BAB I
PENDAHULUAN

1.1 Latar Belakang

Sistem perparkiran di beberapa kota besar di Indonesia pada saat ini menghadapi berbagai kendala, misalnya masalah tempat dimana sempitnya lahan perparkiran dan bertambahnya jumlah kendaraan bermotor. Sempitnya lahan parkir disebabkan oleh perkembangan kota yang sangat pesat, ditandai salah satunya dengan menjamurnya gedung – gedung bertingkat, perkantoran, pusat perbelanjaan, dan industri-industri. Hal ini menyebabkan berkurangya jumlah lahan hijau yang merupakan sumber penghasil oksigen (O_2). Disisi lain seiring dengan meningkatnya kemampuan daya beli kendaraan bermotor dari masyarakat perkotaan, dimana masyarakat perkotaan mempunyai tingkat pendapatan dan sifat konsumtif yang lebih tinggi dari masyarakat desa. Sehingga pertumbuhan jumlah kendaraan bermotor khususnya mobil semakin pesat yang menyebabkan bertambahnya polusi udara yang diakibatkan dari emisi gas buang dari kendaraan tersebut.

Berkenaan dengan hal diatas untuk menyiasati terbatasnya area parkir, pengelola pusat perbelanjaan biasanya membuat area parkir yang bertingkat (vertikal) untuk memenuhi kebutuhan lahan parkir pengunjung di area yang sempit. Tetapi solusi ini sering terjadi kendala bagi penyedia parkir dan pengguna parkir. Bangunan area parkir yang bertingkat dan banyaknya pengunjung pusat perbelanjaan yang memarkir mobilnya tidak teratur seringkali membuat pengendara lain kebingungan mencari tempat parkir yang kosong. Selain itu, kondisi ketidakteraturan parkir kendaraan akan berdampak pada waktu dan tingkat keamanan kendaraan. Secara tidak langsung dengan banyaknya mobil yang berputar-putar mencari lokasi parkir mengakibatkan meningkatnya tingkat polusi emisi karbon monoksida pada tempat tersebut.

Dari permasalahan diatas beberapa orang melakukan penelitian untuk menciptakan sistem parkir yang lebih baik diantaranya yaitu sistem parkir menggunakan *take place and buy system*, sistem parkir menggunakan RFID *(Radio Frequency Identification)* dan *mikrokontroller*. Dari kedua sistem tersebut mempunyai kelebihan

yaitu lebih efisien dan praktis dari segi pembayarannya. Namun, sistem parkir tersebut belum bisa menginformasikan lokasi tempat parkir yang masih kosong dan dengan sistem tersebut tidak memungkinkan untuk bisa melakukan *monitoring* secara *real-time* karena sistem tersebut dikontrol dengan *Visual Basic*.

Berdasarkan permasalahan tersebut maka timbulah ide untuk membuat sistem parkir yang dapat memberikan informasi lokasi tempat parkir yang kosong dengan menggunakan alat kontrol yaitu PLC *(Programmable Logic Controller)*. Selain itu dalam membangun suatu sistem parkir diatas kita juga membutuhkan media komunikasi yaitu *Supervisory Control and Data Acquisition* (SCADA) yang berfungsi sebagai media komunikasi antara manusia, PC dan PLC. Pada penelitian Tugas Akhir ini akan dibuat desain sistem SCADA yang memungkinkan operator tempat parkir dapat melakukan *controlling* dan *monitoring* sistem parkir tersebut. Perangkat lunak yang digunakan untuk membangun sistem SCADA pada penelitian Tugas Akhir ini adalah Vijeo Citect.

1.2 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

a. Membuat desain sistem SCADA sistem parker berbasis PLC yang dapat menampilkan secara otomatis data keadaan tempat parkir yang kosong, harga/tarif parkir, waktu parkir dan dapat mengontrol serta memonitoring keadaan tempat parkir secara *real-time*.
b. Membuat konfigurasi komunikasi *serial port* antara SCADA dan PLC.

1.3 Batasan Masalah

Dalam penelitian ini, penulis memberikan batasan masalah supaya tujuan yang diharapkan dapat memberikan hasil yang maksimal, diantaranya:

a. Pembuatan desain SCADA menggunakan *software* Vijeo Citect v7.10 r2.
b. PLC yang digunakan adalah PLC Omron Sysmac tipe CP-1L-M40DR-A.
c. Bahasa pemrograman PLC yang digunakan adalah *ladder diagram.*
d. Komunikasi SCADA dengan PLC menggunakan kabel serial RS 232.

1.4 Metodologi Penelitian

Metodologi penelitian yang digunakan dalam Tugas Akhir kali ini adalah:

a. Studi Pustaka

   Studi pustaka adalah suatu metode yang dipergunakan dalam penelitian ilmiah yang dilakukan dengan membaca dan mengolah data yang diperoleh dari literatur. Data yang dibaca dan diolah adalah data yang berhubungan dengan hasil-hasil eksperimen yang telah dilakukan dan dibukukan oleh para peneliti sebelumnya.

b. Asistensi dan Konsultasi

   Metode ini bertujuan untuk mendapatkan bimbingan pengetahuan dan masukan dari dosen pembimbing serta koreksi terhadap kesalahan-kesalahan yang terjadi dalam pembuatan Tugas Akhir dan penyusunan laporan.

c. Perancangan Sistem SCADA

   Pada metode ini akan dilakukan proses perancangan terhadap desain sistem parkir yang akan dibuat. Hal-hal yang dilakukan meliputi perancangan diagram alir proses sistem, perancangan *ladder diagram* untuk pemrograman PLC, dan perancangan sistem SCADA untuk *controlling* serta *monitoring* sistem parkir tersebut.

d. Pengujian Sistem dan Pembahasan

   Dalam pengujian sistem SCADA ini bertujuan untuk mengetahui apakah sistem ini berjalan sesuai dengan konsep sistem parkir yang diinginkan. Setelah sistem berjalan dengan sukses, kemudian akan dilakukan pembahasan dari pengujian sistem tersebut. Pembahasan tersebut meliputi pembahasan hasil prancangan dan pengujian sistem SCADA serta pembahasan konfigurasi kominikasi antara PLC dengan SCADA. Untuk selanjutnya dari pembahasan tersebut diperoleh kesimpulan, yang nantinya akan ditulis dalam laporan Tugas Akhir.

