangkat yang memasok informasi adalah Modbus *Slaves* dalam hal ini adalah Factory I/O.

III. Peralatan

- *Software* Do-More Designer
- *Software* Factory I/O
- Komputer atau laptop
- Koneksi internet

IV. Langkah-langkah Instalasi

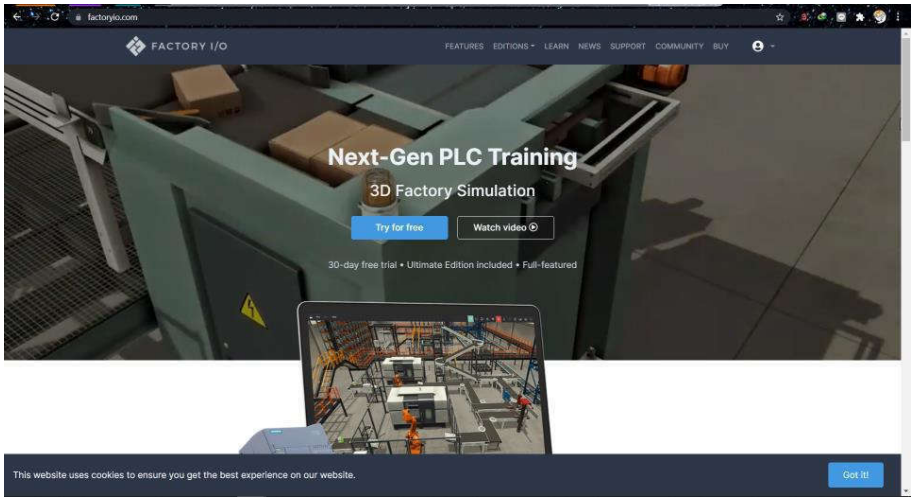
1. Instalasi Do-More Designer
   a. Pertama-tama *download* terlebih dahulu Do-More Designer dari https://www.automationdirect.com/support/*softwaredownloads*?itemcode=Do-more%20Designer, klik '*Download*' dan masukkan email. Kemudian klik '*Download*' Kembali untuk mengunduh RAR Do-More Designer



Gambar 5 *download* Do-More melalui web Automationdirect.com

   b. Setelah *file* ter-*download*, *extract* Do-More Designer yang masih berbentuk RAR.
   c. Kemudian instal Setup yang telah di-*extract* sebelumnya.
   d. Selanjutnya Do-More sudah bisa digunakan

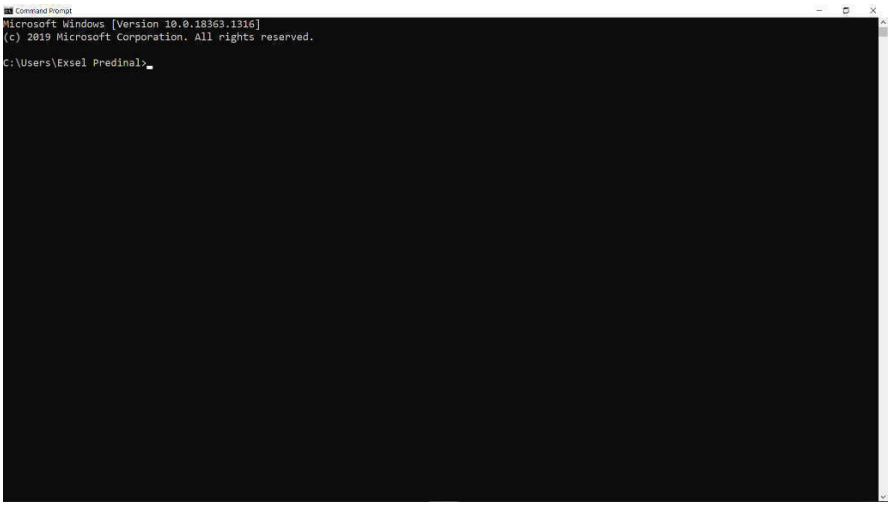
2. Instalasi Factory I/O
   a. Pertama-tama *download* terlebih dahulu Factory I/O melalui *website* https://factoryio.com.



Gambar 6 *download* Factory I/O melalui link di atas

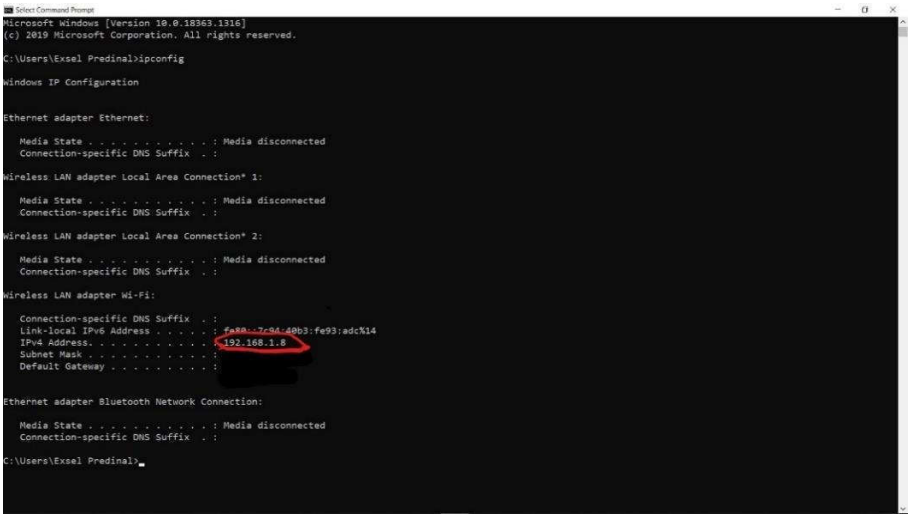
b. Setelah *file* ter-*download*, Kemudian instal setup yang telah di-*extract* sebelumnya.
c. Tunggu hingga setup selesai lalu klik *finish*.

3. Cara mengintegrasikan Do-More dengan Factory I/O menggunakan Modbus
   a. Integrasikan Do-More dan Factory I/O dengan cara buka Factory I/O yang telah di instal.
   b. Kemudian klik open > pilih skema yang ingin di gunakan > kemudian klik *file* > Driver (*shortcut*-nya adalah F4).
   c. Kemudian ketika jendela driver telah muncul klik *drop down* driver dan pilih "Modbus TCP/IP Client" setelahnya *pin-pin* pada Factory I/O sudah otomatis ter-*attach* atau jika tidak otomatis ter-*attach* maka bisa dipasang manual dengan men-*drag part-part* pada *pin* yang telah tersedia jika *pin* yang tersedia kurang *pin* bisa ditambahkan pada menu *configuration*.
   d. Lalu klik *configuration* > klik auto *connect* di atas *host* dan pastikan IP *host* Factory I/O telah sesuai dengan IP pada koneksi internet.
   e. Untuk memastikan IP-nya sama buka Windows > ketik CMD > dan *enter*.



Gambar 7 tampilan CMD

f. Setelah muncul CMD seperti pada gambar 29, ketikan Ipconfig > dan kemudian *enter*.



Gambar 8 tampilan IP pada CMD

g. Ketika telah muncul seluruh tulisan pada CMD kita telah dapat melihat IP pada IPv4 *Address* seperti yang terlihat pada lingkaran merah pada gambar 30 di atas

h. Selanjutnya buka Do-More yang telah diinstal sebelumnya, saat Do-More telah dibuka maka secara otomatis akan muncul jendela *select Project*.

i. kemudian klik “New *Offline Project*..” dan beri nama untuk *Project* Anda. Untuk Hardware *class* silakan pilih Do-More Simulator.

j. Setelah itu Buka *dashboard*, pada bagian *communication* dan *enable* kan Modbus/TCP (Modbus/TCP adalah suatu varian modbus yang ditujukan untuk pengawasan dan kontrol peralatan *otomasi* menggunakan protokol TCP/IP).

k. Setelah melakukan seluruh *setting* di atas Do-More dan Factory I/O telah siap digunakan. untuk meng-*compile* program Do-More dengan meng-klik Do-More/Sim dan untuk memulai Factory I/O dapat menekan tombol *play* seperti pada gambar di bawah.

