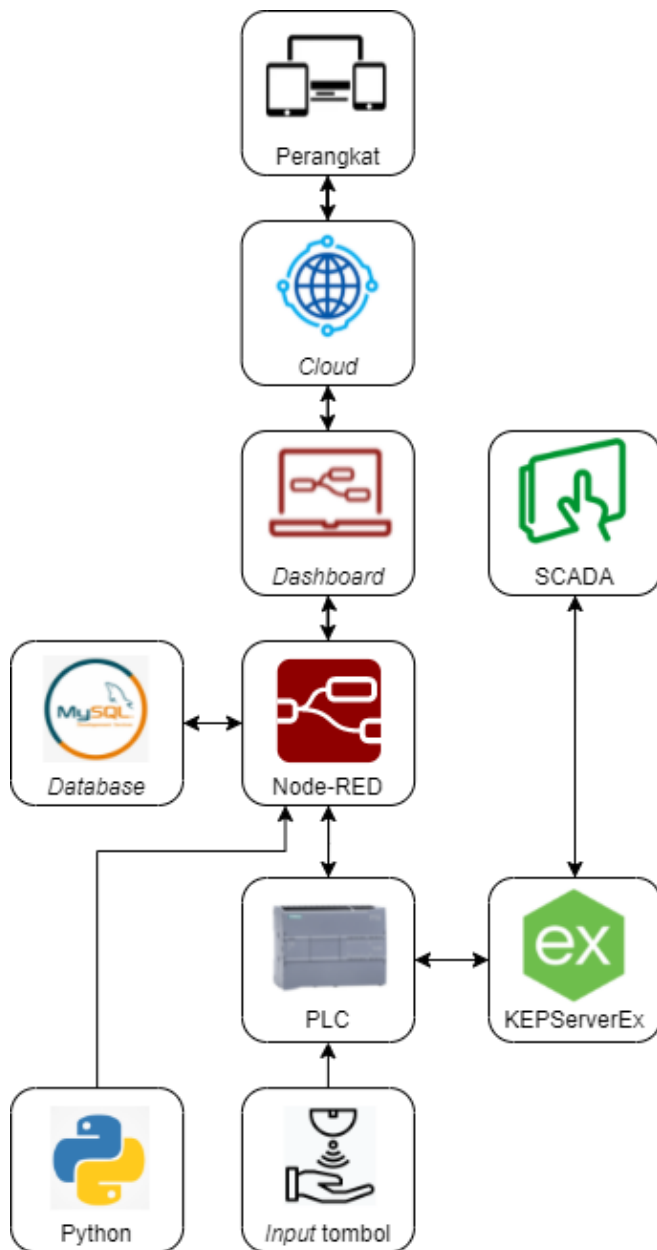


3. PERENCANAAN DAN IMPLEMENTASI SISTEM

3.1 Desain Sistem Keseluruhan

Desain sistem akan dijelaskan dalam bentuk diagram sistem pada Gambar 3.1.



Gambar 3.1 Diagram sistem keseluruhan

Berdasarkan Gambar 3.1 diatas, simulasi akan dilakukan dengan menggunakan 2 mode, yaitu mode otomatis dan mode manual. Mode otomatis akan dilakukan dengan menggunakan

program python, sedangkan mode manual akan dilakukan menggunakan input tombol dan SCADA pada PLC. Simulasi yang dilakukan pada PLC akan tampak pada tampilan SCADA dan mempengaruhi proses yang terjadi pada SCADA, hal yang sama terjadi ketika simulasi dilakukan pada SCADA akan mempengaruhi proses yang terjadi pada PLC.

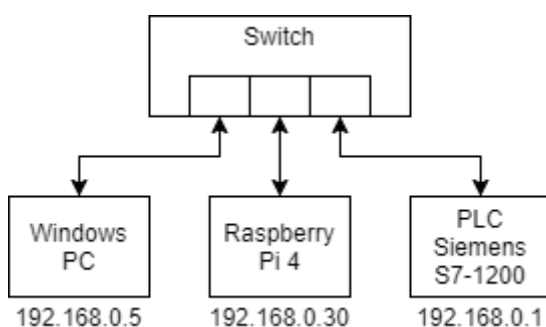
Data dari simulasi tersebut akan dikirim menuju Node-RED dan diolah untuk disimpan pada *database* maupun untuk ditampilkan pada *dashboard*. *Dashboard* juga memiliki pilihan untuk memilih data apa yang ingin ditampilkan sehingga komunikasi antara *dashboard* dan program pada Node-RED adalah komunikasi dua arah. Node-RED kemudian dihubungkan dengan *network* sehingga dapat *dashboard* dapat diakses menggunakan perangkat-perangkat seperti *handphone*, tablet dan laptop.

3.2 Desain Hardware

Pada desain *hardware* terbagi menjadi dua yaitu *wiring* pada PLC dan *wiring* pada sistem komunikasi antara perangkat yang digunakan.

3.2.1 Wiring sistem antar perangkat

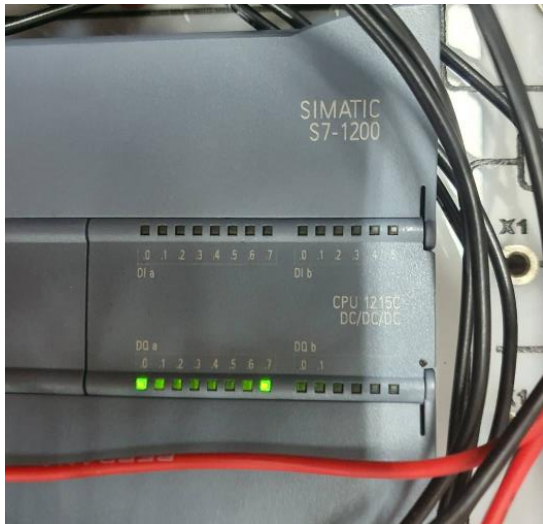
Perangkat yang perlu dihubungkan secara fisik adalah PLC Siemens S7-1200 sebagai tempat menjalankan program produksi pada simulasi mode manual, Windows PC sebagai tempat dijalankan KEServerEX dan SCADA menggunakan Wonderware Intouch dan Raspberry Pi 4 sebagai tempat menjalankan program simulasi python, program utama dan *database*. Sistem komunikasi antara perangkat dihubungkan menggunakan protokol TCP/IP ethernet dengan menggunakan *switch*. *Wiring* akan ditunjukan pada Gambar 3.2.



Gambar 3.2 Topologi *wiring* pada *hardware*

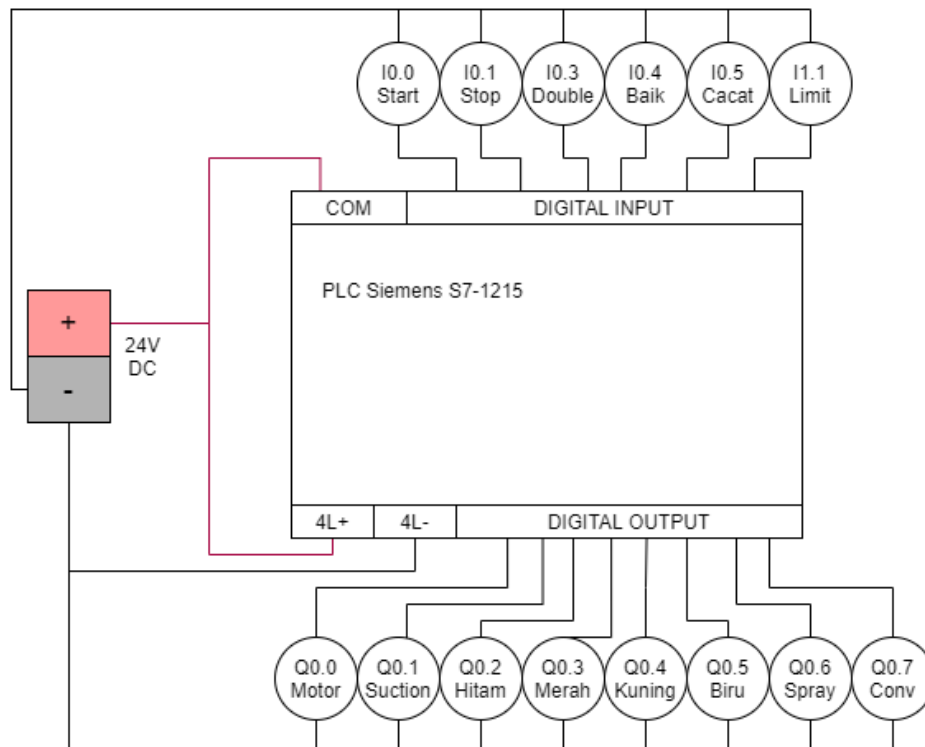
3.2.2 Wiring pada PLC

Perangkat yang digunakan untuk PLC adalah PLC Siemens Simatic S7-1200 tipe S7-1215 DC/DC/DC AG40-0XB0) yang ditunjukkan dengan Gambar 3.3.



Gambar 3.3 Gambar PLC yang digunakan

PLC akan di-wiring seperti yang ditunjukkan pada Gambar 3.4.



Gambar 3.4 Wiring pada PLC

Dari Gambar 3.4, komponen yang tersambung pada PLC berupa 6 *input* dan 8 *output*. Daftar dari *input* dan *output* akan dijelaskan di Tabel 3.1.

Tabel 3.1

Daftar *Input* dan *Output* pada PLC

<i>Digital Input</i>		<i>Digital Output</i>	
<i>Address</i>	<i>Deskripsi</i>	<i>Address</i>	<i>Deskripsi</i>
%I0.0	<i>Push button (start button)</i>	%Q0.0	Lampu LED (<i>motor</i>)
%I0.1	<i>Push button (stop button)</i>	%Q0.1	Lampu LED (<i>suction cup</i>)
%I0.3	<i>Push button (double sheet detector)</i>	%Q0.2	Lampu LED (<i>printer hitam</i>)
%I0.4	<i>Push button (sensor kamera sukses)</i>	%Q0.3	Lampu LED (<i>printer merah</i>)
%I0.5	<i>Push button (sensor kamera cacat)</i>	%Q0.4	Lampu LED (<i>printer kuning</i>)
%I1.1	<i>Switch (limit switch)</i>	%Q0.5	Lampu LED (<i>printer biru</i>)
		%Q0.6	Lampu LED (<i>powder spray</i>)
		%Q0.7	Lampu LED (<i>konveyor</i>)

Berdasarkan Tabel 3.1, digunakan 6 *digital input* yang disimulasikan menggunakan *push button* dan 8 *digital output* yang disimulasikan menggunakan lampu LED.

3.3 Desain Software

Pada bagian ini akan dijelaskan desain dan konfigurasi yang telah dibuat agar proyek dapat berjalan dengan baik. Komponen yang perlu di desain dan konfigurasi adalah PLC sebagai program simulasi manual dan komunikasi dengan komponen lainnya, Wonderware Intouch sebagai SCADA dan komunikasi dengan komponen lainnya, KEPServerEx sebagai penghubung antara SCADA dan PLC, desain program dan konfigurasi pada Node-RED, *dashboard*, *database* dan desain koneksi IoT.