1.5 Sistematika Penulisan

Sistematika penulisan Laporan Tugas Akhir yang direncanakan terdiri dari 5 bab. Bab I Pendahuluan, berisi tentang latar belakang masalah, tujuan penelitian, pembatasan masalah, metode penelitian dan sistematika penulisan. Kemudian dilanjutkan dengan Bab II yang berisi Dasar Teori. Dalam bab ini berisi penjelasan tentang SCADA, PLC, dan konfigurasi komunikasinya. Bab yang berisi tentang diagram alir penelitian,

langkah-langkah dalam pembuatan sistem dan pembuatan simulasi dijelaskan ke dalam Bab III yaitu Perencanaan Sistem.

Setelah Bab III, dilanjutkan dengan bab yang berisi tentang pengujian sistem dan pembahasan terhadap hasil perancangan dan pengujian dari SCADA sistem parkir. Bab tersebut masuk ke dalam bab IV Pengujian Sistem dan Pembahasan. Setelah itu dilanjutkan dengan Bab V Penutup, bab ini berisi tentang kesimpulan dari hasil pembuatan dan saran untuk penelitian selanjutnya agar didapatkan hasil yang lebih baik.

Daftar Pustaka ditulis setelah Bab V penutup, berisi tentang sumber materi yang digunakan. Dan yang terakhir adalah lampiran, berisi tentang data atau hasil tambahan yang berhubungan dengan penelitian.

BAB II
DASAR TEORI

2.1 Pengertian Sistem *Supervisory Control and Data Acquisition* (SCADA)

Sistem SCADA adalah suatu metode dalam sistem kontrol, dimana operator dapat melakukan fungsi kontrol (*controlling*), pengawasan (*monitoring*) dan pengambilan serta perekaman data (*data acquisition*) dari sebuah sistem yang sedang bekerja. SCADA dapat difungsikan sebagai sistem yang dapat mengumpulkan informasi atau data-data dari lapangan dan kemudian mengirimkan-nya ke sebuah komputer sentral yang akan mengatur dan mengontrol data-data tersebut. Sistem SCADA tidak hanya digunakan dalam proses-proses industri, misalnya, pabrik baja, pembangkit, dan pendistribusian tenaga listrik (konvensional maupun nuklir), pabrik kimia, tetapi juga pada beberapa fasilitas eksperimen seperti fusi nuklir[1].

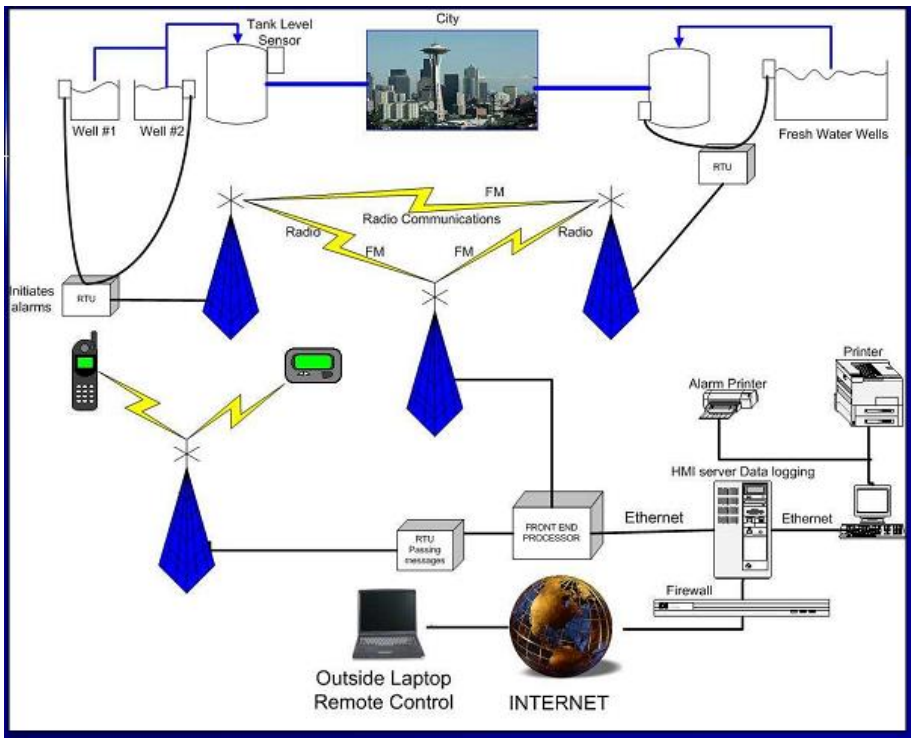
2.1.1 Fungsi Sistem SCADA

SCADA dapat digunakan untuk mengatur berbagai macam peralatan. Biasanya sistem SCADA pada PLC digunakan untuk melakuan proses industri yang kompleks secara otomatis, dapat menggantikan tenaga manusia dan biasanya merupakan proses-proses yang melibatkan faktor-faktor kontrol yang lebih banyak dan berbahaya, serta faktor-faktor kontrol gerakan cepat, dan lain sebagainya. SCADA dapat digunakan dalam aplikasi-aplikasi yang membutuhkan kemudahan dalam pemantauan sekaligus juga pengontrolan, dengan berbagai macam media *interface* dan komunikasi yang tersedia saat ini. Berikut ini beberapa hal yang bisa dilakukan dengan sistem SCADA[1]:

a. Mengakses pengukuran kuantitatif dan proses-proses yang penting, secara langsung saat itu maupun sepanjang waktu.
b. Mendeteksi dan memperbaiki kesalahan secara cepat.
c. Mengontrol proses-proses yang lebih besar dan kompleks dengan staf-staf terlatih yang lebih sedikit.

Sebuah sistem SCADA memberikan keleluasaan dalam mengatur maupun mengkonfigurasi sistem. Semakin banyak hal yang bisa dipantau, semakin detail operasi yang dilihat, dan semuanya bekerja secara *real-time*. Sehingga sekompleks apapun