Gambar 9 tampilan *play* pada Factory I/O

Cara penggunaan *software*

I. Tujuan

- Mahasiswa dapat mengenal cara memprogram PLC menggunakan Do-More.
- Mahasiswa dapat mengoperasikan *software* Do-More dan Factory I/O.

II. Peralatan

- *Software* Do-More Designer
- *Software* Factory I/O
- Komputer atau laptop
- Koneksi internet

III. Dasar Teori

Pada modul sebelumnya kita telah mempelajari cara instalasi dan bagaimana cara mengintegrasikan Do-More dan Factory I/O menggunakan modbus. Maka pada modul ini kita akan mengetahui bagaimana cara menggunakan Do-More dan Factory I/O. Pada modul ini mahasiswa akan mengetahui cara memasang dan menggunakan seluruh instruksi dan address pada Do-More maupun Factory I/O.

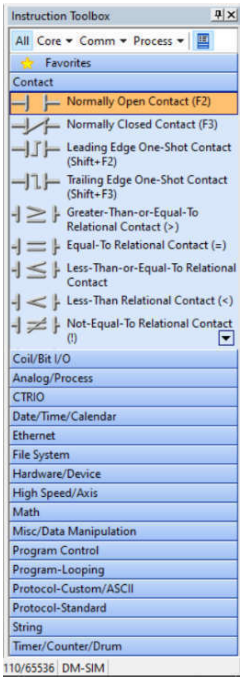
IV. Cara Penggunaan *Software*

1. Pemasangan instruksi pada Do-More Designer
   a. Cara memasang instruksi sebenarnya cukup sederhana. Tetapi pertama-tama kita harus membuat *project* baru pada *software* Do-More dengan cara menekan tab New, kemudian memilih pilihan DM Sim, disini kita juga perlu memasukkan nama *file*-nya, kemudian klik OK.

b. Lalu untuk memunculkan menu instruksi kita hanya perlu menekan tombol edit mode agar kita dapat mengedit program kita. Tombol edit mode berada pada menu yang ada di atas, untuk lebih jelas dapat dilihat kotak merah pada Gambar 1.

Gambar 10 Tampilan letak tombol edit mode

c. Kemudian setelah tombol edit mode ditekan, akan tampil menu seperti pada Gambar 2. pada menu tersebut memiliki banyak tab yang jika dibuka akan terdapat banyak instruksi sesuai dengan kategori yang tertera pada tab-tab kecil di sana.



Gambar 11 Tampilan menu tab instruksi

d. Di sini kita dapat melakukan *Drag* instruksi yang ada pada menu tersebut dan kemudian men-drop instruksi tersebut pada rung tergantung yang kita *drag* adalah *input* atau *output*.

e. Kita dapat melakukan klik sebanyak 2x pada instruksi tersebut untuk memberikan *address* ataupun informasi yang dibutuhkan.

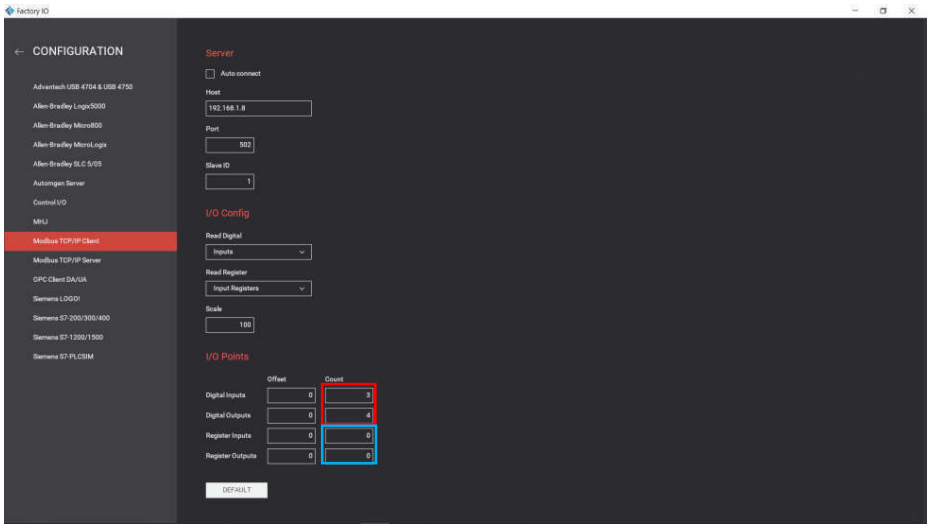
2. Pemasangan *input dan output* pada Factory I/O
   a. Pertama-tama kita dapat memunculkan skema pada menu *scenes* -> kemudian pilih sekam yang akan kita otomasikan.
   b. Kemudian untuk mengetahui *input* ataupun *output* pada skema tersebut tekan menu *file* yang ada di pojok kiri, kemudian klik Drives. Atau bisa juga dengan menekan *shortcut* tombolnya yaitu F4.
   c. Setelah menu driver muncul, tekan menu *drop down* yang ada di pojok kiri. Untuk lebih jelas menu *drop down* tersebut dikotaki warna merah di sebelah pojok kiri atas pada Gambar 3.

Gambar 12 Tampilan menu driver pada Factory I/O

   d. Setelah menu tersebut terbuka, pilih menu Modbus TCP/IP Client.
   e. Kemudian setelah tampilan menu *input dan output* pada driver tersebut muncul, kita dapat menambahkan atau memindahkan *pin input* dan *output* di sana dengan cara *drag pin* tersebut dan *drop* di tempat tujuan.
   f. Jika dirasa pin kurang banyak atau jika kita membutuhkan pin lebih. Kita dapat menambahkannya dengan cara menekan menu *configuration* di pojok kanan atas. Untuk lebih jelas menu tersebut di kotaki berwarna merah pada Gambar 4

Gambar 13 Tampilan driver setelah menu *input* dan *output* muncul

g. Setelah menu *configuration* muncul seperti Gambar 5, kita dapat mengubah banyaknya pin output dan input digital Factory I/O dari menu *count input* maupun *output* pada kotak merah di Gambar 5. Dan jika kita ingin menambahkan jumlah *pin input* maupun *output* analog kita dapat mengubahnya pada menu *register inputs* maupun *outputs* pada kotak biru di Gambar 5.