3.3.1 Skenario Program Simulasi

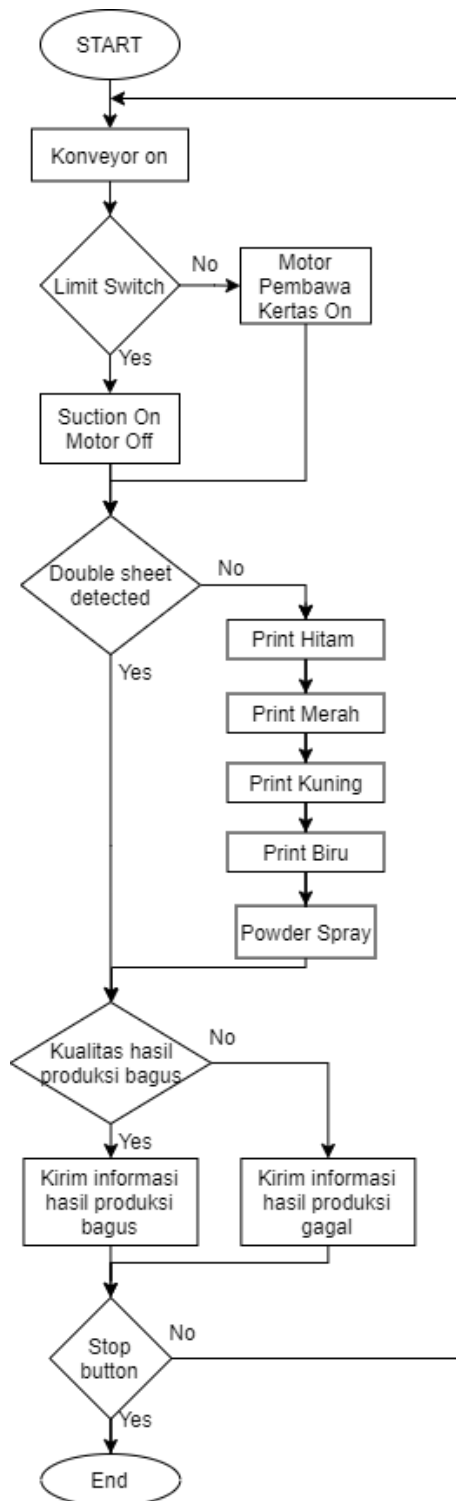
Skenario yang dibuat adalah skenario *printing* pada mesin *printer offset* Lithrone S40. Skenario produksi dibuat dengan mempelajari jurnal mahasiswa Universitas Diponegoro mengenai perhitungan OEE pada mesin percetakan (Wafa & Purwanggono, 2016) dan video mengenai cara kerja mesin *printer offset* di YouTube (Sappi Tube, 2013). Simulasi dijalankan dengan 2 mode yaitu mode manual dan otomatis.

Program simulasi manual dilakukan untuk membuktikan bahwa sistem dapat berjalan pada mesin yang sesungguhnya bila diberikan sensor maupun aktuator yang diperlukan. Namun karena pembuatan mesin bukan menjadi bagian dari tugas akhir ini, maka sensor akan digantikan tombol secara manual yang dihubungkan dengan PLC dan SCADA. Karena simulasi manual tidak memungkinkan untuk menirukan kecepatan proses mesin secara cepat, maka digunakan simulasi otomatis.

Program simulasi otomatis dilakukan untuk menirukan jalannya proses yang terjadi pada mesin yang sedang berjalan dengan kecepatan tinggi. Simulasi otomatis dapat membantu membuktikan bahwa sistem pada proyek ini dapat berjalan *real time* secara lebih konsisten dalam jangka waktu yang panjang dikarenakan setiap data didapatkan secara otomatis melalui program python. PLC dan SCADA tidak digunakan pada mode simulasi ini.

Skenario simulasi dilakukan dengan menekan tombol *start*, maka proses akan berjalan dan konveyor menyala. Sistem akan melihat apakah *limit switch* sedang tertekan atau tidak, jika tidak maka motor menyala, jika tertekan maka *suction cup* akan menyala terus menerus untuk mengambil kertas dan mengirim menuju sensor *double sheet*. Juga dilihat apakah *double sheet detector* menyala, jika ya maka sistem akan menunggu 1 detik, jika tidak maka proses berlanjut.

Proses selanjutnya adalah proses pencetakan pada mesin *printer* hitam, menuju mesin *printer* merah, menuju mesin *printer* kuning dan yang terakhir menuju mesin *printer* biru. Setelah proses *printing* selesai maka akan diteruskan menuju ke proses *powder spray* agar kertas hasil pencetakan tidak melekat 1 dengan yang lainnya, kemudian sensor kamera akan menganalisa produk yang selesai diproduksi. Kemudian sistem akan melihat apakah tombol *stop* sedang menyala atau mati. *Flowchart* pada skenario simulasi dapat dilihat pada Gambar 3.5.

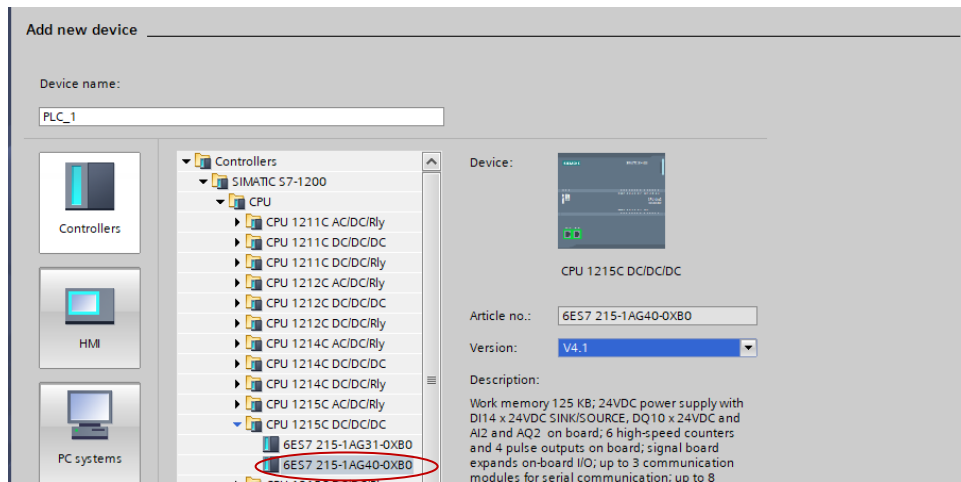


Gambar 3.5 Flowchart ladder diagram pada PLC

3.3.2 Program pada PLC

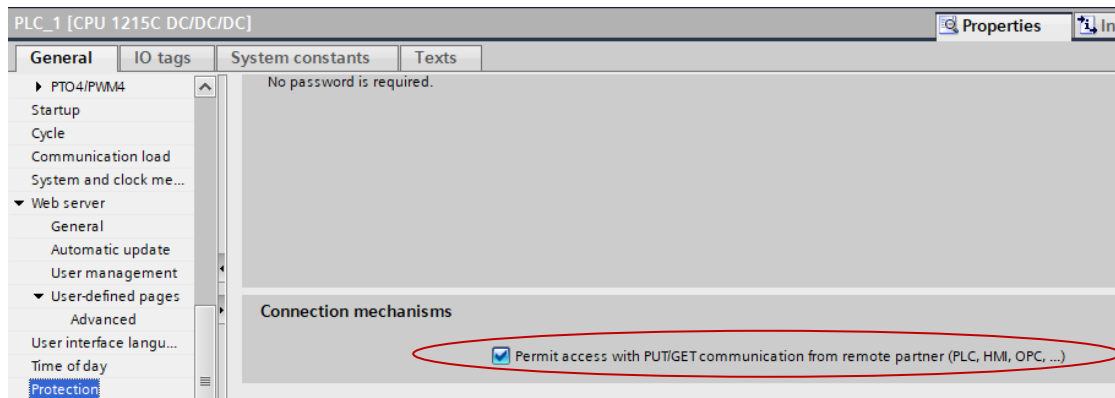
Langkah yang harus dilakukan sebelum membuat program pada PLC adalah melakukan konfigurasi pada aplikasi TIA Portal. Langkah pertama yang perlu dilakukan adalah melakukan

penambahan *device* baru dan kemudian pilih jenis PLC yang akan digunakan seperti pada Gambar 3.6.



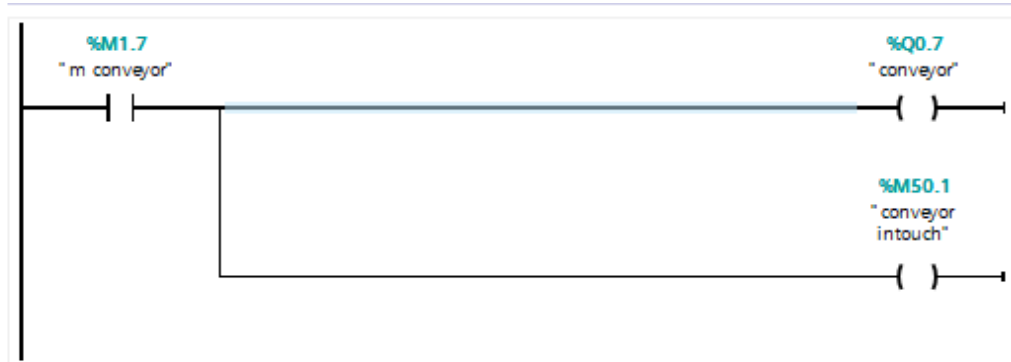
Gambar 3.6 Tampilan pemilihan jenis PLC

Kemudian mencentang opsi *Permit access with PUT/GET communication from remote partner* (PLC, HMI, OPC, ...) agar PLC dapat diakses dari perangkat lain. Tampilan dapat dilihat sesuai Gambar 3.7.



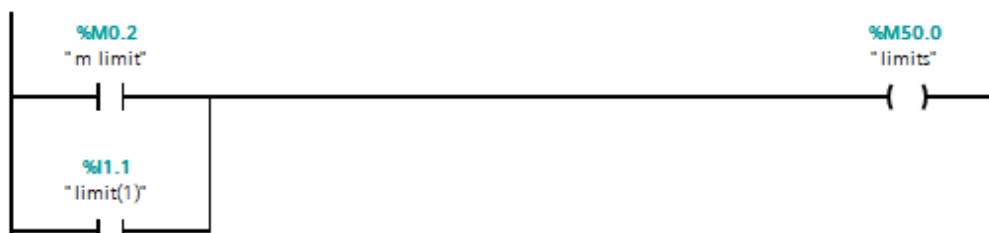
Gambar 3.7 Tampilan konfigurasi akses komunikasi PLC

Setelah konfigurasi selesai maka program dapat dilakukan Program *ladder diagram* dibuat berdasarkan *flowchart* pada Gambar 3.1. Pada *ladder diagram* *input* dan *output* tidak hanya menggunakan *input digital* dan *output digital*, melainkan juga menggunakan memori *discrete* dan memori *word* agar PLC dapat diakses melalui SCADA. Contoh program *ladder* yang menggabungkan *input digital*, *output digital* dan memori dapat dilihat pada Gambar 3.8.



▼ Network 18: input limit switch

Comment



Gambar 3.8 Tampilan *network* 17 dan *network* 18

3.3.3 Program pada SCADA

Program pada SCADA dibuat dengan tujuan untuk melakukan *monitoring* maupun *input* terhadap program pada PLC, oleh karena itu tidak ditampilkan nilai OEE. Program pada Wonderware Intouch dibuat berdasarkan *flowchart* pada Gambar 3.1. Program SCADA akan berkoneksi dengan KEPServerEx agar dapat berkoneksi dengan PLC. Koneksi yang dilakukan berupa hubungan antara *tagname* pada Wonderware Intouch dan *tagname* pada KEPServerEx. Hal pertama yang perlu dilakukan adalah membuat *Access Name* dengan konfigurasi yang sesuai dengan konfigurasi pada KEPServerEx seperti contoh pada Gambar 3.9.