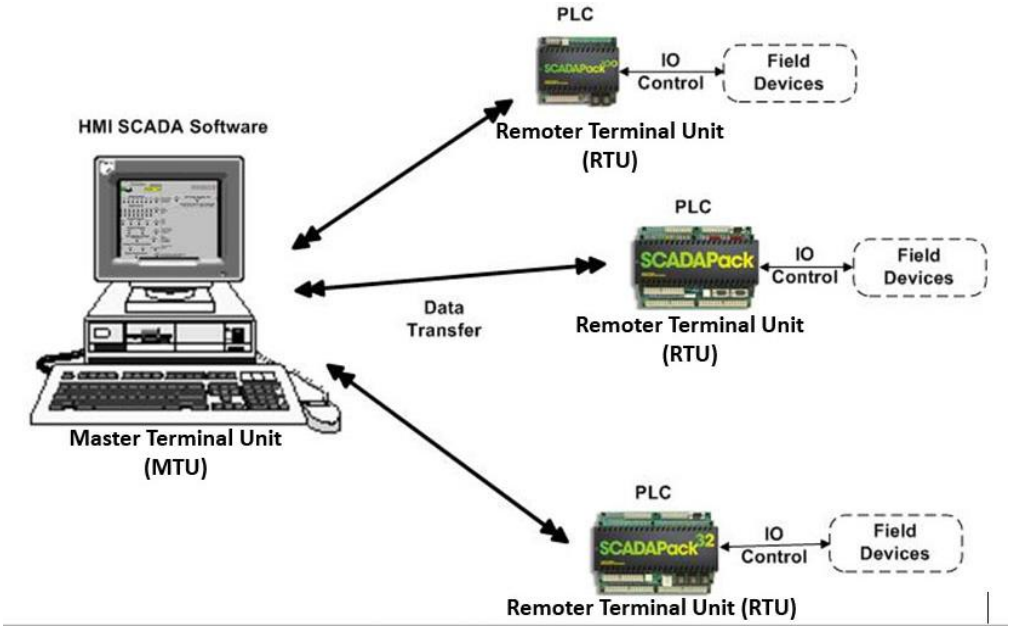
proses yang ditangani oleh PLC, operator dari *plant* bisa melihat operasi proses dalam skala yang besar maupun kecil, dan operator bisa melakukan penelusuran jika terjadi kesalahan untuk meningkatkan efisiensi. Gambar 2.1 merupakan salah satu contoh pemanfaatan sistem SCADA untuk mengontrol distribusi air bersih[2].



Gambar 2.1 Kontrol distribusi air bersih menggunakan sistem SCADA[2]

2.1.2 Perangkat Keras Sistem SCADA

Ada banyak bagian dalam sebuah sistem SCADA. Sebuah sistem SCADA biasanya memiliki perangkat keras sinyal untuk memperoleh dan mengirimkan I/O, kontroler, jaringan, antarmuka pengguna dalam bentuk HMI (*Human Machine Interface*), piranti komunikasi dan beberapa perangkat lunak pendukung. Semua itu menjadi satu sistem, jadi istilah SCADA merujuk pada sistem pusat keseluruhan. Sistem sentral ini biasanya melakukan pemantauan data-data dari berbagai macam sensor di lapangan atau bahkan dari tempat-tempat yang lebih jauh lagi (*remote locations*). Gambar 2.2 merupakan bagian perangkat keras sistem SCADA[3].



Gambar 2.2 Perangkat keras sistem SCADA[3]

Sistem pemantauan dan kontrol industri biasanya terdiri dari sebuah *host* sentral atau master (biasa dinamakan sebagai *master station, master terminal unit* atau MTU), salah satu atau lebih unit-unit pengumpul dan kontrol data lapangan (biasa dinamakan *remote station, remoter terminal unit* atau RTU) dan sekumpulan perangkat lunak standar maupun *customized* yang digunakan untuk memantau dan mengontrol elemen-elemen data di lapangan. Sebagian besar sistem SCADA banyak menggunakan komunikasi jarak jauh, walaupun demikian ada beberapa elemen menggunakan komunikasi jarak dekat. Ada dua elemen dalam aplikasi SCADA, yaitu[3]:

a. Proses, sistem, mesin yang akan dipantau dan dikontrol, bisa berupa *power plant*, sistem pengairan, jaringan komputer, sistem lampu trafik lalu-lintas atau *plant* apa saja.
b. Sebuah jaringan peralatan “cerdas” dengan *interface* ke sistem melalui *sensor* dan kontrol *output*. Jaringan yang merupakan sistem SCADA, akan mempermudah untuk melakukan pemantauan dan pengontrolan komponen-komponen sistem yang melalui *sensor* dan kontrol *output* tersebut.

2.1.3 Perangkat Lunak Sistem SCADA

Sistem SCADA mengacu pada kerja PLC, dimana pada PC akan ditunjukkan dan ditampilkan simulasi dan tombol kontrol pada *plant* secara *real-time* dari sistem dengan bantuan perangkat lunak SCADA (dalam hal ini menggunakan program Vijeo Citect). Jadi PC akan memiliki fungsi untuk melakukan *controlling* dan *monitoring plant*. Perangkat lunak SCADA didukung oleh fitur-fitur untuk menampilkan proses dari sistem dengan memanfaatkan *data acquisition.* Sedangkan untuk menghubungkan perangkat lunak SCADA dengan PC agar dapat dikontrol dan diamati oleh operator serta dengan PLC yang bekerja pada *plant*, maka dibutuhkan media komunikasi seperti jalur komunikasi serial pada PC (*serial port PC*) [3].

Pada perangkat lunak sistem SCADA biasanya mempunyai fitur-fitur kunci untuk mendukung kerja sistem SCADA tersebut yaitu:

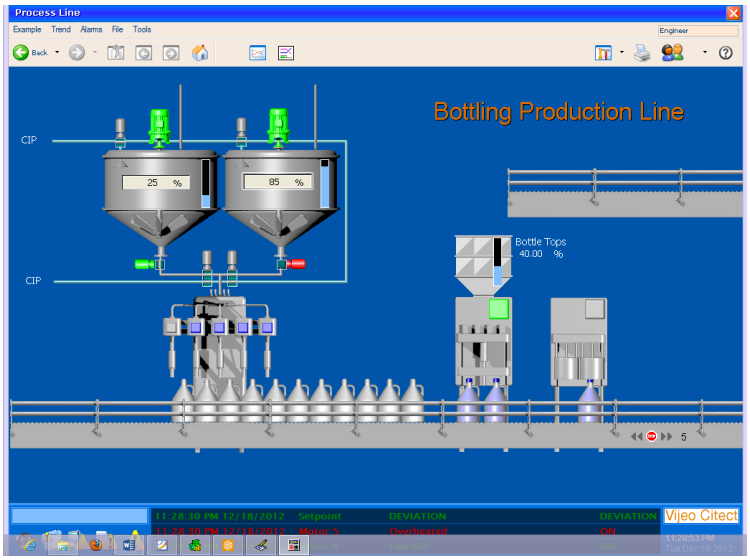
a. *Human Machine Interface*

   Tampilan yang memudahkan manusia (operator) untuk memahami atau mengendalikan mesin (sistem, *plant*) seperti ditunjukkan pada Gambar 2.3.