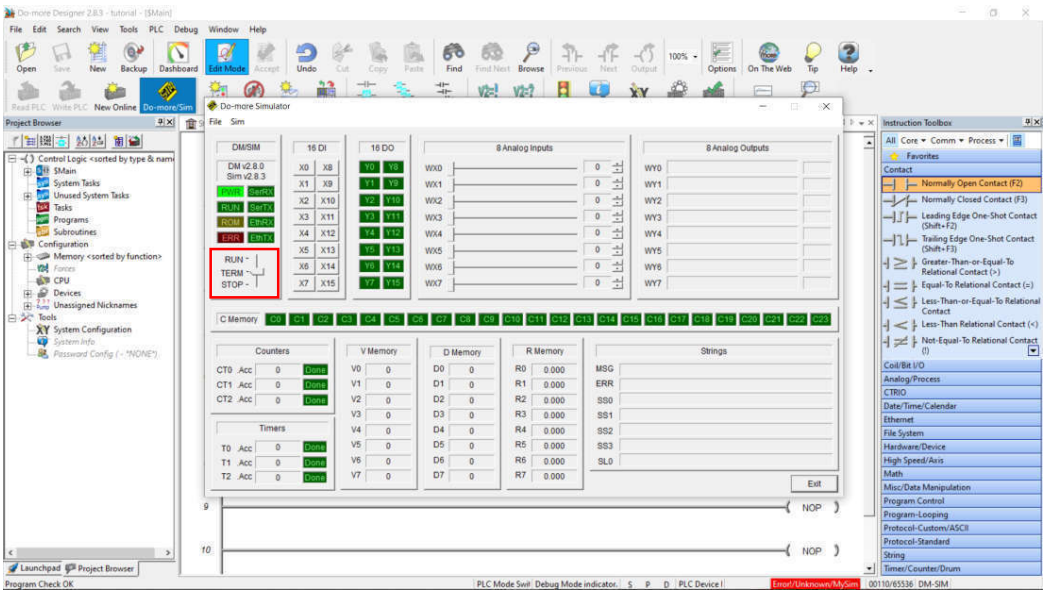
Gambar 14 tampilan menu configuration pada Factory I/O

3. Cara meng-*compile* program Factory I/O dan Do-More

   a. Pada Do-More cara *compile*-nya adalah dengan menekan tombol Do-More/Sim pada menu yang ada di sebelah atas. Lebih jelasnya tombol terdapat pada kotak merah di Gambar 6

Gambar 15 tampilan tombol Do-More/Sim pada menu

   b. Kemudian setelah muncul jendela baru naikkan tuas menjadi tombol RUN. Tuas tersebut berada pada kotak berwarna merah yang ada pada Gambar 7.



Gambar 16 tampilan *window* setelah tombol Do-More/Sim ditekan

   c. Setelah itu kita juga perlu mengaktifkan skema pada Factory I/O dengan cara masuk pada menu driver, kemudian klik tombol *connect* yang berada di sebelah pojok kanan atas. Lebih jelasnya dapat dilihat pada kotak merah di Gambar 8.

Gambar 17 tampilan tombol *connect* berada

d. Setelah itu program sudah siap dijalankan dan kita hanya perlu keluar dari menu driver dan menekan tombol *play* yang ada di pojok kanan atas tepatnya terdapat pada kotak merah yang ada pada Gambar 9.

Gambar 18 tampilan tombol *play* pada Factory I/O

Pengenalan Pada PLC

V. Tujuan

- Mahasiswa dapat mengenal cara memprogram PLC menggunakan Do-more.
- Mahasiswa dapat menggunakan instruksi load dan load not pada konsep PLC.
- Mahasiswa dapat membuat program PLC sederhana.

VI. Peralatan

- *Software* Do-More Designer
- *Software* Factory I/O
- Komputer atau laptop
- Koneksi internet

VII. Dasar Teori

Do-more Designer adalah sebuah *software* pemrograman PLC berfitur lengkap untuk rangkaian *Programmable Logic Controller* (PLC). Do-more memiliki manajemen program yang fleksibel dan mendukung perpaduan antara *stage* dan *ladder logic* untuk pendekatan terbaik dari kedua bagian yang menyederhanakan pemrograman dan mempermudah pada saat *troubleshooting*. Yang dimaksud *stage* di sini sebenarnya adalah istilah lain dari sebuah *State* diagram, step program, STL program, ataupun juga SFC program. *State* diagram ini adalah suatu paradigma pemrograman di mana mesin hanya berada pada salah satu kondisi yang berbeda pada waktu tertentu. Lalu selanjutnya adalah *ladder logic* atau *rung* atau juga sering disebut *ladder* diagram adalah garis penghubung antara instruksi *input* dan *output*.

Gambar 19 tampilan *Rung* atau *ladder logic*

Pada Do-More memiliki *address input* dan *output* digital yang dilambangkan dengan variabel X untuk *input* dan Y untuk *output* dengan masing-masing memiliki total nomor alamat 2048 buah yang bernomor 0 hingga 2047. Pada Do-More *Input* terletak di sebelah kiri dan *Output* terletak di sebelah kanan seperti pada gambar di bawah.

Gambar 20 contoh *Input* dan *output* digital pada Do-More

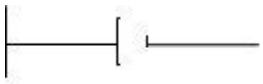
Selain *address input* dan *output* digital Do-More juga memiliki *input* dan *output* analog yang dilambangkan dengan variabel WX untuk *input* dan YX untuk *output* dengan total alamat 256 buah yang bernomor 0 hingga 255. Namun pada Do-More penggunaan *Input* dan *output* analog berbeda dengan digital. Pada *Input* dan *Output* analog membutuhkan instruksi tambahan seperti *Move*.

Dapat dilihat pada gambar di bawah ini cara penggunaan dari *input* dan *output* analog adalah menggunakan instruksi ST1 pada bagian *input*-nya. ST1 ini adalah *input* yang akan selalu *High* atau aktif saat program dimulai dan membuat *Move* ini akan selalu aktif, instruksi ini dapat diganti dengan instruksi lain sesuai kebutuhan. Lalu pada *input* dan *output* analog, *input* dan *output*-nya berada di dalam instruksi *Move* seperti yang terlihat pada contoh di bawah.

Gambar 21 contoh *Input* dan *output* analog pada Do-More

Pada Do-More ini juga memiliki 5 instruksi dasar yang biasa digunakan seperti *load*, *load not*, *counter*, *timer*, dan *compare*. Tetapi pada modul pertama ini kita hanya akan menggunakan *load* dan *load not* saja dan bagaimana cara menghubungkannya pada *software* simulator Factory I/O.

1. ***Load* atau *normally open*** adalah sebuah perintah *input* yang digunakan jika kita menginginkan suatu sistem bekerja pada kondisi logika *High*.



Gambar 22 Simbol instruksi *load*

2. ***Load not* atau *normally close*** adalah sebuah perintah *input* yang digunakan jika kita menginginkan suatu sistem bekerja pada kondisi logika *Low*.

Gambar 23 Simbol instruksi *load not*

Pada modul ini *software* simulasi yang akan digunakan adalah Factory I/O. Pada Factory I/O memiliki banyak *pin* yang dapat dimodifikasi jumlahnya pada menu *configuration* seperti yang dapat dilihat pada gambar di bawah dengan 4 jenis *pin* seperti *Coil* sebagai *output*, *Input* sebagai *input*, *Input* Register sebagai *Input* analog, dan *Holding* register yang biasa digunakan sebagai *output* analog tetapi sebenarnya *Holding* Register dapat digunakan sebagai *input* maupun *output* pada analog.

Gambar 24 tampak *Pin* pada drive simulasi Factory I/O

Untuk menghubungkan Do-More pada Factory I/O kita akan menggunakan jaringan Modbus. Untuk menggunakan Modbus pada Do-More, *address* atau alamat sumber yang digunakan untuk *input* digital adalah MI sedangkan *address* atau alamat yang digunakan sebagai *output* digital adalah MC. MI adalah singkatan dari Modbus *Input* yang cara menggunakannya sama dengan variabel Y pada Do-More tetapi fungsinya berbeda karena MI digunakan sebagai *input* untuk Factory I/O atau *output* dari Do-More sedangkan untuk MC adalah singkatan dari Modbus *Coil* dengan cara menggunakannya sama dengan variabel X pada Do-More tetapi fungsinya juga berbeda karena MC digunakan sebagai *Output* dari Factory I/O atau *input* yang akan masuk pada Do-More.