Modify Access Name

Gambar 3.9 Konfigurasi *Access Name* pada Intouch

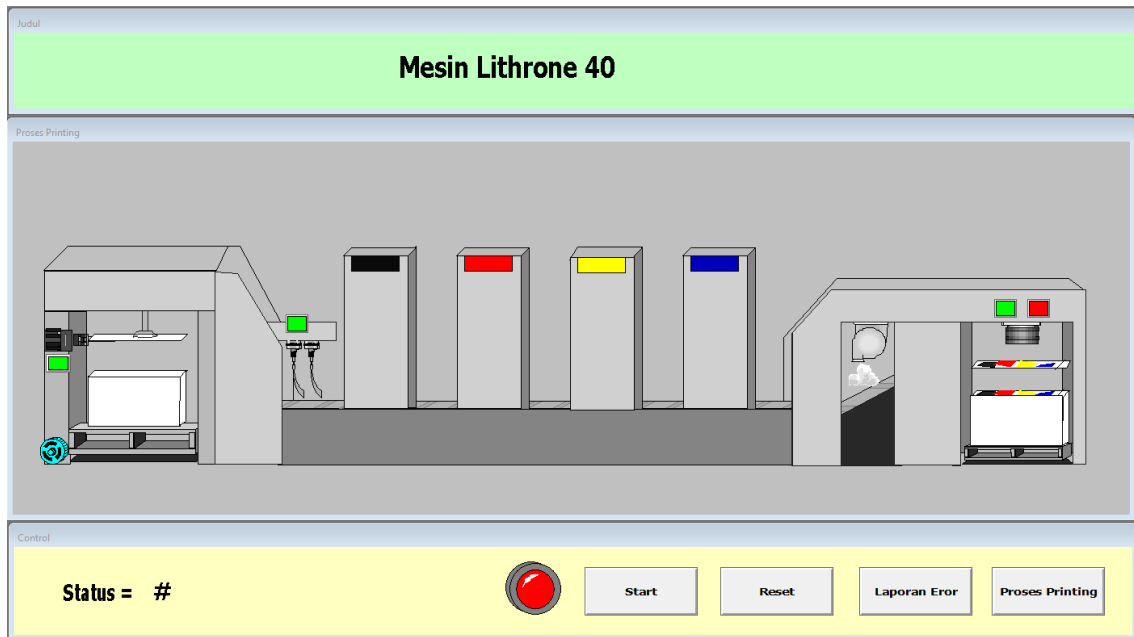
Langkah selanjutnya adalah membuat *tagname* yang akan digunakan dengan konfigurasi tipe *tagname* I/O, kemudian pilih *Access Name* yang telah dibuat. Kemudian beri *item name* sesuai dengan nama *tagname* yang akan dibuat pada KEPServerEx. Konfigurasi akan terlihat seperti pada Gambar 3.10.

Gambar 3.10 Konfigurasi *tagname* pada Intouch

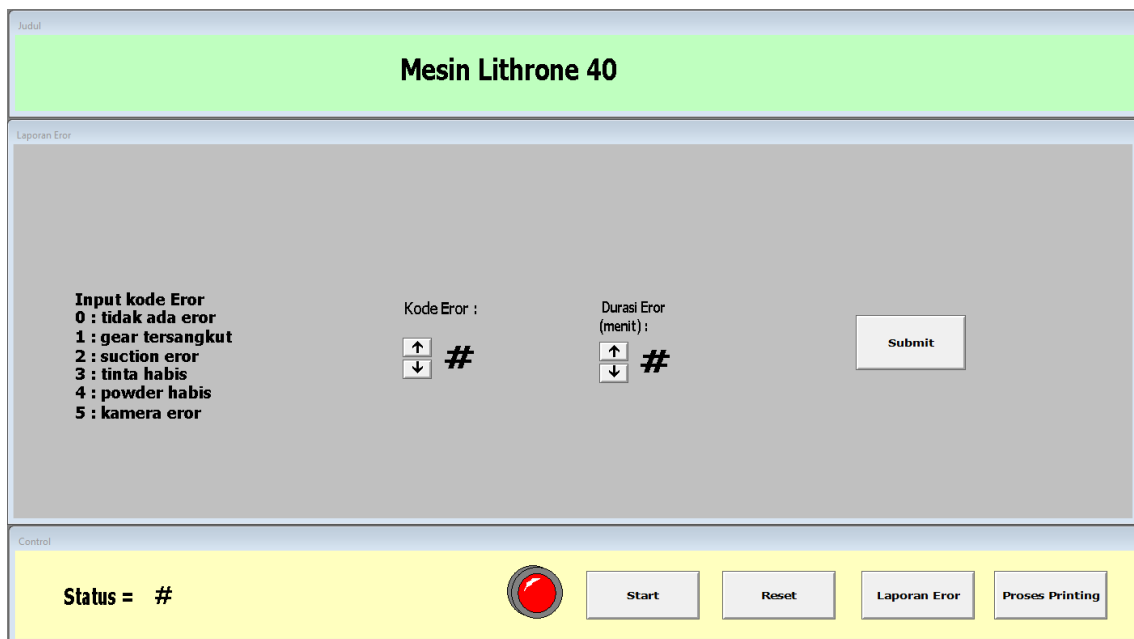
Jika konfigurasi *tagname* sudah selesai, langkah terakhir adalah melakukan save pada *tagname* tersebut. Setelah konfigurasi selesai dilakukan, maka pembuatan program pada SCADA dapat dimulai.

Pada proyek ini SCADA Wonderware Intouch digunakan sebagai media *monitoring* dan media simulasi. SCADA memiliki 4 halaman yang digunakan, yaitu halaman *Control* sebagai

tempat kontrol utama proses dan kontrol halaman pada SCADA, halaman Judul untuk menunjukkan menu yang sedang dipilih, halaman Laporan *Error* untuk mengirim laporan *error* dan halaman Proses *Printing* untuk melihat ataupun memberi *input* program. Tampilan SCADA dapat dilihat pada Gambar 3.11 dan Gambar 3.12.



Gambar 3.11 Tampilan SCADA (1)



Gambar 3.12 Tampilan SCADA (2)

Desain SCADA dibagi menjadi 2 fungsi yaitu:

a. Media Monitoring

Pada tampilan SCADA diberikan berbagai macam animasi untuk dapat memantau program *ladder* yang sedang berjalan pada PLC. Tampilan pada *monitoring* berupa animasi perubahan warna pada komponen (hijau jika menyala dan abu-abu jika mati) dan berupa animasi pergerakan pada motor, konveyor dan pada akhir proses *printing* yang ada pada halaman Proses *Printing*. Animasi terjadi ketika mendapat *input* data berupa memori dari PLC yang kemudian diolah menggunakan *script* pada masing-masing komponen ataupun melalui *application script* yang ditunjukkan pada Gambar 3.13.

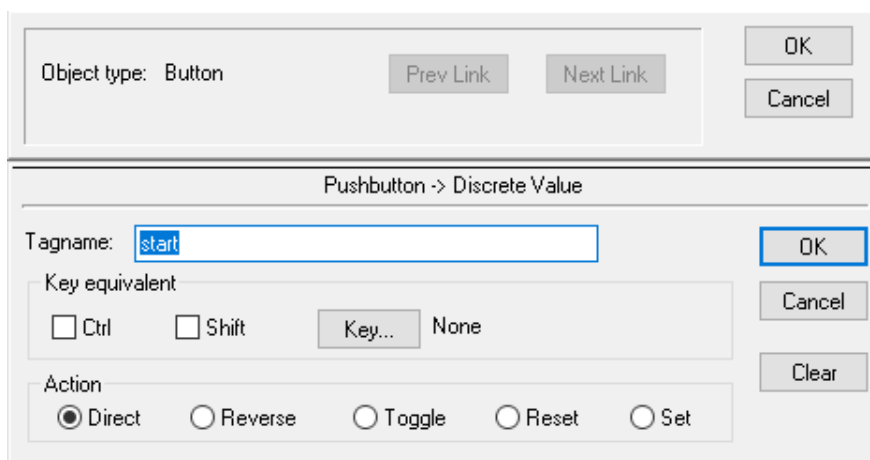
```
IF motor_intouch==1 THEN
    IF feed<20 THEN
        feed=feed+1;
    ENDIF;
ENDIF;
IF limits ==1 THEN
    IF suct<41 THEN
        suct=suct+1;
    ENDIF;
ENDIF;
IF double_intouch==1 OR y==1 THEN
    y=1;
    IF suct<160 THEN
        suct=suct+1;
    ENDIF;
ENDIF;
IF gagal_go==1 OR sukses_go==1 OR x==1 THEN
    x=1;
    IF z<11 THEN
        z=z+1;
    ENDIF;
ENDIF;
```

Gambar 3.13 *Application script* untuk animasi pada Intouch

b. Simulasi

Simulasi pada SCADA dilakukan dengan mengirim *input* menuju ke KEPServerEx dan diterima sebagai memori pada KEPServerEx dan pada PLC. Memori tersebut kemudian akan melakukan *trigger* untuk proses pada *ladder diagram* di PLC. Simulasi ini dilakukan

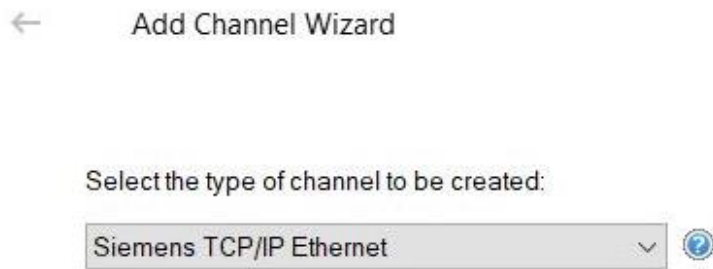
dengan cara menetapkan tombol-tombol pada tampilan SCADA untuk memberi nilai pada variabel yang telah ditentukan pada konfigurasi komponen. Tampilan halaman untuk melakukan simulasi produksi dapat dilihat pada Gambar 3.11 sedangkan tampilan halaman untuk melakukan simulasi sistem *error* dapat dilihat pada Gambar 3.12. Simulasi produksi dapat dilakukan dengan menekan tombol pada SCADA sedangkan simulasi sistem *error* dapat dilakukan dengan melakukan input kode *error* dan durasi *error* tersebut. Tampilan konfigurasi pada komponen untuk input nilai ditunjukkan pada Gambar 3.14.



Gambar 3.14 Konfigurasi komponen

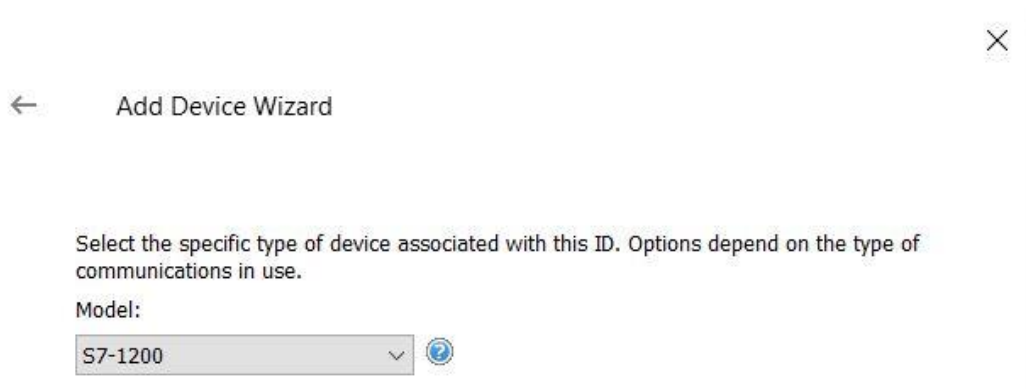
3.3.4 Konfigurasi Koneksi pada KEPServerEx

KEPServerEx berfungsi untuk menghubungkan antara SCADA Wonderware Intouch dan PLC Siemens S7-1200. Langkah pertama yang perlu dilakukan adalah membuat *Channel* yang baru dengan menyesuaikan konfigurasi pada PLC. Pastikan jenis *channel* yang dibuat sesuai dengan koneksi yang akan digunakan, dalam kasus ini adalah akan digunakan tipe "Siemens TCP/IP Ethernet" seperti yang ditunjukkan pada Gambar 3.15.