Gambar 2.3 *Human machine interface*[4]

b. *Graphic Displays*

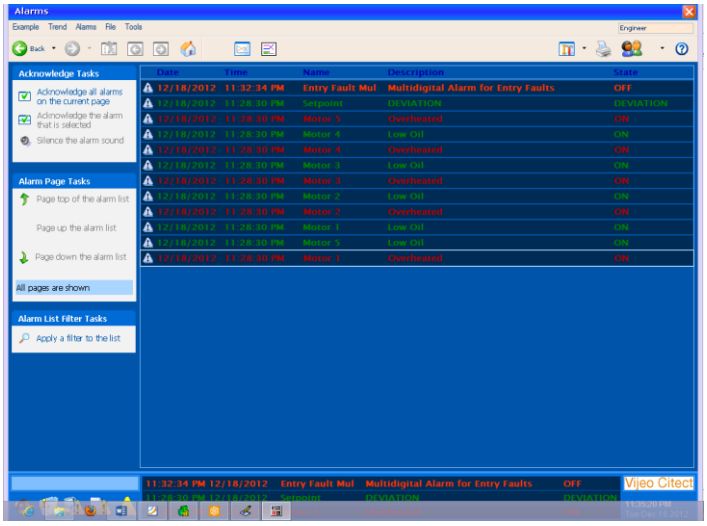
Tampilan grafis, bukan hanya angka, untuk mempermudah pengamatan seperti ditunjukkan pada Gambar 2.4.



Gambar 2.4 *Graphic displays*[4]

c. *Alarms*

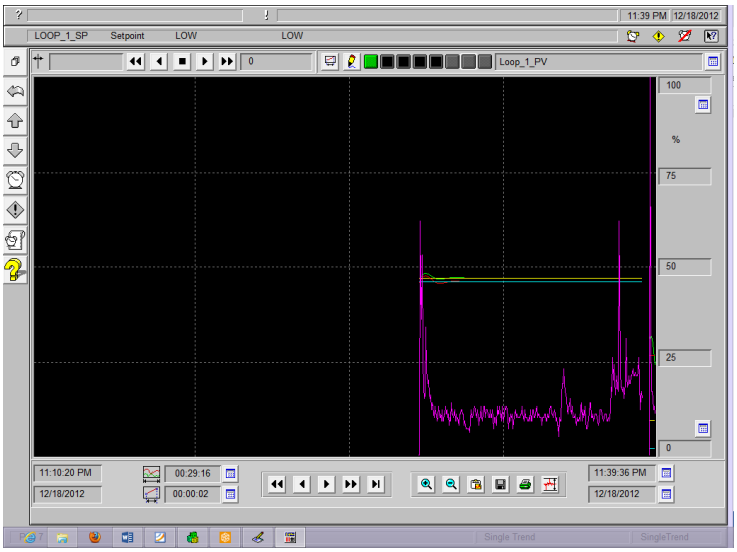
Alarm untuk memberi peringatan saat sistem dalam kondisi *abnormal*. Gambar 2.5 merupakan contoh *alarms* pada sistem SCADA.



Gambar 2.5 *Alarms*[4]

d. *Trends*

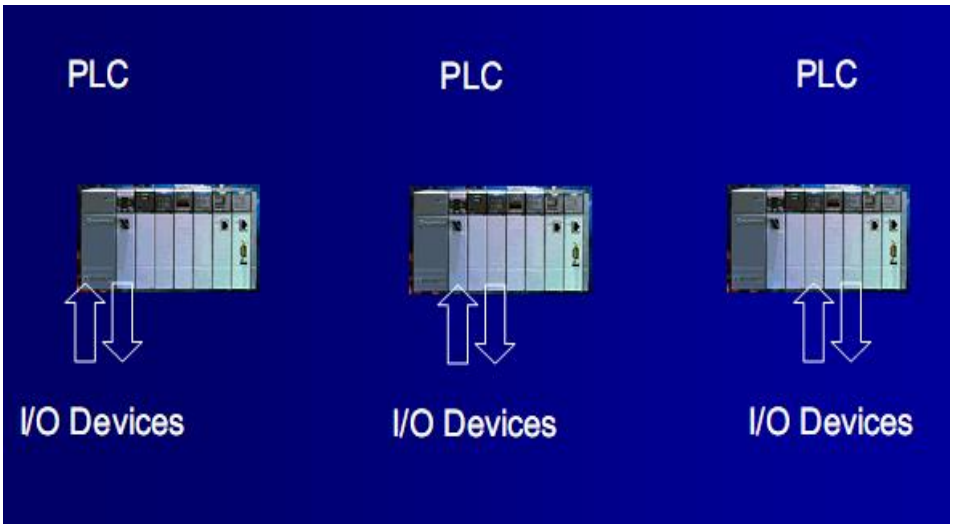
Trend ialah grafik garis yang menggambarkan kondisi atau status suatu *device*. Gambar 2.6 merupakan contoh *trends* pada sistem SCADA.



Gambar 2.6 *Trends*[4]

e. *RTU / PLC Interface*

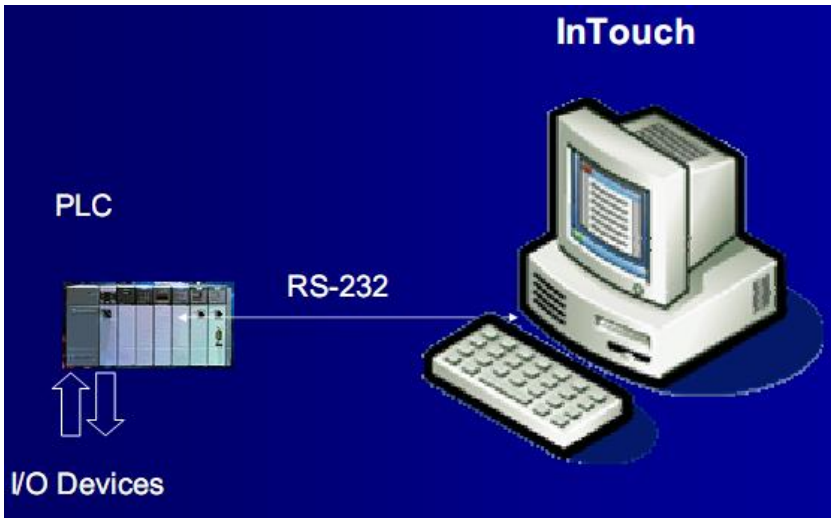
Bagian program yang menghubungkan PLC dengan perangkat lunak SCADA. Gambar 2.7 merupakan RTU atau PLC pada sistem SCADA.



Gambar 2.7 RTU atau PLC *interface*[4]

f. *Networking*

Program ini dapat berjalan dalam suatu jaringan, baik pada LAN maupun *internet*. Gambar 2.8 merupakan contoh komunikasi serial pada sistem SCADA.