Gambar 25 contoh *input* dan *output* Digital Modbus pada Do-More

Sedangkan instruksi untuk penggunaan *input* dan *output* analog adalah MIR atau Modbus *Input* Register sebagai *input* pada Factory I/O maupun *output* dari Do-More dengan cara penggunaannya sama dengan variabel WY pada Do-More dan MHR atau Modbus *Holding* Register yang biasa digunakan sebagai *output* analog dari *software* Factory I/O dan *input* analog pada Do-More dengan cara penggunaannya sama dengan variabel WX, tetapi sebenarnya modbus *Holding* register dapat menjadi *output* maupun *input* pada Factory I/O maupun Do-More.

Gambar 26 contoh *input* dan *output* analog pada Do-More

VIII. Tugas

Gambar 27 tampilan sekilas skema pada modul ini

Download-lah skema Factory I/O dari link Google drive berikut: https://drive.google.com/u/3/uc?id=1fWKIRPRdBCLBsQ6h43MNW0GVDUxFnfd0&export=download. Kemudian buatlah agar masing-masing tombol dapat menyala ketika tombol telah ditekan dan mati saat ditekan kembali. Lalu buatlah agar ketika potensiometer diputar LCD dapat menunjukkan angka sesuai pergerakan potensiometer. **(petunjuk: gunakan *input output* digital untuk menyalakan tombol dan analog untuk potensiometer dan LCD yang telah dijelaskan sebelumnya).**

Pelatihan Dasar

I. Tujuan

- Mahasiswa dapat membuat program PLC sederhana
- Mahasiswa dapat mengontrol konveyor pada konsep PLC
- Mahasiswa dapat menggunakan instruksi *counter* pada konsep PLC

II. Peralatan

- *Software* Do-More Designer
- *Software* Factory I/O
- Komputer atau laptop
- Koneksi internet

III. Dasar Teori

Pada modul sebelumnya kita telah mempelajari cara menggunakan *Input & output* digital dan analog. Maka pada modul ini kita akan mempelajari cara menggunakan sensor dan mesin pada industri seperti konveyor dan juga kita akan mempelajari cara menggunakan instruksi ketiga yaitu *counter*. Sebenarnya *counter* dan *timer* memiliki beberapa kesamaan dan perbedaan seperti halnya nomor alamat *timer* dan *counter* pada PLC Do-More berjumlah 266 buah yang bernomor 0 sampai dengan 255. Alamat *timer* dan *counter* pada Do-More dilambangkan dengan *Code* alamat yang berbeda dengan T untuk *timer* dan CT untuk *counter*. *Timer*/*Counter* pada PLC berfungsi dengan cara saat setelah mencapai angka nol maka *contact* (T(nomor alamat).Done / CT(nomor alamat).Done) pada *timer*/*counter* akan ON.

Sebagai catatan: dalam satu program alamat nomor *counter*/*timer* tidak boleh sama dengan *counter*/*timer* yang lain. Misal, jika alamat nomor *counter*

pertama adalah CT1 maka alamat *counter* yang lain harus menggunakan alamat nomor yang berbeda seperti CT2, CT3, dst begitu pula dengan *timer*.

Counter adalah sebuah perintah *output* yang berfungsi untuk menghitung jumlah *pulse* yang masuk pada *Input*. Pada *counter* memiliki beberapa *output* yang biasanya dijadikan *Input* pada *ladder logic* lain seperti:

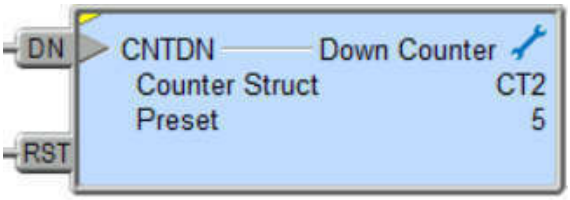
1. CT(nomor alamat).Done, contohnya CT1.Done. kontak ini akan aktif bila *timer* telah berjalan selama waktu *preset* yang telah ditentukan. Biasanya CT(nomor alamat).Done ini digunakan sebagai *Input* digital.
2. CT(nomor alamat).Acc, contohnya CT1.Acc. kontak ini biasanya dijadikan sebagai *Input* analog.

Pada Do-More memiliki 3 jenis *counter* yang paling sering dipakai yaitu:

1. Up *Counter* (CNT)

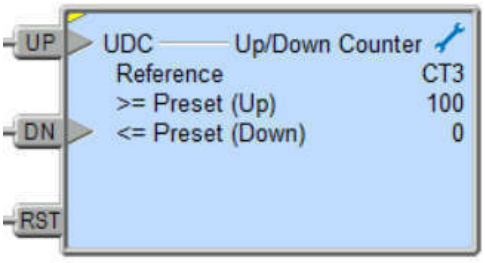
Gambar 28 Instruksi CNT

2. *Down Counter* (CNTDN)



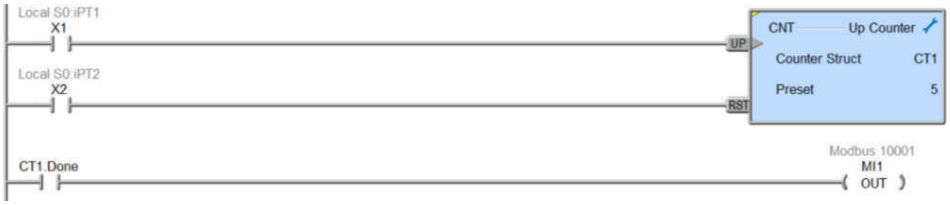
Gambar 29 Instruksi CNTDN

3. Up/*Down Counter* (UDC)



Gambar 30 Instruksi UDC

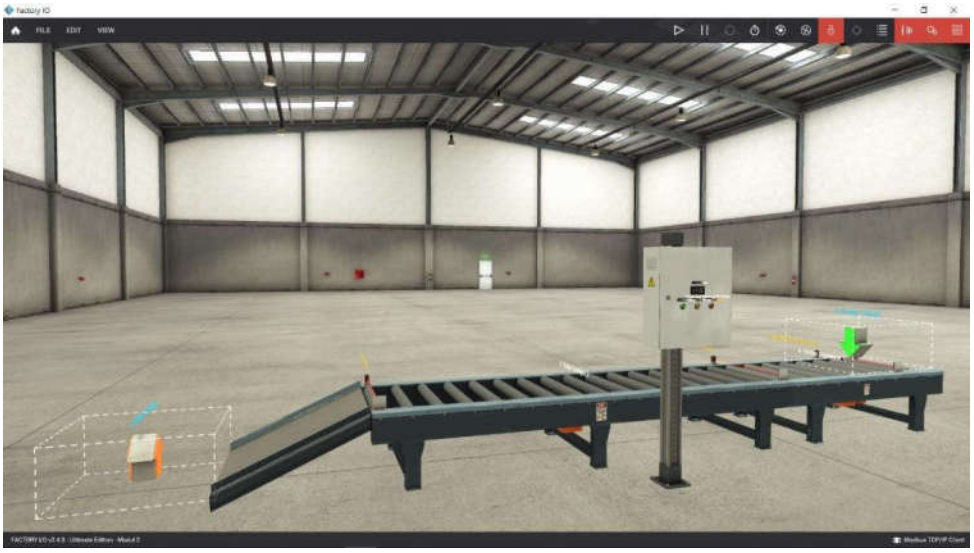
Cara kerja instruksi *counter* pada contoh di bawah ini adalah, Ketika *counter* (CT1) mendapat *Input* sebanyak dari *preset* yang telah diset sebelumnya (4095) maka *counter* akan mengaktifkan *contact* CT1.Done sehingga *output* (MI1) akan aktif, dan pada saat reset mendapatkan *Input* dari MC2 maka *counter* akan mengulang kembali.