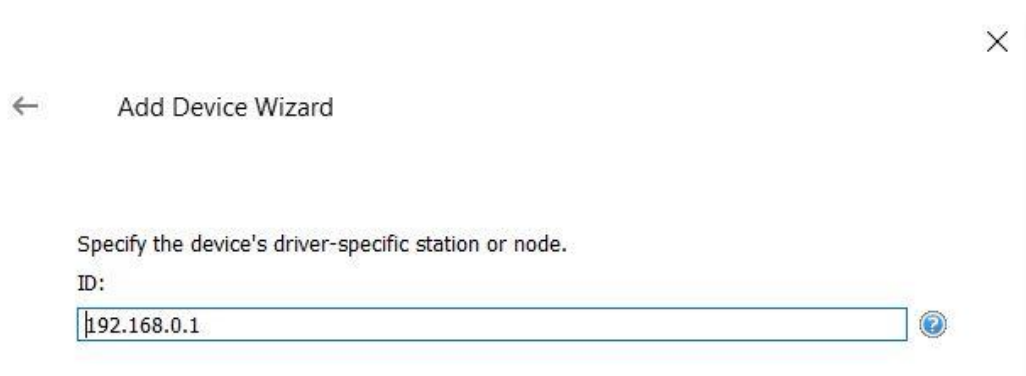


Gambar 3.15 Pemilihan tipe *channel* pada KEPServerEx

Langkah selanjutnya adalah melakukan penambahan *device* baru pada *channel* yang sudah kita buat. Pastikan jenis PLC dan IP Address PLC telah sesuai seperti yang ditunjukkan pada Gambar 3.16 dan Gambar 3.17.



Gambar 3.16 Pemilihan jenis PLC



Gambar 3.17 *Input* untuk IP Address PLC

Langkah selanjutnya adalah untuk menambah *tagname* pada *device* yang telah dibuat. Lakukan konfigurasi *tagname* sesuai dengan *item name* dan *tagname type* yang telah dikonfigurasi pada Intouch dan masukan *Address* yang sesuai dengan konfigurasi pada PLC seperti Gambar 3.18.

Identification	
Name	start
Description	
Data Properties	
Address	M0.0
Data Type	Boolean
Client Access	Read/Write
Scan Rate (ms)	100

Gambar 3.18 Konfigurasi *tagname* pada KEPServerEx

Untuk memastikan apakah koneksi sudah berjalan dengan baik, dapat dilakukan *Quick Client*. Jika baris *Value* pada *quick client* telah tampil dan baris *Quality* berisi *Good* seperti pada Gambar 3.19, maka koneksi KEPServerEx dan PLC telah berjalan dengan baik.

Item ID	Data Type	Value	Timestamp	Quality	Update
TA.PLC Siemens S7 - TA_CurrentPDUSize	Word	240	13:50:35.454	Good	2
TA.PLC Siemens S7 - TA_Rack	Byte	0	13:50:34.449	Good	1
TA.PLC Siemens S7 - TA_Slot	Byte	1	13:50:34.449	Good	1
TA.PLC Siemens S7 - TA.biru	Boolean	0	13:50:34.517	Good	1
TA.PLC Siemens S7 - TA.conveyer	Boolean	1	13:56:57.478	Good	4
TA.PLC Siemens S7 - TA.conveyor_intou...	Boolean	1	13:56:57.478	Good	4
TA.PLC Siemens S7 - TA.double	Boolean	0	13:50:34.517	Good	1
TA.PLC Siemens S7 - TA.double_intouch	Boolean	0	13:50:34.517	Good	1
TA.PLC Siemens S7 - TA.durasi_m	Word	4	14:06:02.530	Good	2

Gambar 3.19 Tampilan *quick client*

3.3.5 Program pada Python

Program pada python berisi program simulasi skenario yang diambil serupa dengan skenario pada PLC namun data hasil simulasi didapatkan secara otomatis setiap detiknya melalui program Node-RED. Proses pengambilan data dari simulasi program python dapat dilihat pada Sub Bab 3.3.5.2. Program simulasi pada program python terbagi menjadi 2 bagian, yaitu bagian mengisi nilai pada variabel yang diinginkan dan bagian pengiriman data. Pada bagian simulasi nilai variabel disimulasikan untuk menirukan nilai-nilai yang didapatkan pada saat sistem produksi berjalan secara *real time*. Simulasi akan dijalankan dengan interval 1 detik. Ada

beberapa bagian yang disimulasikan, yang pertama adalah simulasi *available time* yang bertujuan untuk menambah *available time* setiap kali simulasi dijalankan. Simulasi *planned downtime* bertujuan untuk menentukan apakah proses industri sedang mengalami *planned downtime* (setiap jam 11.00 – 12.59) atau tidak, ketika sedang mengalami *planned downtime* maka nilai *planned downtime* akan ditambah dan produksi akan diberhentikan, sedangkan jika sedang tidak mengalami *planned downtime* maka produksi akan berjalan dengan normal. Saat produksi sedang berjalan normal, angka total produksi bertambah dan akan dilakukan pengacakan angka untuk mensimulasikan 1.5% kemungkinan produk gagal diproduksi setiap detiknya, jika kemungkinan 1.5% terjadi, maka jumlah total produk gagal akan bertambah 1 dan jika tidak maka jumlah total produk gagal tidak akan bertambah. Simulasi kemungkinan *error* yang terjadi dilakukan dengan melakukan *random number generator* untuk mensimulasikan kemungkinan 0.03% terjadinya *error* setiap detiknya. Jika kemungkinan 0.03% terjadi, maka waktu *unplanned downtime* akan bertambah dan memicu simulasi jenis *error* yang terjadi. Simulasi jenis *error* yang terjadi dilakukan dengan mengacak 5 angka *random* untuk menentukan jenis *error* apa yang sedang terjadi, jika tidak terjadi *error* maka jenis *error* tidak akan diisi. Simulasi yang terakhir adalah simulasi *shift*. Simulasi *shift* berfungsi untuk menentukan *shift* yang sedang berjalan pada saat itu. Simulasi dilakukan dengan mendeteksi jam, jika jam sekarang adalah antara jam 00.00-11.59 maka statusnya adalah *shift* 1, sedangkan jika sekarang adalah antara jam 12.00-23.59 maka statusnya adalah *shift* 2. Simulasi pada penelitian ini belum memiliki fitur yang interaktif dengan pengguna dan nilai parameter yang digunakan belum bisa diubah-ubah tanpa melakukan *hard code*. Detail program python dapat dilihat pada Gambar 3.20.

```

#simulasi available time
avl=1
#simulasi planned downtime
if 11<jam<13:
    p_dt=1
    total_produksi=0
    total_gagal=0
else:
    p_dt=0
    total_produksi = 1 #28.800/hari
    a=random.randint(1,1000)
    if a<15:total_gagal=1
    else:total_gagal=0
#simulasi kemungkinan error terjadi
b=random.randint(1,79200)
if b<25:
    c=1
    up_dt=20
else:
    c=0
    up_dt=0
#simulasi jenis error yang mungkin terjadi
if c==1:
    d=random.randint(1,5)
    if d==1: eror="gear tersangkut"
    elif d==2: eror="suction eror"
    elif d==3: eror="tinta habis"
    elif d==4: eror="powder habis"
    else: eror="kamera eror"
else: eror="-"
#simulasi shift
if jam<12: shift=1
else: shift=2

```

Gambar 3.20 Program bagian pengisian variabel

Sedangkan program pengiriman data akan menggabungkan semua variabel tersebut kedalam 1 variabel bernama hasil. Variabel itulah yang nantinya akan dikirim menuju ke Node-RED untuk diolah. Detail program bagian pengiriman data dapat dilihat pada Gambar 3.21.

```

hasil =
str(total_produksi)+", "+str(total_gagal)+", "+str(shift)+", "+
str(erator)+", "+str(up_dt)+", "+str(p_dt)+", "+str(avl)
print(hasil)

```

Gambar 3.21 Program bagian pengiriman data

Pada bagian program pengiriman data, semua variabel digabung didalam menjadi 1 variabel dengan menggunakan koma sebagai pemisah antara 1 variabel dengan variabel lainnya. Variabel gabungan tersebut kemudian dimasukan kedalam perintah *print* agar dapat diambil oleh *node* pada Node-RED. Variabel yang didapatkan dari simulasi yaitu total produksi, total gagal, shift, jenis *error*, durasi *error*, *planned downtime* dan *available time*.

3.3.6 Program pada Node-RED

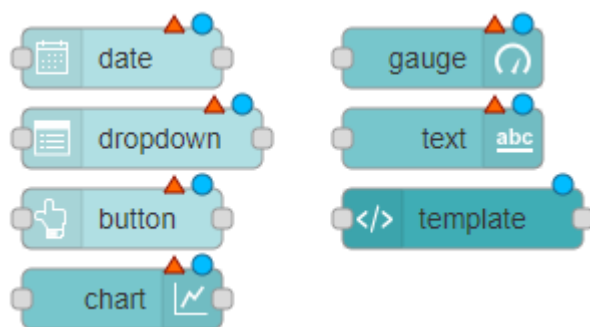
Program pada Node-RED berfungsi untuk mengambil dan mengolah data simulasi dari python dan PLC. Data kemudian disimpan dan diambil kembali sesuai perintah dan diolah sebelum kemudian ditampilkan pada *dashboard*. Proses tersebut dijalankan pada 4 *flow*. *Flow* simulasi, *flow Dashboard Real Time*, *Flow Dashboard History* dan *Flow Dashboard Tabel*. Untuk dapat menjalankan tugas program tersebut dibutuhkan *palette* atau *library* dan *node* yang butuh dikonfigurasi terlebih dahulu.

3.3.6.1 Penggunaan *Palette*

Berikut adalah daftar *node* pada *palette* yang dibutuhkan dalam proyek ini:

a. Node-red-dashboard

Palette ini berisi berbagai macam *node* yang berfungsi untuk menampilkan maupun mengambil data pada *dashboard*. *Node* yang akan digunakan adalah *node datepicker*, *dropdown*, *button* untuk melakukan *input data*, kemudian *node chart*, *gauge*, *text* dan *template* digunakan untuk menampilkan data yang telah diambil. Tampilan *node* pada *palette dashboard* yang digunakan dapat dilihat pada Gambar 3.12.



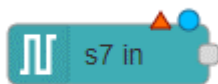
Gambar 3.22 Tampilan *node* pada *palette dashboard* yang digunakan

Node pada *palette dashboard* perlu dilakukan konfigurasi terlebih dahulu agar dapat digunakan dengan baik. Konfigurasi berupa memilih grup dan *tab* yang akan digunakan, memberi nama data yang akan diambil dan data yang akan ditampilkan, kemudian melakukan konfigurasi umum bagi masing-masing *node* seperti *range* dan *colour gradient*. Tampilan konfigurasi pada *node gauge* akan ditampilkan pada Gambar 3.23.