Gambar 2.8 Komunikasi sistem SCADA[4]

g. *Scalability / Expandability*

Program dapat diperluas tanpa mengganggu program lama yang sudah ada.

h. *Access to data*

Program memiliki akses pada data tertentu yang diinginkan.

i. *Database*

Penyimpanan data ke dalam *database*

j. *Fault tolerance and redundancy*

Program memiliki toleransi tertentu terhadap kesalahan yang terjadi. Sistem SCADA juga harus bersifat *redundant*, dimana saat MTU utama *down* akan digantikan oleh MTU cadangan.

k. *Client/Server distributed processing*

Pemrosesan data bersifat *distributed*, dimana *server* maupun *client* memiliki bagian pemrosesan tersendiri.

2.2 PLC (*Programmable Logic Controller*)

Dalam dunia industri automasi, preses kontrol pada mulanya dilakukan dengan mengunakan *relay conventional*. *Coil relay* masing-masing dihubungkan dengan sensor-sensor, sedangkan *output contact*-nya dihubungkan pada bagian mesin yang akan dikendalikan . Proses kontrol menggunakan *relay* sangatlah rumit pada bagian *wiring* sehingga sangatlah sulit untuk memperbaiki sistem yang sudah ada. Selain itu, kontrol dengan menggunakan *relay* ini sangatlah terbatas. Oleh karena itu sekarang dibuat sebuah alat yang dapat menggantikan fungsi *relay* tersebut yaitu PLC.

PLC diperkenalkan pertama kali pada tahun 1969 oleh Richard E. Moerley yang merupakan pendiri Modicon (*Modular Digital Controller*) sekarang bagian dari *Gauld Electronics* untuk *General Motors Hydermatic Division*. Kemudian beberapa perusahaan seperti *Allan Breadly*, *General Electric*, *GEC*, *Siemens* dan *Westinghouse* memproduksi dengan harga standar dan kemampuan kerja tinggi. Pemasaran PLC dengan harga rendah didominasi oleh perusahaan Jepang seperti *Mitsubishi*, *Omron*, dan *Toshiba*[5].

PLC itu sendiri mempunyai definisi sebagai berikut[5]:

a. *Programmable*:

   Artinya dapat diprogram (diubah-ubah) sesuai dengan program yang diinginkan, kemudian menyimpan program tersebut pada memori.

b. *Logic*:

   Artinya dapat memproses *input* secara aritmatik atau mampu melakukan operasi matematika.

c. *Control*:

   Artinya dapat mengontrol dan mengatur suatu proses sehingga dapat menghasilkan *output* yang diinginkan.

Perancangan program dari PLC ini dapat dirancang dengan menggunakan *statement list* ataupun *ladder diagram*. Untuk memprogram suatu PLC, hal pertama yang harus dilakukan adalah program ditulis di PC dengan menggunakan proses khusus PLC, setelah itu program yang sudah selesai dibuat di-*download* ke PLC dengan menggunakan kabel serial yang merupakan sarana komunikasi antara PC dengan PLC.

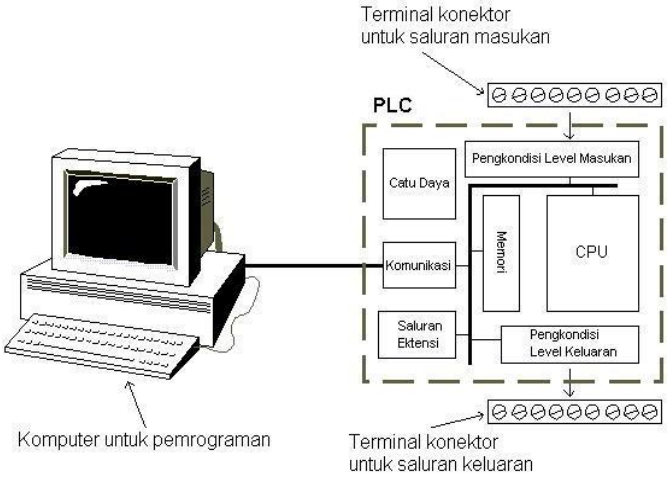
Setelah itu PLC yang sudah deprogram dapat bekerja sebagai pengontrol yang *independent*.

Karena harga dari PLC ini tergolong mahal, maka untuk menentukan jenis/tipe dari PLC yang akan digunakan sebaiknya memperhatikan kriteria-kriteria sebagai berikut[6]:

a. *Hardware*, yang meliputi jumlah *input/output* maksimum, tegangan operasi dari *input* dan *output*, indikator status *I/O*, jumlah *timer* dan *counter*, jumlah *flag*, ukuran memori, jenis memori, manipulasi *bit/word*, waktu siklus per 1x *statements*, *interface* ke PC, kondisi sekitar (suhu, kelembaban, pendinginan, getaran, *supply* arus), *housing* dan ukuran.
b. *Software*, yang meleputi bahasa pemrograman, kualitas dokumentasi, kualifikasi kemampuan operator yang dibutuhkan.

Penggunaan PLC memungkinkan kita untuk mengubah suatu sistem kontrol tanpa harus terlebih dahulu mengubah instalasi yang sudah ada sebelumnya. Jika ingin mengubah jalannya proses maka yang harus diubah hanyalah program yang ada dalam memori PLC saja (tanpa harus mengubah *hardware* yang telah jadi). Penggunaan PLC memudahkan pemakai dalam melakukan instalasi dan dapat mempersingkat waktu untuk mengubah jalanya proses kontrol. PLC dapat bekerja pada lingkungan industri dengan kondisi yang cukup berat, seperti temperature yang tinggi, pengaruh dari peralatan-peralatan lain yang berada disekitarnya.

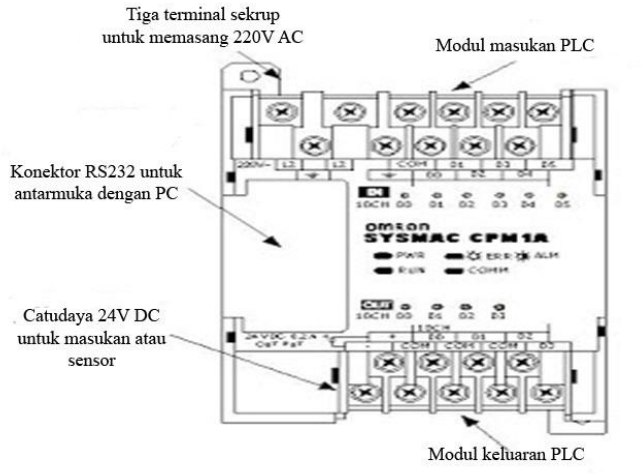
PLC yang diproduksi oleh berbagai perusahaan sistem kontrol terkemuka saat ini biasanya mempunyai ciri-ciri sendiri yang menawarkan keunggulan sistemnya, baik dari segi aplikasi (perangkat tambahan) maupun modul utama sistemnya. Meskipun demikian pada umumnya setiap PLC (sebagai computer pribadi yang cenderung mengalami standarisasi dan kompatibel satu sama lain) mengandung empat bagian (piranti) yaitu modul catu daya, modul *CPU*, modul perangkat lunak, dan modul *I/O*. Gambar 2.9 merupakan bagian-bagian utama yang terdapat pada PLC[6].