Gambar 31 contoh penggunaan instruksi *counter*

Gambar 32 Tampilan instruksi *timer* pada saat program berjalan

IV. Tugas



Gambar 33 tampilan sekilas skema pada modul ini

Download-lah skema Factory I/O dari link Google drive berikut: https://drive.google.com/u/3/uc?id=1rNu3pCFNkkFuiyVyTlcsFsigJa7oR_o

q&export=download. Kemudian buatlah agar konveyor dapat berjalan saat tombol *start* ditekan dan berhenti saat tombol *stop* ditekan, LCD *counter* akan menghitung jumlah *Box* yang turun dan akan Kembali ke 0 ketika tombol reset ditekan, batasi agar hanya ada 3 *Box* yang berjalan pada seluruh konveyor. **(petunjuk: pada skema terdapat 3 sensor, 2 konveyor kecil dan besar, dan *emitter* akan berhenti memunculkan *Box* ketika masih terdapat *Box* di dalam area *emitter*. Gunakan lah seluruh sensor dan konveyor dengan baik dan Anda dapat menggunakan sensor yang ada dan instruksi *counter* yang telah dijelaskan sebelumnya untuk membuat LCD *counter* menghitung jumlah *Box*)**

Perancangan Sederhana

I. Tujuan

- Mahasiswa dapat membuat program PLC sederhana.
- Mahasiswa dapat menggunakan instruksi *timer* pada konsep PLC.
- Mahasiswa dapat menggunakan instruksi compare pada konsep PLC.

II. Peralatan

- *Software* Do-More Designer
- *Software* Factory I/O
- Komputer atau laptop
- Koneksi internet

III. Dasar Teori

Pada modul ini kita akan mempelajari 2 instruksi dasar yang tersisa yaitu *timer* dan compare. Pada modul sebelumnya kita sudah mengetahui perbedaan dan persamaan antara instruksi dasar *count*er dan *timer*, maka pada modul ini kita dapat langsung mempelajari bagaimana cara menggunakan instruksi dasar *Timer*.

Timer adalah sebuah perintah *output* yang digunakan untuk menentukan interval yang dihitung dari suatu kondisi atau keadaan. Pada *timer* memiliki beberapa *output* yang biasanya dijadikan *input* pada *ladder logic* lain seperti:

1. C(nomor alamat).Done, contohnya C1.Done. kontak ini akan aktif bila *timer* telah berjalan selama waktu *preset* yang telah ditentukan. Biasanya C(nomor alamat).Done ini digunakan sebagai *input* digital.
2. C(nomor alamat).Acc, contohnya C1.Acc. kontak ini biasanya dijadikan sebagai *input* analog.

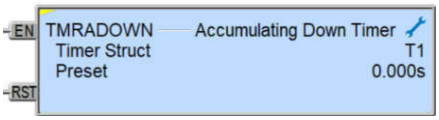
Timer yang paling sering dipakai pada PLC Do-more berjumlah 4 yang dibagi menjadi 2 sifat yaitu hitung mundur dan maju lalu pada masing-masing sifat juga memiliki 2 tipe lain.

Pada sifat hitung mundur:

1. *Down Timer* (TMR*DOWN*)

Gambar 34 Instruksi TMR*DOWN*

2. *Accumulating Down Timer* (TMRA*DOWN*)



Gambar 35 Instruksi TMRA*DOWN*

Pada Sifat hitung Maju:

3. *Up Timer* (TMR)

Gambar 36 Instruksi TMR

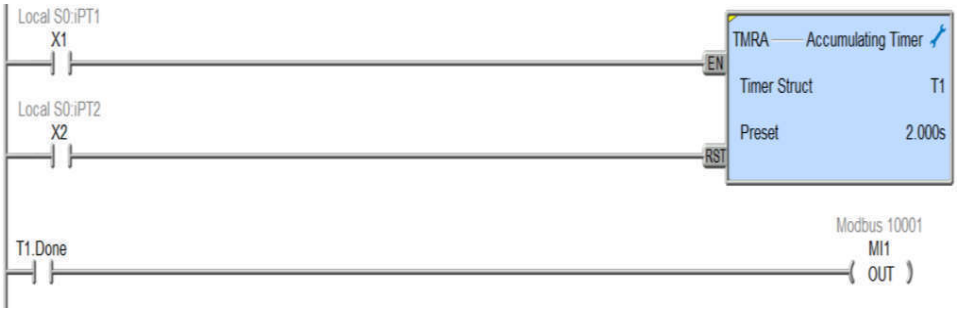
4. *Accumulating Up Timer* (TMRA)

Gambar 37 Instruksi TMRA

Perbedaan 2 tipe ini adalah seperti yang terlihat gambar, pada tipe *accumulating* instruksi memiliki 2 kaki *input ladder logic* untuk reset dan EN sedangkan pada tipe lain hanya memiliki 1 kaki *input ladder logic* dengan kaki *input ladder logic* untuk EN dan reset digabung menjadi satu.

Cara kerja dari instruksi *Timer* pada contoh di bawah ini adalah, ketika EN pada *timer* (T1) mendapatkan *input* selama *preset* yang telah ditentukan (2s) maka *timer* akan mengaktifkan *contact-contact* (T1.Done)

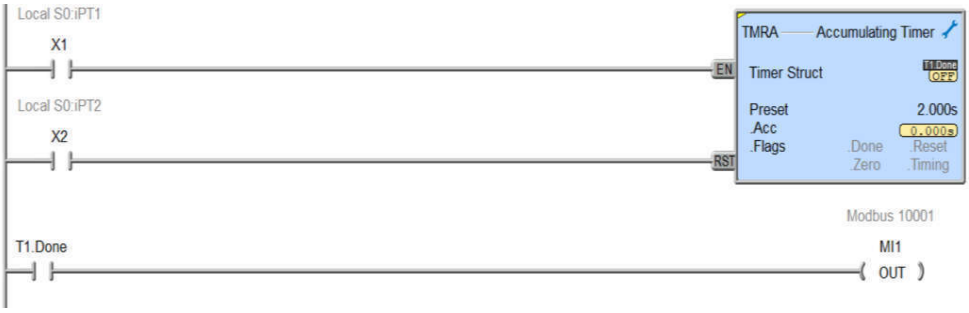
sehingga *output* (MI1) akan aktif, dan pada saat reset mendapatkan input dari MC2 maka *timer* akan mengulang kembali.



Gambar 38 contoh penggunaan instruksi *timer*

Keterangan :

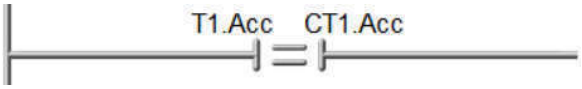
Timer akan aktif bila EN pada kondisi ON dan reset pada kondisi *OFF*. Ketika pertama kali dieksekusi *timer* akan mengukur preset dalam orde 0,1 detik.