Gambar 3.23 Tampilan konfigurasi pada *node gauge*

b. Node-red-contrib-s7

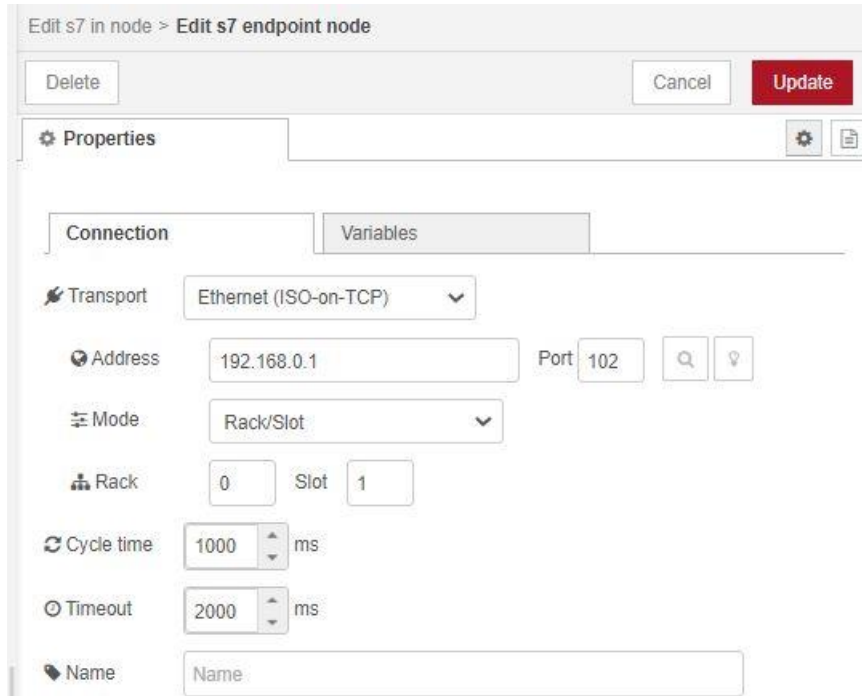
Palette ini digunakan untuk mengambil data maupun mengirim data menuju ke PLC Siemens *s7* secara *real time*. *Node* yang akan digunakan adalah *node s7 in*. *Node* ini berfungsi untuk mengambil data dari PLC secara *real time*. Tampilan *node s7 in* dapat dilihat pada Gambar 3.24.



Gambar 3.24 Tampilan *node s7 in*

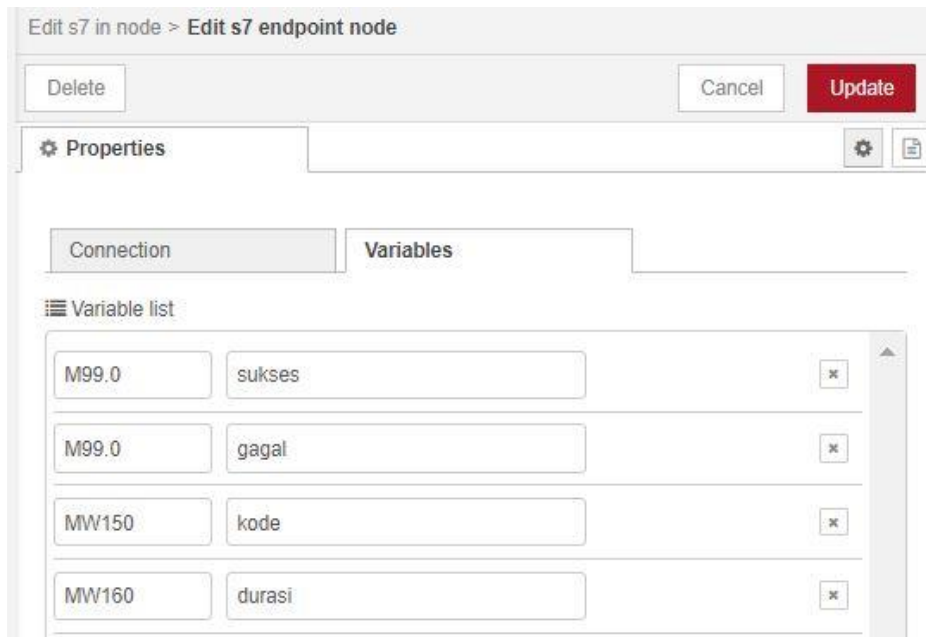
Node s7 in membutuhkan konfigurasi agar dapat berkomunikasi dengan PLC dan variabel yang tepat. Langkah pertama adalah melakukan *setting* pada *connection tab* dengan

memasukan informasi dasar yang telah disesuaikan dengan konfigurasi pada PLC. Konfigurasi pada *connection tab* dapat dilihat pada Gambar 3.25.



Gambar 3.25 Tampilan konfigurasi *connection tab* pada *node s7 in*

Kemudian diperlukan konfigurasi pada *variables tab*. Di *tab* ini perlu dilakukan konfigurasi terhadap variabel yang ingin diambil datanya. Tampilan konfigurasi *variables tab* dapat dilihat pada Gambar 3.26.



Gambar 3.26 Tampilan konfigurasi *variables tab* pada *node s7 in*

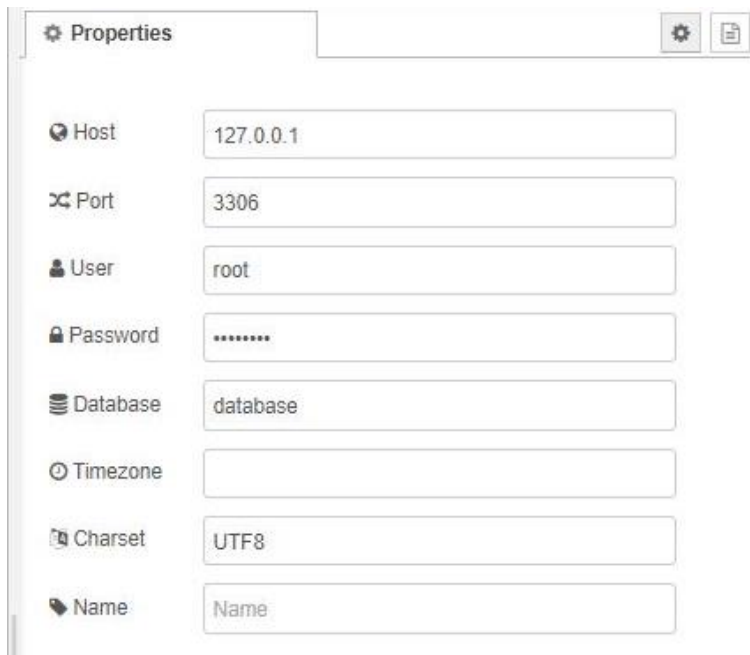
c. Node-red-node-mysql

Pallette ini berfungsi untuk menjalankan perintah *query* dan menerima kembali respon dari *database*. *Node* yang digunakan adalah *node mysql*. Tampilan *node mysql* dapat dilihat pada Gambar 3.27.



Gambar 3.27 Tampilan *node mysql* pada *dashboard pallette*

Node mysql perlu dilakukan konfigurasi agar dapat digunakan dengan baik. Konfigurasi dilakukan dengan mengisi informasi dasar yang telah disesuaikan dengan konfigurasi pada *database MySQL*. Tampilan konfigurasi *node mysql* dapat dilihat pada Gambar 3.28.



Gambar 3.28 Tampilan konfigurasi *node mysql* pada *palette mysql*

3.3.6.2 Flow untuk *Dashboard*

Pada Node-RED program dibagi menjadi 4 *flow*. *Flow* simulasi, *dashboard history*, *dashboard real time* dan *dashboard tabel*. Penjelasan masing-masing *flow* sebagai berikut:

1. *Flow* Simulasi

Flow simulasi adalah letak berjalannya program simulasi. Pada *flow* simulasi terdapat 2 jenis program simulasi yaitu simulasi otomatis dan manual. *Flow* simulasi dapat dilihat pada Gambar 3.29.

a. Simulasi Otomatis

Dapat dilihat pada Gambar 3.29 bahwa pada program simulasi otomatis diawali dengan *timestamp* untuk menjalankan simulasi setiap sekian waktu yang ditetapkan, pada proyek kali ini jarak interval yang dipilih adalah 1 detik. Kemudian menuju ke *node simpletime* untuk mendapatkan waktu. Node berikutnya adalah *node exec* berfungsi untuk menjalankan program python yang sudah dijelaskan pada Sub Bab 3.3.5. *Node exec* (lihat pada Gambar 3.29) perlu dilakukan konfigurasi agar dapat menjalankan tugasnya dengan baik dengan memberi tujuan *directory file* yang ingin di eksekusi. Konfigurasi *node exec* dapat dilihat pada Gambar 3.30.

Proses selanjutnya adalah menuju *node* fungsi *parsing* (lihat Gambar 3.29). Fungsi ini berfungsi untuk melakukan *parsing* terhadap data yang telah didapat dari simulasi program python. Isi dari fungsi *parsing* dapat dilihat pada Gambar 3.31.

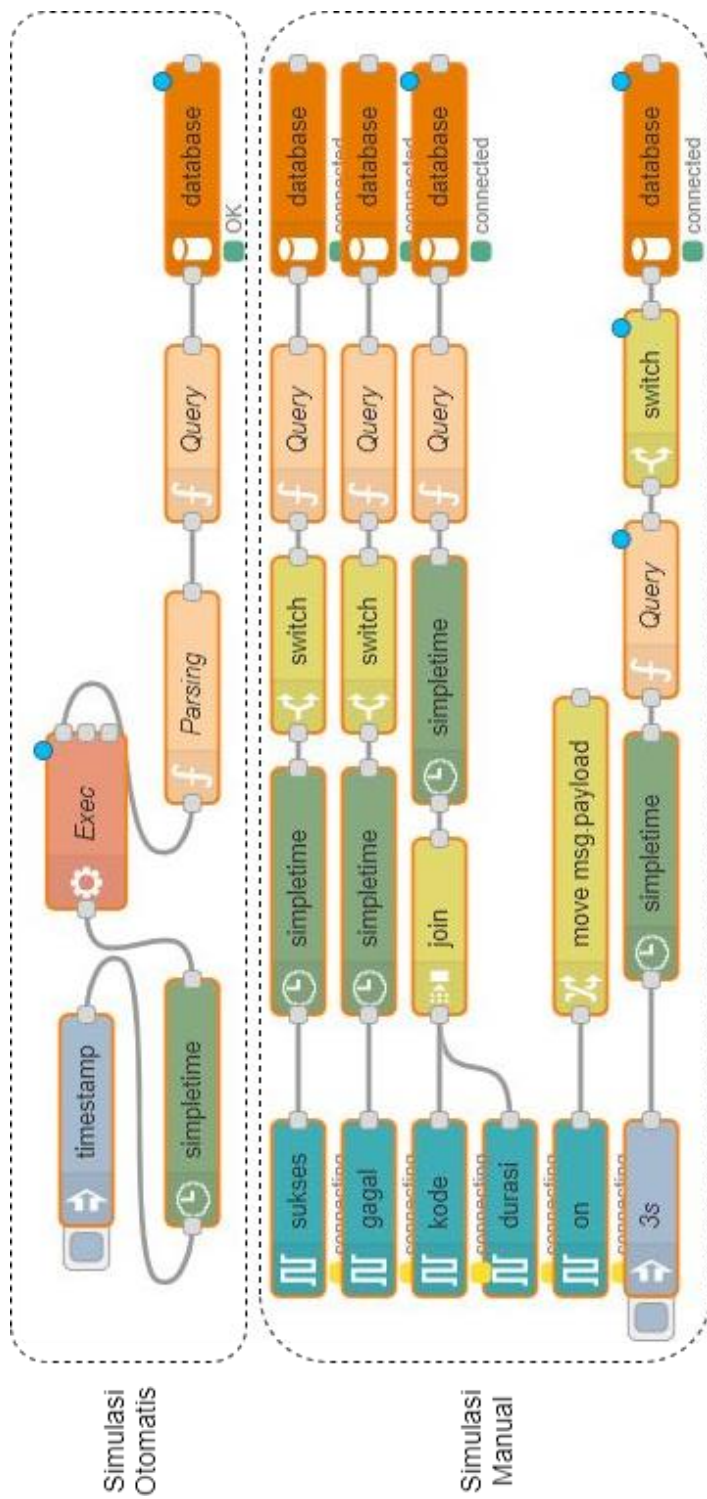
Proses berikutnya adalah menuju *node* fungsi *query* (lihat Gambar 3.29). Data yang telah selesai dilakukan *parsing* dimasukan kedalam perintah *query* yang digunakan untuk menyimpan data pada *database*. Isi fungsi *query* ditunjukan pada Gambar 3.32. *Node* paling kanan pada flow NodeRed (lihat Gambar 3.29) adalah *node* MySQL yang berfungsi untuk mengeksekusi *query* tersebut menuju *database* yang telah dikonfigurasi.