Gambar 2.9 Elemen dasar PLC[6]

2.2.1 Modul Catu Daya (*Power Supply*)

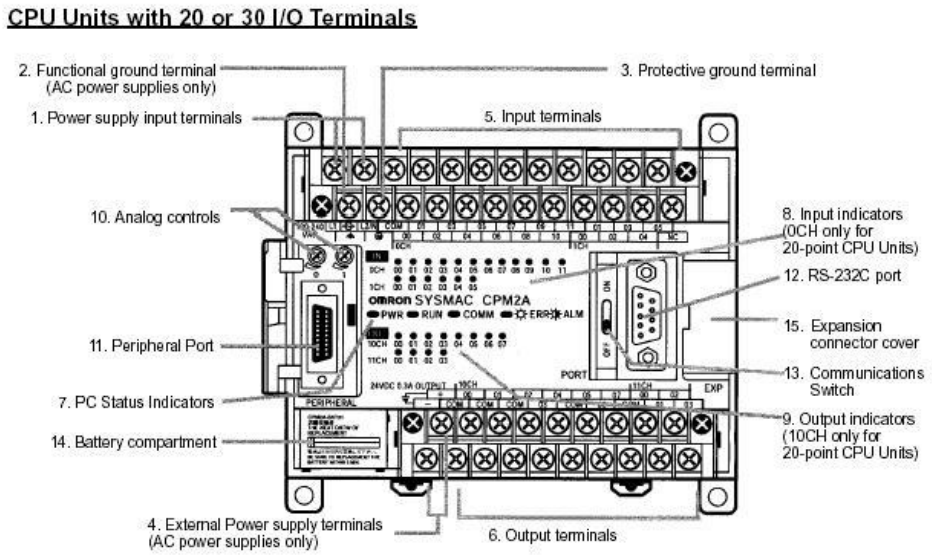
Power supply berguna sebagai penyedia daya bagi PLC, tegangan pada *power supply* bisa berupa tegangan AC (120/240 V) dan tegangan DC (24 V). Daya tersebut digunakan untuk berbagai modul PLC lainnya selain modul tambahan yang digunakan sebagai *memory backup* pada PLC. Jadi seandainya *power supply* mati, dengan adanya *memory backup* tersebut, maka isi memori akan tetap terjaga. PLC juga memiliki *power supply* (24 V DC) internal yang bisa digunakan untuk menyediakan daya bagi *input/output* PLC seperti yang ditunjukkan pada Gambar 2.10[6].



Gambar 2.10 Catu daya PLC[6]

2.2.2 Modul CPU

Modul CPU (*Central Processing Unit*) yang disebut juga modul *controller* atau *processor* terdiri dari dua bagian yaitu *processor* dan *memory* seperti yang ditunjukkan pada Gambar 2.11[6].



Gambar 2.11 CPU PLC omron[6]

a. *Processor*

Berfungsi untuk:

- Mengoperasikan dan mengkomunikasikan modul-modul PLC melalui bus-bus serial atau paralel yang ada.
- Mengeksekusi program kontrol.

b. *Memory*

Memory yang terdapat dalam PLC berfungsi untuk menyimpan program dan memberikan lokasi-lokasi dimana hasil-hasil perhitungan dapat disimpan didalamnya. PLC menggunakan peralatan memory semi konduktor seperti RAM (*Random Acces Memory*), ROM (*Read Only Memory*), dan PROM (*Programmable Read Only Memory*). RAM mempunyai waktu akses yang cepat dan program-program yang terdapat didalamnya dapat deprogram ulang sesuai dengan keinginan pemakainya. RAM disebut juga sebagai *volatile memory*, maksudnya program-

program yang tersimpan mudah hilang jika *supply* listrik padam. Dengan demikian untuk mengatasi *supply* listrik yang padam tersebut maka diberi *supply* cadangan daya listrik berupa baterai yang disimpan pada RAM. Baterai ini mempunyai jangka waktu kira-kira lima tahun sebelum harus diganti.

Berikut ini adalah jenis-jenis memori yang terdapat didalam PLC[6]:

1. *Spesial Relay*

   Special relay (SR) merupakan *relay* yang menghubungkan fungsi-fungsi khusus seperti *flag* (misalnya: instruksi penjumlahan terdapat kelebihan digit pada hasilnya (*carry flag*), kontrol bit PLC, informasi kondisi PLC, dan *system clock* (pulsa).

2. *Auxiliary Relay (AR)*

   Auxiliary relay terdiri dari *flags* dan *bit* untuk tujuan khusus. Dapat menunjukkan kondisi PLC yang disebabkan oleh kegagalan sumber tegangan, kondisi special I/O, kondisi *input/output* unit, kondisi CPU PLC, memori PLC dan lain-lain.

3. *Holding Relay*

   Holding relay (HR) dapat difungsikan untuk menyimpan data (*bit-bit* penting) karena tidak hilang walaupun sumber tegangan PLC mati.

4. *Link Relay*

   Link relay (LR) digunakan untuk *data link* pada PLC *link system. Link system* digunakan untuk tukar-menukar informasi antar dua PLC atau lebih dalam satu sistem kendali yang saling berhubungan satu dengan yang lainnya dengan menggunakan PLC minimum dua unit.

5. *Temporary Relay*

   Temporary relay (TR) berfungsi untuk menyimpan sementara kondisi logika program pada *ladder diagram* yang mempunyai titik percabangan khusus.

6. *Timer/Counter*

   Timer/counter (T/C) untuk mendefinisikan suatu waktu tunda /*time delay* (*timer*) ataupun untuk menghitung (*counter*). Untuk *timer* mempunyai orde 100 ms, ada yang mempunyai orde 10 ms yaitu TIMH (15). Untuk TIM 000 sampai dengan

TIM 015 dapat dioperasikan secara *interrupt* untuk mendapatkan waktu yang lebih presisi.