Gambar 39 Tampilan instruksi *timer* pada saat program berjalan

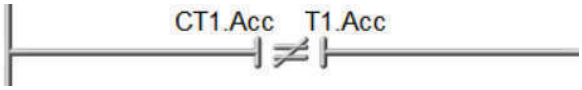
Selanjutnya kita akan mempelajari cara menggunakan instruksi *compare*. *Compare* adalah sebuah instruksi yang digunakan untuk membandingkan dua buah data. Fungsi *compare* sebenarnya memiliki fungsi yang hampir sama dengan fungsi *If Else* pada *programming language* lain yaitu ketika syarat *compare* terpenuhi maka kontak akan aktif. Pada Do-More memiliki 6 jenis instruksi *compare* yang paling sering dipakai yaitu:

1. Equal-To Relational Contact



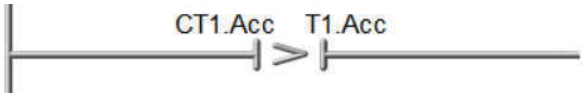
Gambar 40 Equal-To Relational Contact

2. Not-Equal-To Relational Contact



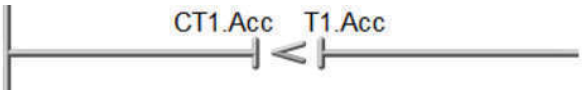
Gambar 41 Not-Equal-To Relational Contact

3. Greater-Than Relational Contact



Gambar 42 Greater-Than Relational Contact

4. Less-Than Relational Contact



Gambar 43 Less-Than Relational Contact

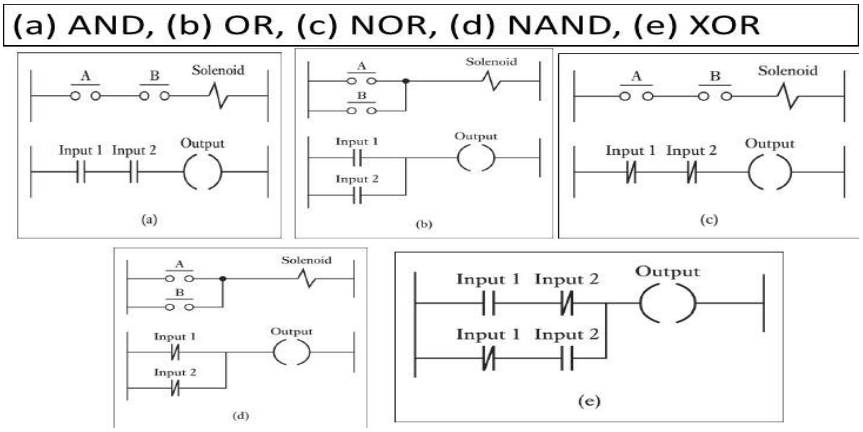
5. Greater-Than-or-Equal-To Relational Contact

Gambar 44 Greater-Than-or-Equal-To Relational Contact

6. Less-Than-or-Equal-To Relational Contact

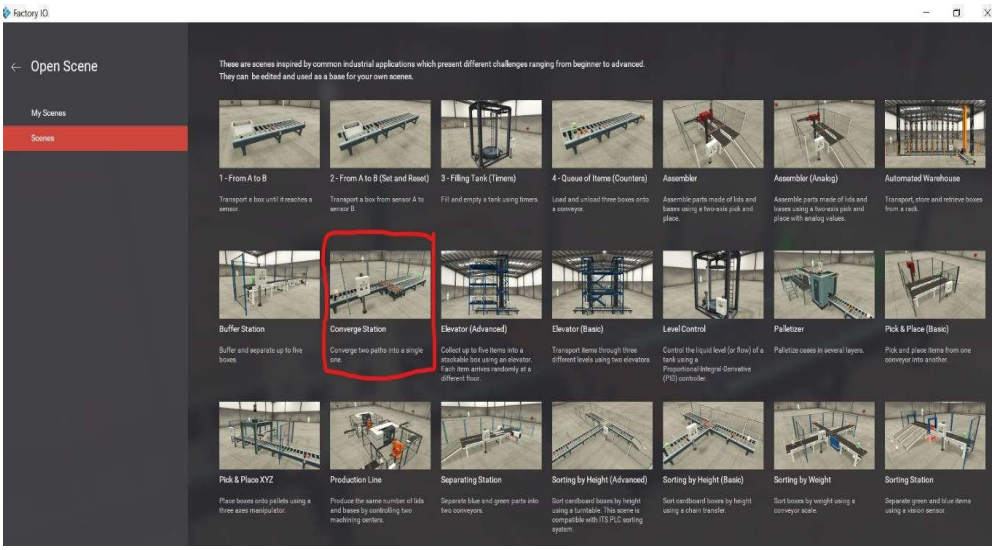
Gambar 45 Less-Than-or-Equal-To Relational Contact

Selain keenam instruksi *compare* di atas, *compare* juga dapat dilakukan pada rangkaian penempatan *ladder logic* seperti *AND, OR,* NOR*, NAND*, dan XOR. Rangkaian instruksi dapat dilihat pada gambar di bawah.



Gambar 46 Instruksi *Compare* pada *ladder logic*

IV. Tugas



Gambar 47 Tampilan skema *Converge Station* pada Factory I/O

Pilihlah skema *Converge Station* pada skema yang tersedia pada Factory I/O. Kemudian buatlah agar konveyor dapat berjalan saat tombol *start* ditekan dan berhenti saat tombol *stop* ditekan dengan keadaan konveyor hanya bisa bergerak ketika *switch* pada kondisi auto, LCD *count*er akan menghitung jumlah *Box* yang turun dan akan Kembali ke 0 ketika tombol reset ditekan, programlah agar barang dari kedua jalur agar dapat menuju *remover*, lalu sebagai tindak pengamanan buatlah agar seluruh kegiatan akan berhenti bila *emergency stop* ditekan. **(petunjuk: pada skema terdapat 2**

transfer konveyor dengan 2 sensor pada masing-masing transfer konveyor tersebut dan juga terdapat 1 sensor yang berada di akhir transfer konveyor. Gunakan lah seluruh sensor dan konveyor dengan baik dan Anda dapat menggunakan sensor yang berada di akhir transfer konveyor dengan instruksi *count*er yang telah dijelaskan sebelumnya untuk membuat LCD *count*er dapat menghitung jumlah *Box*. Gunakanlah instruksi *timer* untuk mengatur arah dan lamanya transfer konveyor bekerja dan gunakanlah instruksi *compare* untuk mengatur per*pin*dahan transfer konveyor agar barang pada kedua transfer konveyor tidak bertabrakan)

Perancangan Kompleks

I. Tujuan

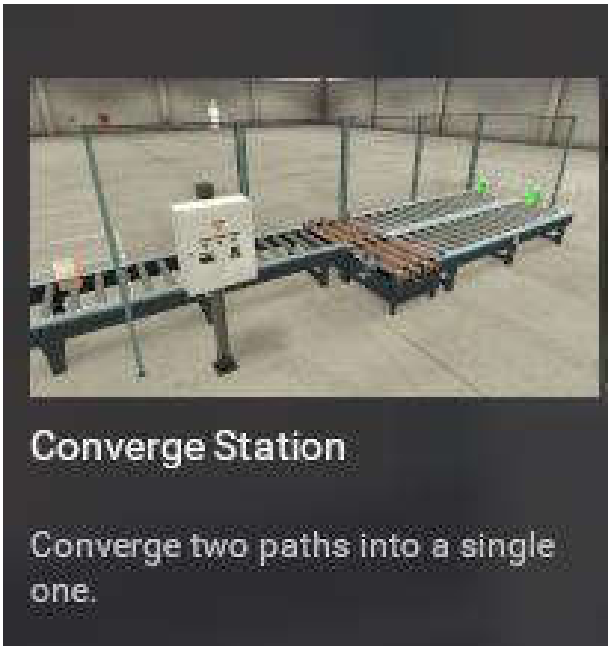
- Mahasiswa dapat membuat program PLC sederhana.
- Mahasiswa dapat memanfaatkan seluruh instruksi dasar yang telah dipelajari pada modul sebelumnya dengan baik.