b. Simulasi Manual

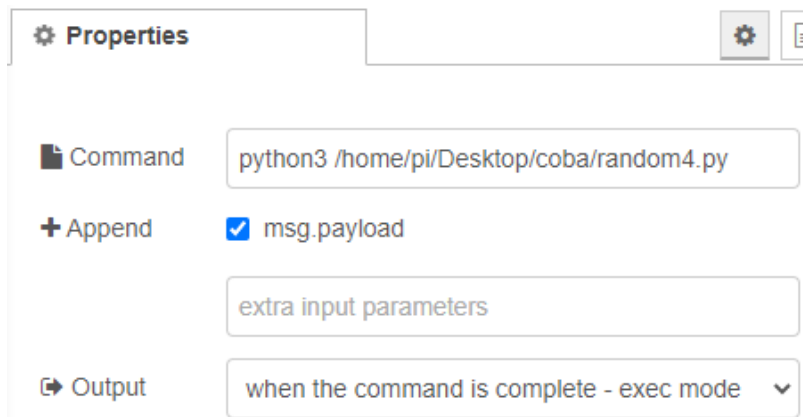
Pada program simulasi manual diawali dengan *node* *s7 in* yang berfungsi untuk mengambil data dari PLC (lihat Gambar 3.29). Data kemudian diolah menggunakan *node* *join*, *move* dan *switch*. *Node* *join* (lihat pada Gambar 3.29) berfungsi untuk menggabungkan lebih dari 1 data yang diterima dan dijadikan *array*. Tampilan konfigurasi *node* *join* dapat dilihat pada Gambar 3.33.

Node *move* (lihat pada Gambar 3.29) berfungsi untuk memindah data dari sebuah variabel menuju ke variabel lainnya. Konfigurasi *node* *move* dapat dilihat pada Gambar 3.34. Sedangkan *node* *switch* (lihat pada Gambar 3.29) berfungsi untuk membandingkan kondisi yang terjadi sebelum melanjutkan data ke proses yang selanjutnya. Konfigurasi *node* *switch* dapat dilihat pada Gambar 3.35.

Data hasil olahan kemudian melalui *node* *simpletime* untuk mendapat format waktu dan diteruskan menuju ke fungsi *query* yang berfungsi untuk memasukan data menuju ke *database* pada *node* MySQL. Isi dari fungsi *query* sama dengan fungsi *query* pada Gambar 3.32. *Node* *timestamp* di bagian paling bawah kiri pada Gambar 3.29 berfungsi untuk mengatur jarak waktu *input query*.



Gambar 3.29 Tampilan *flow* simulasi



Gambar 3.30 Tampilan konfigurasi pada *node exec*

```
var output = msg.payload.split(",");
msg.total_produksi = parseInt(output[0]);
msg.total_gagal = parseInt(output[1]);
msg.shift= parseInt(output[2]);
msg.error = output[3];
msg.up_dt= parseInt(output[4]);
msg.p_dt=parseInt(output[5]);
msg.avl=parseInt(output[6]);

return msg;
```

Gambar 3.31 Isi fungsi *parsing*

```
msg.topic          =          "INSERT          into
PLC(total_produksi,total_gagal,shift)
VALUES (" +msg.total_produksi+", "+msg.total_gagal+", "+msg.
shift+"); INSERT into PLC_eror(error,unplanned_dt,shift)
VALUES (' "+msg.error+"', "+msg.up_dt+", "+msg.shift+");
INSERT into PLC_waktu(available_t,planned_dt,shift)
```

Gambar 3.32 Isi fungsi *query*

Mode manual ▼

Combine each ▼ msg. payload

to create an Array ▼

Send the message:

- After a number of message parts 2
- After a timeout following the first message seconds

Gambar 3.33 Konfigurasi *node join*

Rules

Move ▼ ▼ msg. payload

to ▼ global. on

Gambar 3.34 Konfigurasi *node move*

Property ▼ msg. payload

is true ▼ → 1

Gambar 3.35 Tampilan konfigurasi *node switch*

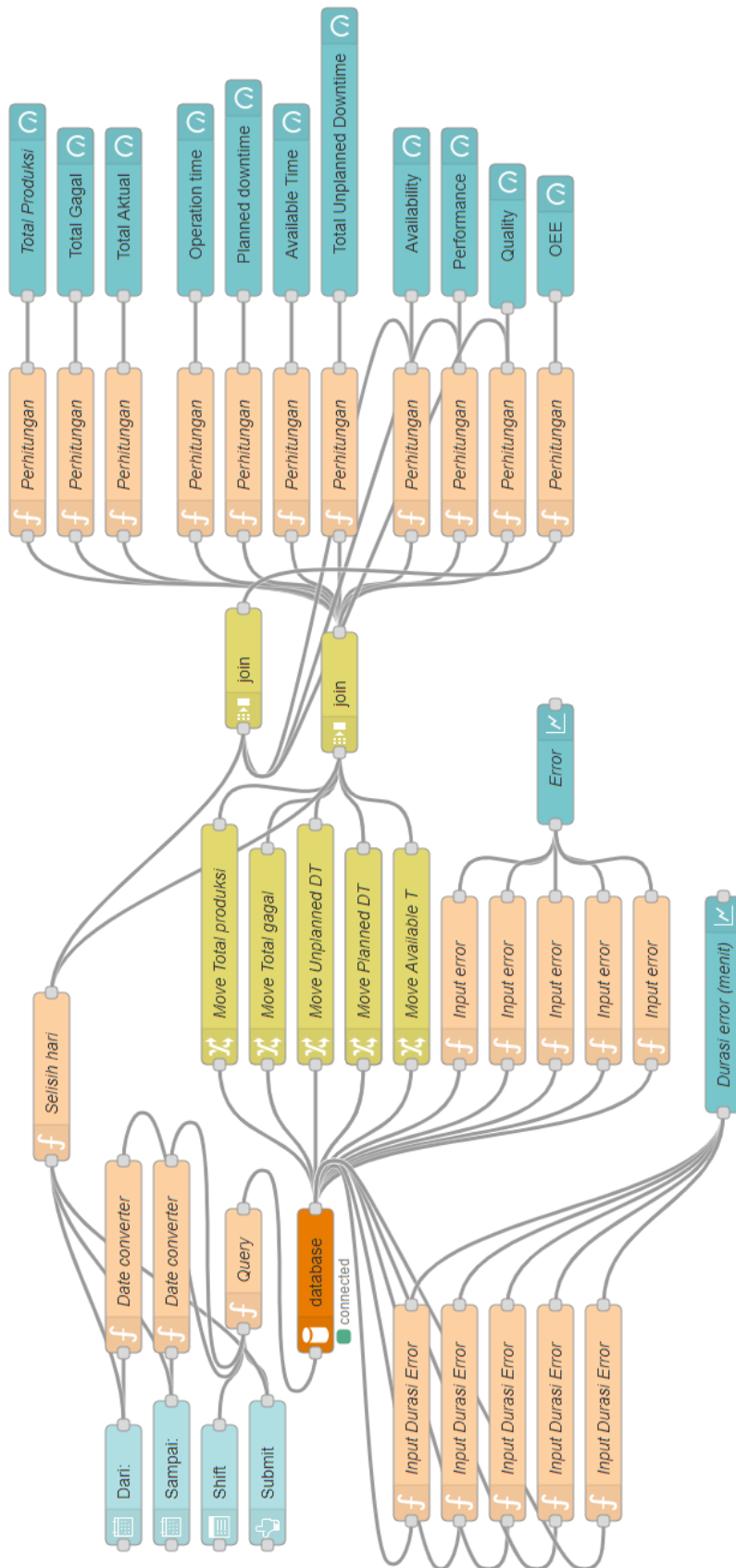
2. *Flow Dashboard History*

Flow dashboard history berfungsi untuk mengambil data secara spesifik berdasarkan parameter jarak hari dan *shift*. Tampilan dari *flow dashboard history* dapat dilihat pada Gambar 3.36.

Input parameter dilakukan menggunakan *node date picker* dan *node dropbox* yang kemudian dimasukkan kedalam variabel *context* pada fungsi *query* (lihat Gambar 3.30). Untuk *date picker* perlu melalui fungsi *date converter* untuk mengubah format waktu yang didapat agar

dapat sesuai dengan format pada *database*. Fungsi pada *node query* juga memiliki fungsi mengirim *query* berisi parameter yang telah tersimpan pada variabel *context* menuju *database* bila tombol *submit* ditekan.

Node database berfungsi untuk mengirim *query* dan menerima balasan dari *query* tersebut (Lihat Gambar 3.27). Data yang didapat kembali dari *query* kemudian dipilah dan diolah agar dapat ditampilkan pada *dashboard*. *Node move* berfungsi untuk memindah nilai dari suatu variabel menuju ke variabel lainnya, sedangkan *node join* berfungsi untuk menggabungkan nilai variabel dari banyak *node* yang berbeda agar dapat diolah dalam *function*. *Node function* perhitungan berisikan rumus penghitungan data yang diinginkan kemudian dibagi dengan selisih hari pemilihan tanggal agar muncul rata-rata data dari tanggal yang dipilih. Penghitungan selisih hari dihitung menggunakan *function* selisih hari yang isinya ditampilkan pada Gambar 3.37.