7. Data Memory

Data memory (DM) berfungsi untuk menyimpan data-data program karena isi DM tidak akan hilang (*reset*) walaupun sumber tegangan PLC mati. Macam-macam DM adalah sebagai berikut:

a. DM *read/write*

Pada DM *read/write* data-data program dapat dihapus dan ditulis oleh program yang dibuat, sehingga sangat berguna untuk manipulasi data program.

b. DM *special I/O unit*

DM *special I/O* berfungsi untuk menyimpan dan mengolah hasil dari *special I/O unit*, mengatur dan mendefinisikan sistem kerja *special I/O unit*.

c. DM *history Log*

Pada DM *history log* disimpan informasi-informasi penting pada saat PLC terjadi kegagalan sistem operasionalnya. Pesan-pesan kesalahan sistem PLC yang di simpan berupa kode-kode angka tertentu.

d. DM *link test area*

DM *link test area* berfungsi untuk menyimpan informasi-informasi yang menunjukan status dari system link PLC.

e. DM *setup*

DM *setup* berfungsi untuk kondisi *default* (kondisi kerja saat PLC aktif). Pada DM inilah kemampuan kerja suatu PLC didefinisikan untuk pertama kalinya sebelum PLC tersebut diprogram dan dioperasikan pada suatu sistem kontrol. Tentu saja *setup* PLC tersebut disesuaikan dengan sistem kontrol yang bersangkutan.

8. Upper Memory

Upper memory (UM) berfungsi untuk menyimpan dan menjalankan program. Kapasitas tergantung dari pada masing-masing tipe PLC yang dipakai.

Semua memori (selain DM dan UM) dapat dibayangkan sebagai *relay* yang mempunyai koil, kontak NO (*Normally Open*) dan NC (*Normally Close*). *Timer* dan

Counter juga dapat dibayangkan seperti *timer* danc*Counter* pada umumnya dan mempunyai kontak NO dan NC. DM tidak mempunyai kontak, hanya ada *channel/word* saja. DM dapat difungsikan untuk menyimpan data-data penting yang tidak boleh hilang waktu sumber tegangan mati atau memanipulasi program. Pada penelitian Tugas Akhir ini jenis PLC yang akan digunakan adalah PLC Omron Sysmac tipe CP-1L yang memiliki struktur memori seperti yang ditunjukkan pada Tabel 2.1[7].

Tabel 2.1 Struktur memori PLC omron CP-IL[7]

Area			Size	Range	Task usage	Allocation	Bit access	Word access	Access		Change from CX-Programmer	Forcing bit status
									Read	Write		
CIO Area	I/O Area	Input Area	1,600 bits (100 words)	CIO 0 to CIO 99	Shared by all tasks	CP1L CPU Units and CP-series Expansion Units or Expansion I/O Units	OK	OK	OK	OK	OK	OK
		Output Area	1,600 bits (100 words)	CIO 100 to CIO 199			OK	OK	OK	OK	OK	OK
	1:1 Link Area		1,024 bits (64 words)	CIO 3000 to CIO 3063		1:1 Links	OK	OK	OK	OK	OK	OK
	Serial PLC Link Area		1,440 bits (90 words)	CIO 3100 to CIO 3189		Serial PLC Links	OK	OK	OK	OK	OK	OK
	Work Area		14,400 bits (900 words)	CIO 3800 to CIO 6143		---	OK	OK	OK	OK	OK	OK
Work Area			8,192 bits (512 words)	W000 to W511		---	OK	OK	OK	OK	OK	OK
Holding Area			8,192 bits (512 words)	H000 to H511 (Note 6)		---	OK	OK	OK	OK	OK	OK
Auxiliary Area			15,360 bits (960 words)	A000 to A959		---	OK	---	OK	Note 1	Note 1	No
TR Area			16 bits	TR0 to TR15		---	OK	OK	OK	OK	No	No
Data Memory Area			32,768 words	D00000 to D32767 (Note 7)		---	No (Note 2)	OK	OK	OK	OK	No
Timer Completion Flags			4,096 bits	T0000 to T4095		---	OK	---	OK	OK	OK	OK
Counter Completion Flags			4,096 bits	C0000 to C4095		---	OK	---	OK	OK	OK	OK
Timer PVs			4,096 words	T0000 to T4095		---	---	OK	OK	OK	OK	No (Note 4)
Counter PVs			4,096 words	C0000 to C4095		---	---	OK	OK	OK	OK	No (Note 5)
Task Flag Area			32 bits	TK0 to TK31		---	OK	---	OK	No	No	No
Index Registers			16 registers	IR0 to IR15	Function separately in each task (Note 3)	---	OK	OK	Indirect addressing only	Specific instructions only	No	No
Data Registers			16 registers	DR0 to DR15		---	No	OK	OK	OK	No	No

Pada PLC tertentu terkadang dijumpai pula adanya beberapa *processor* dalam satu modul sekaligus yang ditujukan untuk mendukung kehandalan sistem. Beberapa *processor* tersebut bekerja sama dengan suatu prosedur tertentu untuk meningkatkan kinerja pengendalian.

2.2.3 Modul *I/O*

Modul *I/O* merupakan modul masukan (*input*) dan modul keluaran (*output*) yang bertugas mengatur hubungan PLC dengan piranti eksternal atau *peripheral* yang bisa berupa suatu *computer host*, saklar-saklar, unit penggerak motor, dan berbagai macam sumber sinyal yang terdapat dalam *plant*[6].

a. Modul *Input*

   Modul *input* berfungsi untuk menerima sinyal dari unit pengindera *peripheral*, dan memberikan pengaturan sinyal, terminasi, isolasi, maupun indikator keadaan sinyal *input*. Sinyal-sinyal dari piranti *peripheral* akan di-*scan* dan keadaannya akan dikomunikasikan melalui modul antarmuka dalam PLC.

   Beberapa jenis tegangan masukan pada PLC Omron CP-1L:

   - Tegangan masukan DC (24 V).
   - Tegangan AC (120/240 V).
   - Masukan TTL (5 V).