II. Peralatan

- *Software* Do-More Designer
- *Software* Factory I/O
- Komputer atau laptop
- Koneksi internet

III. Dasar Teori

Pada modul sebelumnya Anda telah mempelajari cara menggunakan *address* dan ke 5 instruksi dasar PLC. Maka pada modul ini mahasiswa akan di uji pemahamannya terhadap modul yang telah dipelajari sebelumnya. Pemahaman mahasiswa akan di uji dengan cara merancang sendiri proses otomasi pada skema-skema yang telah dipilih oleh setiap orang maupun *Group* sesuai dengan deskripsi yang tertera di bawah skema tersebut menggunakan *address* dan instruksi dasar yang telah dipelajari pada modul sebelumnya.



Gambar 48 contoh tampilan skema *Converge Station* dengan deskripsinya pada Factory I/O

Mahasiswa juga dapat membuat *sequence table* dahulu sebelum langsung membuat program otomasi pada skema yang akan dipilih nantinya. *Sequence table* ini dapat berguna untuk membantu mahasiswa dalam memikirkan secara sistematis apa saja yang harus terjadi dalam sebuah skema nantinya. Saya akan memberikan sebuah contoh *sequence table* dari skema pada modul 3 yaitu *converge station*.

Input ***Output***

Step	Sensor *entry* 1	Sensor *entry* 2	Sensor transfer 1	Sensor transfer 2	Sensor *exit*	*Start button*	Reset button	*Stop* button	*Emergency stop*	*Switch* Auto
1	x	x	x	x	x	x	x	x	x	0
2	0	0	0	0	0	1	0	1	1	1
3	1	1	0	0	0	0	0	1	1	1
4	0	0	1	1	0	0	0	1	1	1
5	0	0	0	1	1	0	0	1	1	1
6	0	1	1	0	0	0	0	1	1	1
7	1	0	0	1	1	0	0	1	1	1
8	0	0	1	1	0	0	0	1	1	1
9	0	0	0	1	1	0	0	1	1	1
10	0	1	1	0	0	0	0	1	1	1
11	1	0	0	1	1	0	0	1	1	1
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	1	1	0

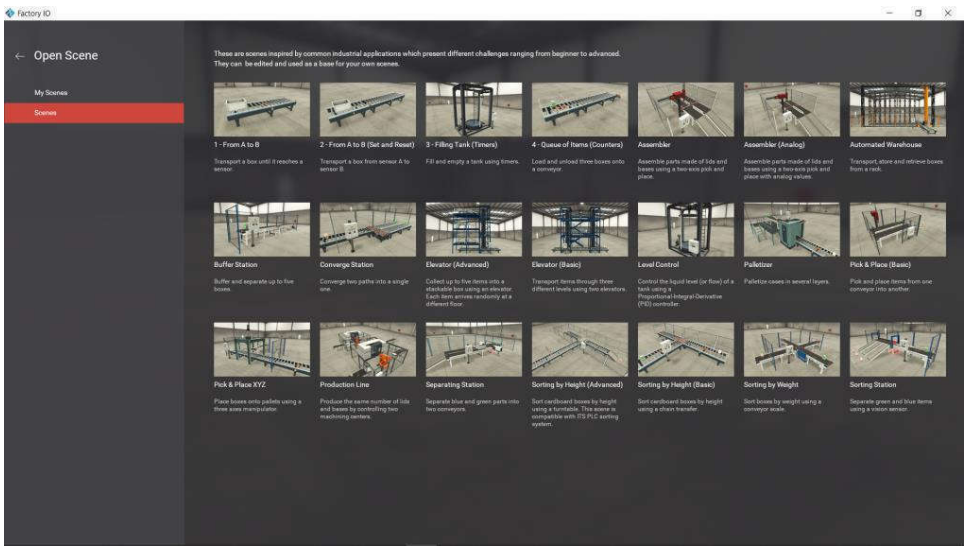
Step	Konveyor 1	Load 1	*Unload* 1	Transfer *right* 1	Transfer *left* 1	Konveyor 2	Load 2	*Unload* 2	Transfer *right* 2	Transfer *left* 2	*Exit* konveyor	*Start light*	Reset *light*	*Stop light*	*Counter*
1	0	0	X	X	0	0	0	X	X	0	0	0	0	0	0
2	1	1	X	X	0	1	1	X	X	0	1	1	0	0	0
3	1	1	X	X	0	1	1	X	X	0	1	0	0	0	0
4	0	1	X	X	0	0	1	X	X	0	1	0	0	0	0
5	0	1	X	X	1	0	1	X	X	1	1	0	0	0	1
6	0	1	X	X	0	1	1	X	X	0	1	0	0	0	0
7	1	1	X	X	0	0	1	X	X	0	1	0	0	0	1
8	0	1	X	X	0	0	1	X	X	0	1	0	0	0	0
9	0	1	X	X	1	0	1	X	X	1	1	0	0	0	1
10	0	1	X	X	0	1	1	X	X	0	1	0	0	0	0
11	1	1	X	X	0	0	1	X	X	0	1	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabel 2 contoh *sequence table converge station*

Pada *sequence table* di atas *input* dan *output* diberi angka 1 sebagai ON, 0 sebagai *OFF*, dan X sebagai *ignore* atau tidak penting. Pada step 1 menunjukkan bahwa tidak masalah *input* apa pun jika *switch* auto *off* maka seluruh sistem akan tetap mati. Lalu dapat dilihat pada *output* seluruh *unload* dan transfer *right* ditandai dengan X yang artinya kita tidak menggunakannya atau tidak penting. Pada *sequence table* tersebut ada 5 warna yang saya tambahkan untuk membantu agar tabel dapat lebih mudah dibaca yaitu hijau yang berarti barang pada konveyor 1 telah melewati sensor *exit* dan telah di

count maka barang pada transfer konveyor 2 dapat di transfer ke transfer konveyor 1 untuk dijalankan melewati sensor *exit* dan di *count*, lalu warna biru yang berarti bahwa barang pada transfer konveyor 2 telah selesai berpindah ke transfer konveyor 1 maka konveyor 2 dapat menyala kembali untuk mengirimkan barang baru kepada transfer konveyor 2, selanjutnya warna oren yang berarti bahwa barang yang ada pada transfer 1 telah melalui sensor *exit* dan telah di *count* maka konveyor 1 dapat menyala kembali untuk mengirimkan barang baru kepada transfer konveyor 1, lalu warna kuning artinya bahwa program akan me-*looping* kembali pada step sebelum warna hijau, dan yang terakhir warna merah yang artinya saat *emergency stop*, *stop button*, atau *switch* auto diputar maka seluruh *output* akan mati dan program akan berhenti saat itu juga.

IV. Tugas



Gambar 49 tampilan kumpulan skema yang akan dirancang proses otomasinya oleh para mahasiswa

Pilihlah skema yang menurut Anda mudah pada Factory I/O kecuali 1-*From* A *To* B, 2-*From* A *To* B (*Set and Reset*), 3-*Filling Tank (Timers),* 4-*Queue Of Items (Counter), Level Control, Converge Station*. Kemudian buatlah agar proses otomasi dari skema tersebut dapat berjalan saat tombol *start* ditekan dan berhenti saat tombol *stop* ditekan dengan keadaan proses otomasi hanya bisa bergerak ketika *switch* pada kondisi auto, buatlah agar proses otomasi pada alat industri di skema tersebut berjalan sesuai dengan deskripsinya, lalu sebagai tindak pengamanan buatlah agar seluruh kegiatan akan berhenti bila *emergency stop* ditekan. (**petunjuk: gunakanlah seluruh instruksi yang telah dipelajari pada modul sebelumnya untuk melakukan proses otomasi, perhatikan deskripsi dari skema tersebut, Anda dapat melihat Youtube untuk mengetahui jalan kerja skema tersebut, dan Anda juga dapat membuat *sequence table* terlebih dahulu seperti yang telah di contohkan di atas untuk mempermudah Anda dalam berpikir secara sistematis)**

Perancangan *Safety*

I. Tujuan

- Mahasiswa dapat membuat program PLC sederhana.
- Mahasiswa dapat mempelajari dan memprogram *safety* dalam konsep PLC.