Gambar 3.36 Tampilan *flow dashboard history*

```

var data=context.get('data')||{shift:""};
var topic=msg.topic;
var payload=msg.payload;

if(topic=="sdate")
data.awal=msg.payload;
if(topic=="edate")
data.akhir=msg.payload;
context.set('data',data)

if(topic=="submit"){
    msg.payload=(data.akhir-
data.awal)/(1000*3600*24)+1;
    if (msg.payload<0){
        msg.payload=msg.payload*-1}
    else{
        msg.payload=msg.payload;}
    return msg;}
return;

```

Gambar 3.37 Isi *function* perhitungan selisih hari

Sedangkan isi salah satu dari fungsi perhitungan data dapat dilihat pada Gambar 3.38.

```

msg.payload = ((msg.payload[1]-msg.payload[2])
*100/msg.payload[1]);
msg.payload = msg.payload.toFixed(2);
return msg;

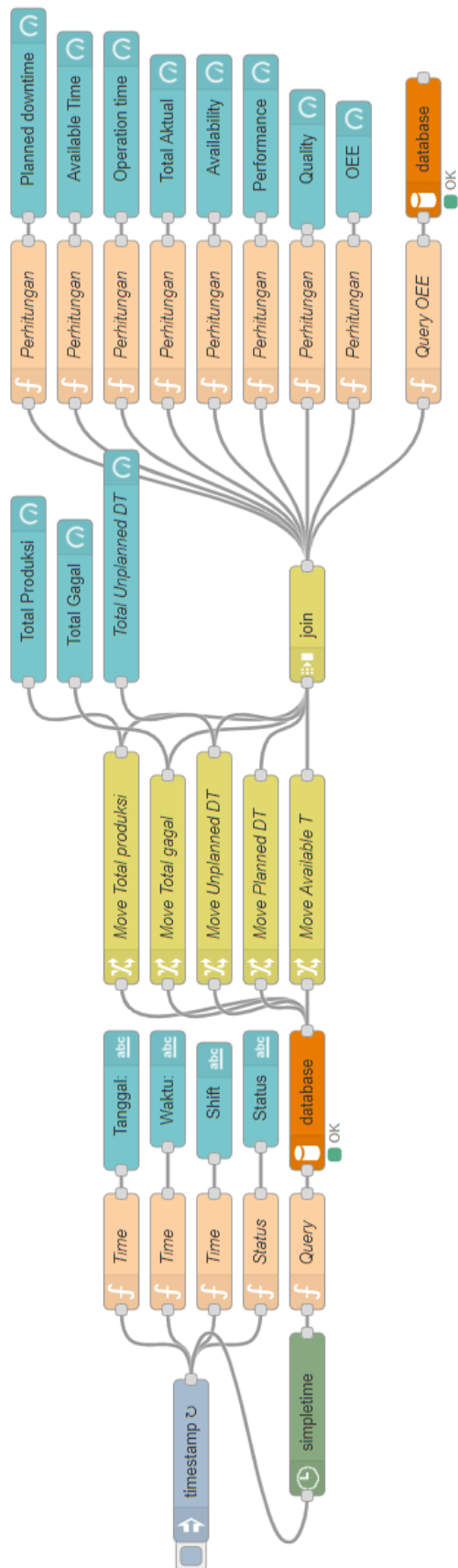
```

Gambar 3.38 Isi *function* perhitungan data

Data hasil dari perhitungan kemudian akan dikirim menuju *node* UI sesuai dengan kebutuhan (lihat Gambar 3.36).

3. *Flow Dashboard Real Time*

Flow dashboard real time berfungsi untuk menampilkan data *real-time* simulasi yang sedang terjadi. Tampilan dari *flow dashboard* dapat dilihat pada Gambar 3.39.



Gambar 3.39 Tampilan *flow dashboard real time*

Flow ini dimulai dengan *timestamp* yang mengatur jarak interval dalam menjalankan *flow* (lihat Gambar 3.39). Selanjutnya fungsi *time* dipakai untuk menampilkan informasi mengenai waktu, shift, dan status secara *real-time*. *Node simpletime* dipakai untuk mendapatkan format waktu yang dibutuhkan untuk memanggil data secara *real-time*. Format waktu tersebut kemudian diolah pada Fungsi *query* untuk mengambil data yang dibutuhkan. Fungsi *query* berisi *query* yang memanggil total jumlah data tiap variabel yang disimpan pada *database*. Tampilan fungsi *query* dapat dilihat pada Gambar 3.40.

```
msg.topic="SELECT SUM(`total_produksi`) FROM Produksi WHERE
tanggal      ='" +msg.myymd+"';SELECT SUM(`total_gagal`) FROM
Produksi      WHERE      tanggal      ='" +msg.myymd+"';SELECT
SUM(`unplanned_dt`) FROM Sistem_error WHERE tanggal
='"+msg.myymd+"';SELECT SUM(`planned_dt`) FROM Waktu_produksi
WHERE tanggal ='" +msg.myymd+"';SELECT SUM(`available_t`) FROM
Waktu_produksi WHERE tanggal ='" +msg.myymd+"';";

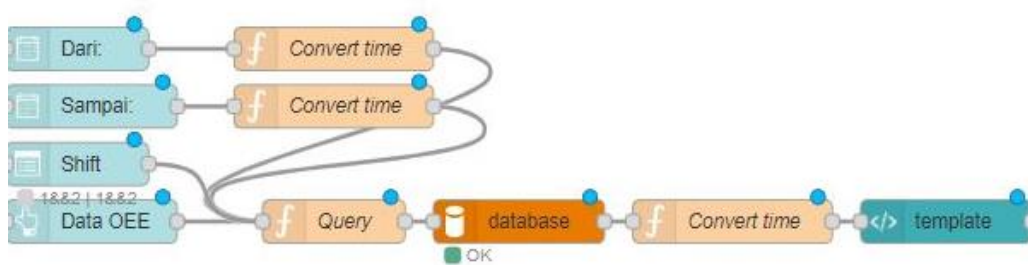
return msg;
```

Gambar 3.40 Isi fungsi *query* pada *flow dashboard real time*

Node database berfungsi untuk mengeksekusi *query* dan mendapat respon dari *query* tersebut dan diteruskan menuju *node move* untuk disimpan ke variabel yang diinginkan (lihat Gambar 3.30). Data kemudian dikirim ke *node join* untuk digabung agar dapat diolah dan dihitung pada *node function* perhitungan. Data yang telah diolah kemudian ditampilkan pada *dashboard* menggunakan *node UI* yang telah dikonfigurasi. Pada *flow* ini juga terdapat *function* khusus yang berfungsi untuk menyimpan hasil OEE per-shift ke dalam *database* yang terdapat pada *node* paling kanan bawah pada Gambar 3.39.

4. *Flow Dashboard Table*

Flow dashboard table berfungsi untuk menerima *input* jarak tanggal dan *shift* dari *dashboard* untuk mengambil data *availability*, *performance* dan *quality* dalam bentuk tabel. Tampilan dari *flow* data OEE dapat dilihat pada Gambar 3.41.



Gambar 3.41 Tampilan *flow* data OEE

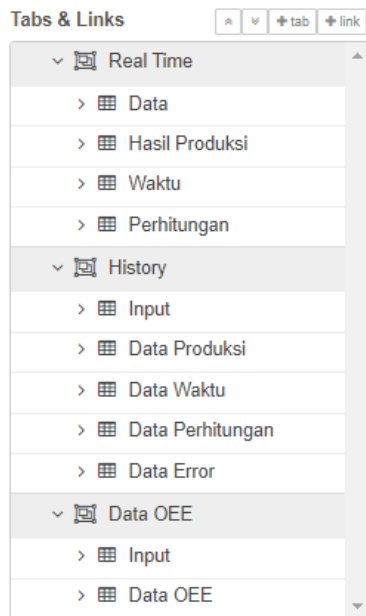
Flow dimulai dengan *input* tanggal, *input* shift dan input tombol. *Input* tanggal perlu diolah terlebih dahulu dengan *function* *convert time* agar format tanggal sesuai dengan format pada *database* (lihat Gambar 3.35). Kemudian pada fungsi *query input*, tanggal dan *input* shift dimasukkan kedalam variabel *context* yang akan dimasukkan kedalam *query*. *Query* akan dieksekusi menuju ke *node mysql* saat *input* tombol ditekan dan akan menerima hasil dari *query* tersebut. Data yang didapat dari *query* yang telah dieksekusi kemudian perlu di ubah format tanggalnya agar dapat ditampilkan dengan baik. Data kemudian dikirim menuju *node template* untuk diolah dan ditampilkan dalam bentuk tabel. *Script* pada *template* dapat dilihat pada Gambar 3.42.

```
<style>table
{background:lightgreen;}.main
{height:200px;}
</style><div class="main">
<table style="width:100%">
  <tr><th>Tanggal</th>
    <th>Shift</th>
    <th>Availability</th>
    <th>Performance</th>
    <th>Quality</th>
    <th>OEE</th>
  </tr><tr ng-repeat="x in msg.payload | limitTo:100">
    <td style="text-align:center">
      {{msg.payload[0][$index].tanggal}}
    </td>
    <td style="text-align:center">
      {{msg.payload[1][$index].shift}}
    </td>
    <td style="text-align:center">
      {{msg.payload[2][$index].availability}}
    </td>
    <td style="text-align:center">
      {{msg.payload[3][$index].performance}}
    </td>
    <td style="text-align:center">
      {{msg.payload[4][$index].quality}}
    </td>
    <td style="text-align:center">
      {{msg.payload[5][$index].oeo}}
    </td>
  </tr>
</table></div>
```

Gambar 3.42 Isi *node template* pada *flow dashboard* tabel

3.3.7 Desain Dashboard

Dashboard proyek ini memiliki 11 grup yang dibagi kepada 3 *tab*, yaitu *tab real time*, *tab history* dan *tab data OEE*. Daftar *tab* dan grup dapat dilihat pada Gambar 3.43.

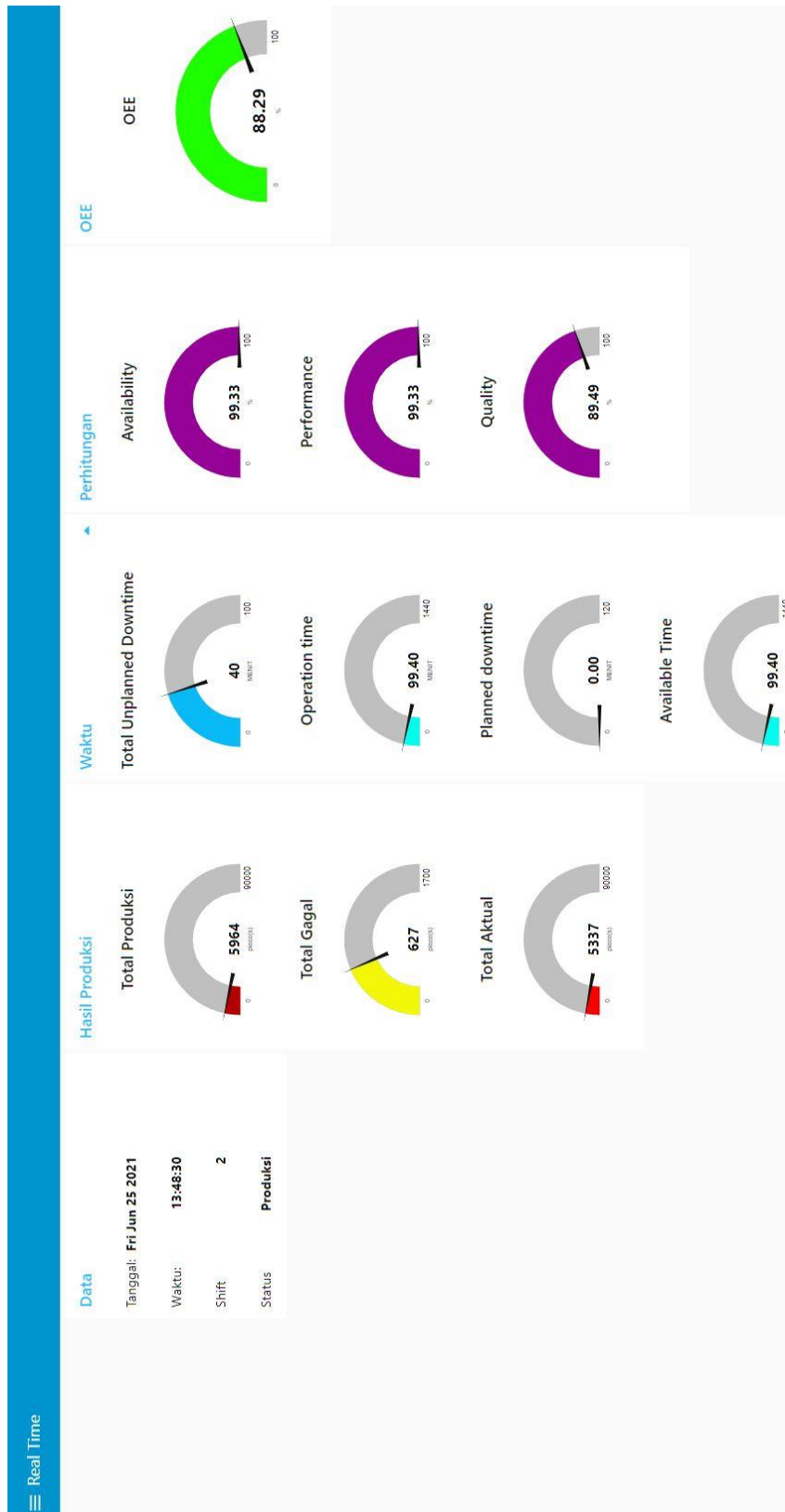


Gambar 3.43 Daftar *tab* dan grup

Penjelasan masing-masing *tab* sebagai berikut:

a. *Tab Real Time*

Pada *tab real time* terdapat nilai data simulasi yang sedang terjadi saat itu juga dan akan *update* secara *real time*. Tampilan *tab real time* dapat dilihat pada Gambar 3.44.