II. Peralatan

- *Software* Do-More Designer
- *Software* Factory I/O
- Komputer atau laptop
- Koneksi internet

III. Dasar Teori

Akhimya kita mencapai pada modul terakhir, pada modul ini kita akan mempelajari materi terakhir yaitu *safety* PLC. Sebenarnya pada beberapa modul sebelumnya kita sudah mempelajari salah satu materi tentang *safety* PLC yaitu *emergency stop*, tetapi pada modul ini kita akan mempelajari lebih lanjut tentang *safety* PLC. *Safety* PLC adalah fungsi keselamatan terintegrasi pada konsep PLC yang memungkinkan untuk mengontrol *safety control* dan *Standard control*. Keuntungan menggunakan *safety* PLC adalah dapat meminimalisir kemungkinan terjadinya *error*, sekaligus juga dapat menghemat waktu dan uang untuk memperbaiki *part* yang rusak.

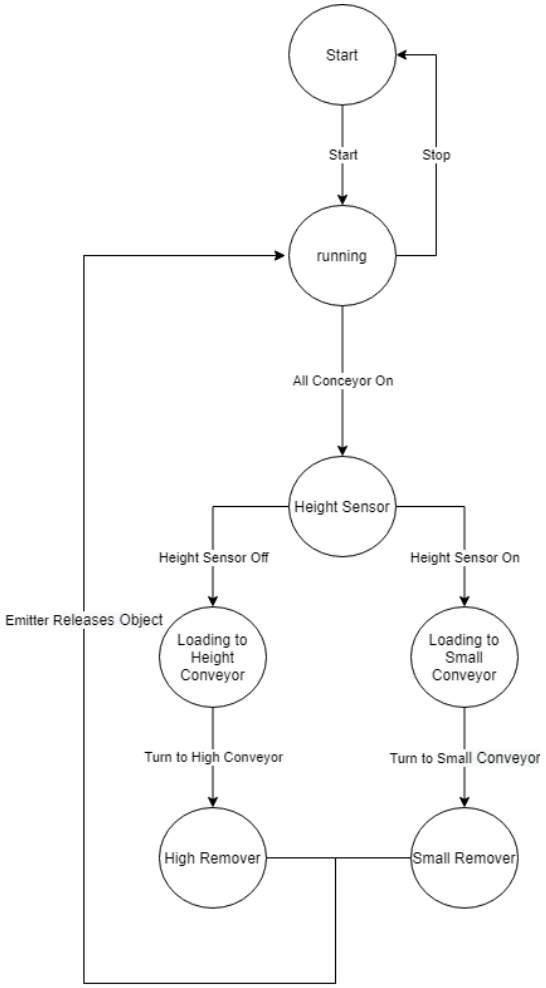
Pada modul *safety* PLC ini kita akan mempelajari materi tentang *switch* manual. Pada setiap rangkaian mesin industri atau proses otomasi industri biasanya diwajibkan agar selalu terdapat *switch* auto dan manual pada panel *Box*. Untuk auto pada *switch* tentunya kita semua sudah tahu fungsinya adalah sebagai *Standard control* untuk memulai proses otomasi, tetapi kita masih belum mengetahui apa sebenarnya fungsi dari *switch* manual tersebut. Pada panel *Box* setiap proses otomasi industri diharuskan selalu ada *switch* manual,

ini karena *switch* manual adalah sebuah *safety control* yang biasanya digunakan untuk menguji apakah terdapat bug atau *error* pada *part* mesin di setiap step dalam proses otomasi industri.

Agar lebih dapat dimengerti apa itu yang dimaksud dengan step yang ada pada proses otomasi pada industri kita akan mengingat kembali yang telah dijelaskan pada modul pertama yaitu *stage* program memiliki beberapa istilah yaitu *State* diagram, step program, STL program, dan juga SFC program. Nah di sini artinya step yang ada pada proses otomasi sebenarnya hanyalah istilah lain dari *stage* ataupun *State*. Terdapat 4 langkah mudah untuk membuat manual program pada PLC yaitu:

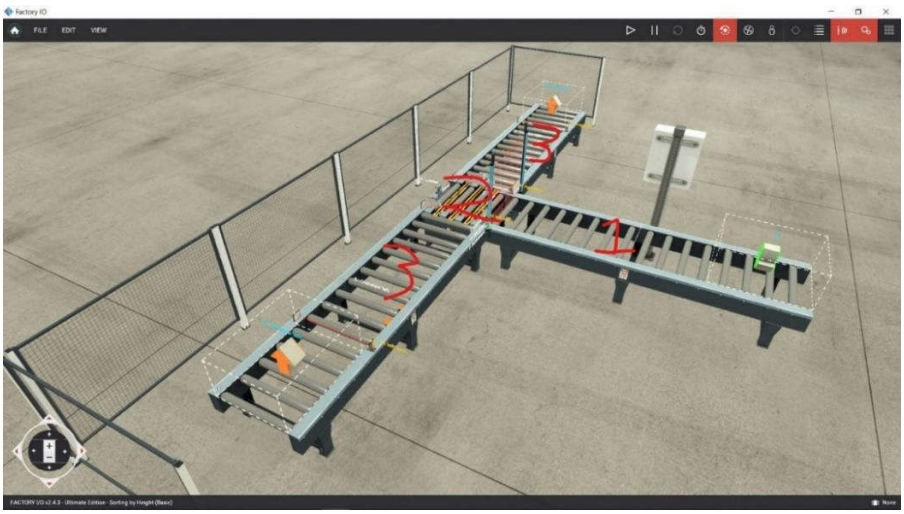
1. Deskripsikan fungsi program Anda sedetail mungkin terutama yang dapat dimengerti oleh sudut pandang orang lain.
2. Identifikasi dan beri nama semua *input* dan *output* Anda agar lebih mudah dipahami.
3. Buat daftar seluruh step yang dibutuhkan mesin, dan indikasikan di mana transisinya.
4. Buat programnya.

Di bawah ini terdapat contoh *State* diagram *sorting By height* (*Basic*) pada gambar 1.



Gambar 50 contoh *State* diagram *Sorting By Height* (*Basic*)

IV. Tugas



Gambar 51 contoh tampilan skema *Sorting By Height* (*Basic*) yang memiliki 3 step

Setelah Anda membuat program proses otomasi pada skema yang Anda pilih pada modul sebelumnya berjalan sesuai seharusnya. Maka sekarang lanjutkanlah skema tersebut dengan menambahkan agar proses otomasi dapat berjalan per-step setiap tombol *start* ditekan dan akan langsung berhenti saat tombol *stop* ditekan dengan keadaan proses ini hanya akan bergerak ketika *switch* pada kondisi manual, lalu sebagai tindak pengamanan buatlah agar seluruh kegiatan akan berhenti bila *emergency stop* ditekan. **(petunjuk: gunakanlah *count*er untuk menghitung setiap stepnya agar hanya satu step yang berjalan di setiap proses. jika Anda bingung apa itu step Anda dapat melihat contoh pada gambar 2, pada gambar tersebut contoh skema yang diambil adalah *sorting By height* (*Basic*) yang memiliki tiga step pada setiap satu proses *sorting*).**