Gambar 3.44 Tampilan *tab real time*

Tab real time terdiri dari 4 grup, yaitu grup data, grup hasil produksi, grup waktu dan grup perhitungan. Grup data berisi informasi mengenai tanggal, waktu, *shift* dan status produksi. Grup hasil produksi berisi data barang total produksi, total gagal dan total aktual. Grup waktu berisi tentang data waktu pada produksi yaitu total *unplanned downtime*, *operation time*, *planned downtime* dan *available time*. Sedangkan grup perhitungan berisi data hasil perhitungan dari rumus yaitu *availability*, *performance*, *quality* dan *OEE*. Tab ini akan menampilkan data *real time* dalam 1 hari.

b. *Tab History*

Tab history adalah halaman untuk melihat data produksi dan perhitungan yang telah terjadi dalam *shift* dan jarak waktu tertentu. Pada *tab* ini terdapat 5 grup, yaitu grup *input*, grup data produksi, grup data waktu, grup data perhitungan dan grup data *error*. Pada grup *input* terdapat pemilihan *shift*, pemilihan *start date*, pemilihan *end date* dan tombol *submit*. Grup *input* berfungsi untuk melakukan *input* parameter yang ingin diambil. Pada grup data produksi terdapat 3 variabel yaitu total produksi, total gagal dan total aktual. Grup data waktu memiliki 4 variabel yaitu *total unplanned downtime*, *operation time*, *planned downtime* dan *available time*. Grup perhitungan memiliki 4 variabel hasil perhitungan yaitu *availability*, *performance*, *quality* dan *OEE*. Sedangkan grup *error* memiliki 2 variabel data *error* yang terjadi yaitu jenis *error* dan durasi *error*. Tampilan *tab history* dapat dilihat pada Gambar 3.45.



Gambar 3.45 Tampilan *tab history*

c. *Tab Data OEE*

Tab data OEE adalah halaman untuk melihat daftar OEE dalam bentuk tabel agar lebih mudah untuk menganalisa nilai OEE dalam untuk lebih dari 1 jangka waktu. Tampilan dari *tab* data OEE dapat dilihat pada Gambar 3.46.

Input

Dari:  11/05/2021

Shift 1 dan 2

Sampai:  12/05/2021

DATA OEE

Data OEE

Tanggal	Shift	Availability	Performance	Quality	OEE
11/5/2021	1	99.93	91.59	98.37	90.02
11/5/2021	2	99.91	91.58	98.26	89.91
12/5/2021	1	99.92	91.59	98.33	89.99
12/5/2021	2	99.91	91.59	98.39	90.03

Gambar 3.46 Tampilan *tab* data OEE

Tab data OEE memiliki 2 grup yaitu grup *input* dan grup data OEE. Dalam grup *input* terdapat pilihan *start date*, *end date*, *shift* dan tombol *submit*. Grup *input* berfungsi untuk memilih parameter yang ingin ditampilkan di tabel. Grup data OEE merupakan grup yang berisi tabel yang akan menampilkan tanggal, *shift*, *availability*, *performance*, *quality* dan OEE dalam parameter yang telah dipilih pada grup *input*.

3.3.8 Desain Database

Database memiliki peran penting dalam proyek ini karena *database* akan berfungsi untuk menyimpan seluruh data yang akan diolah dan ditampilkan pada *dashboard*. *Database* proyek ini menggunakan 4 tabel yaitu tabel Produksi, Sistem_error, Waktu_produksi dan OEE.

a. Tabel Produksi

Tabel Produksi adalah tabel tempat menyimpan data hasil produksi. Data yang disimpan adalah total produksi, total gagal, *shift*, tanggal dan waktu. Tampilan struktur dari tabel Produksi dapat dilihat pada Gambar 3.47.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 🗝️	int(99)			No	None		AUTO_INCREMENT
2	total_produksi	int(11)			No	None		
3	total_gagal	int(11)			No	None		
4	shift	int(2)			No	None		
5	tanggal	date			No	current_timestamp()		
6	jam	time			No	current_timestamp()		

Gambar 3.47 Struktur tabel Produksi pada *database*

b. Tabel Sistem_error

Tabel Sistem_error berisi informasi mengenai *error* yang terjadi pada saat produksi. Data yang disimpan adalah jenis *error*, durasi *error*, tanggal, waktu dan *shift*. Tampilan struktur dari tabel Sistem_error dapat dilihat pada Gambar 3.48.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 🗝️	int(50)			No	None		AUTO_INCREMENT
2	eror	varchar(20)	utf8mb4_general_ci		No	None		
3	unplanned_dt	int(10)			No	None		
4	tanggal	date			No	current_timestamp()		
5	jam	time			No	current_timestamp()		
6	shift	int(2)			No	None		

Gambar 3.48 Struktur tabel Sistem_error pada *database*

c. Tabel Waktu_produksi

Tabel Waktu_produksi berisi informasi mengenai durasi yang terjadi pada saat produksi. Data yang disimpan adalah *available time*, *planned downtime*, tanggal, waktu dan *shift*. Tampilan struktur dari tabel Waktu_produksi waktu dapat dilihat pada gambar 3.49.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 	int(50)			No	None		AUTO_INCREMENT
2	available_t	int(11)			No	None		
3	planned_dt	int(11)			No	None		
4	tanggal	date			No	current_timestamp()		
5	jam	time			No	current_timestamp()		
6	shift	int(2)			No	None		

Gambar 3.49 Struktur tabel Waktu_produksi waktu pada *database*

d. Tabel OEE

Tabel OEE berisi data hasil perhitungan OEE dalam setiap *shift* produksi. Data yang disimpan adalah *availability*, *performance*, *quality*, *OEE*, tanggal, waktu dan *shift*. Tampilan struktur dari tabel OEE dapat dilihat dari Gambar 3.50.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 	int(11)			No	None		AUTO_INCREMENT
2	availability	float			No	None		
3	performance	float			No	None		
4	quality	float			No	None		
5	oee	float			No	None		
6	tanggal	date			No	current_timestamp()		
7	jam	time			No	current_timestamp()		
8	shift	int(2)			No	None		

Gambar 3.50 Struktur tabel OEE pada *database*

3.3.9 Koneksi Sistem IoT

Koneksi IoT dilakukan dengan memanfaatkan Dataplicity, NGINX dan WLAN pada Raspberry Pi 4. Dataplicity dan NGINX diperlukan agar Node-RED *dashboard* dapat diakses oleh *client* melalui internet menggunakan koneksi WLAN.

Karena Raspberry Pi 4 menggunakan WLAN sebagai koneksi internet dan ethernet sebagai koneksi data ke sistem proyek, maka perlu dilakukan konfigurasi terhadap prioritas jaringan. Konfigurasi prioritas jaringan dapat dilakukan dengan membuka terminal dan menggunakan perintah `sudo nano /etc/dhcpd.conf`. Setelah itu masukan perintah sesuai pada Gambar 3.51 untuk mengubah prioritas jaringan.

```
interface wlan0
metric 1

interface eth0
noipv6
static
ip_address=192.168.0.11/24
metric 2
```

Gambar 3.51 Tampilan perintah pengubahan prioritas jaringan

Selanjutnya perlu melakukan konfigurasi pada Dataplicity. Konfigurasi dilakukan dengan mengakses situs dataplicity.com dan melakukan pendaftaran akun. Setelah akun sudah dibuat maka akan didapatkan perintah untuk dieksekusi pada terminal Raspberry Pi agar Raspberry Pi dan Dataplicity terkoneksi. Jika sudah maka pastikan pilihan wormhole menyala agar fitur *web hosting* dapat digunakan dan Raspberry Pi sebagai server. Kemudian langkah berikutnya adalah melakukan instalasi NGINX dengan menggunakan perintah `sudo apt install nginx` pada terminal Raspberry Pi. Kemudian hal selanjutnya yang perlu dilakukan adalah melakukan perubahan pada situs yang ingin ditampilkan, dengan cara mengakses file *default* di folder *sites-available* pada folder *nginx* atau memasukan perintah `sudo nano /etc/nginx/sites-available/default` dan memasukan perintah sesuai pada Gambar 3.52 dimana `http://localhost:1880:ui/` adalah alamat *dashboard* yang akan dihosting.

```
location / {
    proxy_pass http://localhost:1880/ui;
}
```

Gambar 3.52 Tampilan perintah pada file *default*

Perintah tersebut berfungsi untuk menghubungkan alamat *dashboard* dengan *link wormhole* yang telah diberikan pada situs Dataplicity yaitu `https://obtudent-salmon-6076.dataplicity.io/`. *Link* dari *wormhole* kemudian dipersingkat menggunakan situs bitly.com sehingga menjadi `petra.id/dashboardOEE`.

PHPMyAdmin dan NGINX menggunakan *port* yang sama untuk beroperasi yaitu *port* 80. Hal ini menyebabkan PHPMyAdmin dan NGINX tidak dapat berjalan secara paralel. Sehingga jika ingin mengakses NGINX maka apache2 harus diberhentikan terlebih dahulu menggunakan `sudo service apache2 stop` dan menyalakan NGINX dengan menggunakan perintah `sudo`

`service nginx start` pada terminal Raspberry Pi, hal yang sebaliknya harus dilakukan bila ingin mengakses PHPMyAdmin.

3.3.10 Implementasi Secara Nyata

Sistem yang digunakan pada proyek ini dapat digunakan pada sistem produksi secara nyata namun membutuhkan sedikit penyesuaian. Hal yang perlu dilakukan untuk menggunakan sistem ini secara *real* adalah dengan mengganti program pada PLC sesuai dengan *flowchart* produksi pada lapangan. Kemudian *input* dan *output* pada program PLC perlu dilakukan *wiring* pada PLC maupun pada mesin. Yang terakhir adalah menyesuaikan program yang pada PLC dan mesin pada lapangan dengan program pada SCADA untuk menunjukkan tampilan maupun *input* yang dibutuhkan.