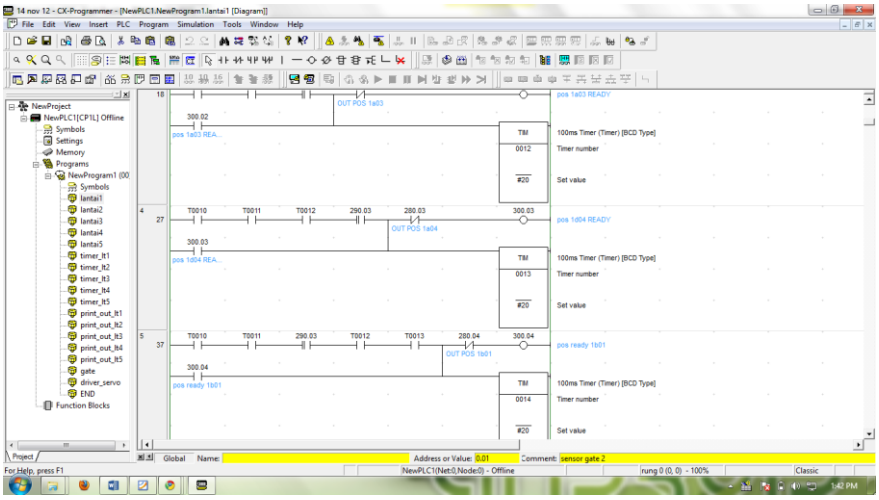
b. Modul *Output*

   Modul output mengaktivasi berbagai macam piranti seperti *actuator hydrolic*, *pneumatic*, *solenoid*, *starter motor*, dan tampilan status titik-titik *peripheral* yang terhubung dalam sistem. Fungsi modul *output* lainnya mencakup *conditioning*, terminasi dan juga pengisolasian sinyal-sinyal digital dan analog yang relevan, berdasarkan sifat PLC sendiri yang merupakan piranti digital. Tegangan keluaran pada PLC Omron CP-1L adalah tegangan DC (24 V).

2.2.4 Modul Perangkat Lunak PLC

Perangkat lunak merupakan modul yang pasti dimiliki oleh PLC. PLC mampu mengenali berbagai macam bahasa pemrograman / perangkat lunak seperti *state language*, bahasa pemrograman C, dan yang paling popular digunakan adalah RLL

(*relay ladder logic*). Sistem yang akan digunakan untuk mengontrol *plant*, dibuat dalam bahasa pemrograman yang *compatible* dengan PLC yang digunakan. Semua instruksi yang telah ditulis disimpan dalam memori PLC, dan akan dieksekusi oleh modul CPU. Penulisan program pada PLC dapat dilakukan pada keadaan *on line* maupun *off line*. Pada perangkat lunak PLC omron yang digunakan untuk menulis program adalah CX-Programmer seperti ditunjukkan pada Gambar 2.12[8].



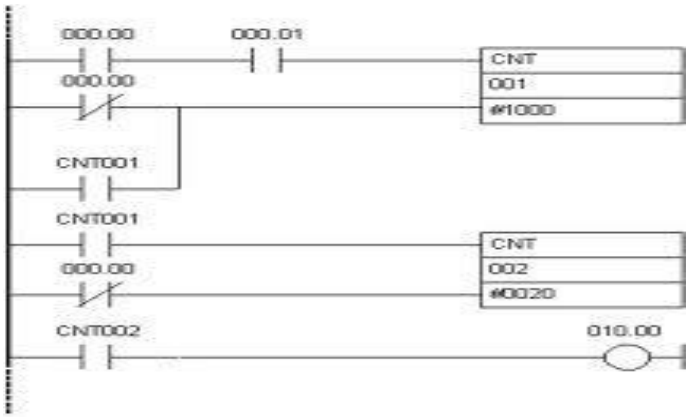
Gambar 2.12 Tampilan *software* CX-Programmer[8]

2.2.5 Bahasa dan Logika Pemrograman PLC

PLC memiliki bermacam-macam bahasa program yang ditetapkan oleh (*International Electrotecnic Commission*) IEC 61131-3, ada 5 bahasa pemrograman PLC yaitu[9]:

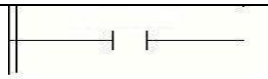
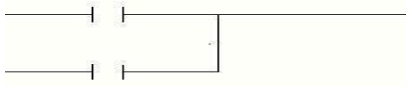
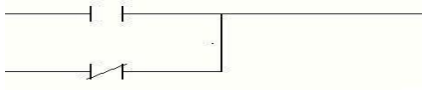
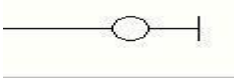
a. *Ladder Diagram* (LD)

Adalah bahasa pemrograman yang dibuat dari persamaan fungsi logika dan fungsi-fungsi lain berupa pemrosesan data atau fungsi waktu dan pencacahan. *Ladder diagram* terdiri dari susunan kontak-kontak dalam satu kelompok perintah secara horizontal dari kiri ke kanan, dan terdiri dari banyak kelompok perintah secara vertikal. *Ladder diagram* termasuk bahasa pemrograman PLC yang paling banyak digunakan. Gambar 2.13 merupakan contoh program *ladder diagram*[9].



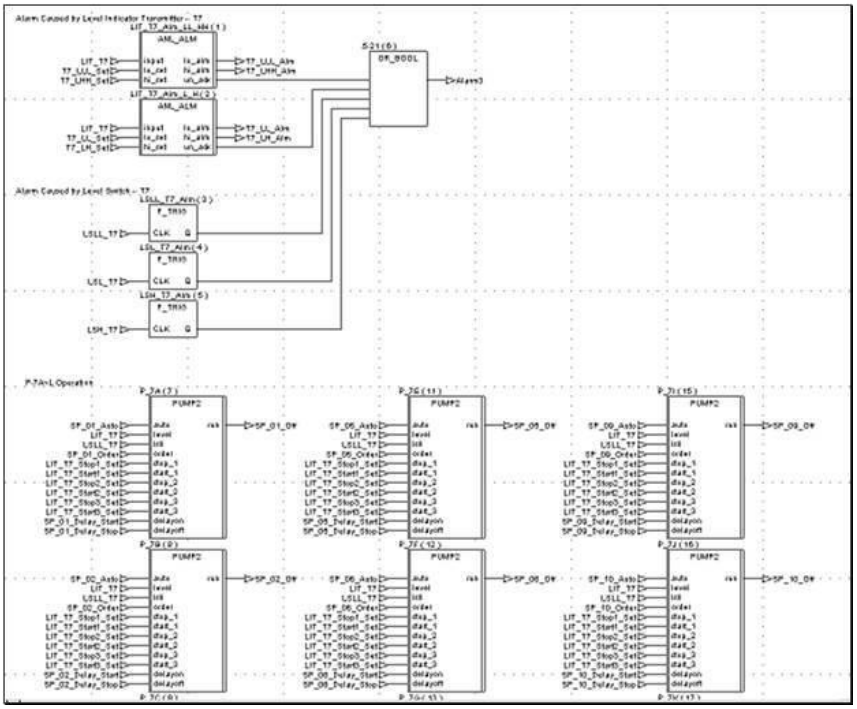
Gambar 2.13 *Ladder diagram*[9]

Tabel 2.2 Simbol-simbol *ladder diagram* PLC[10]

Nama	Simbol	Fungsi
Load		*Start* pada *normally open input*
Load Not		*Start* pada *normally close input*
And		Menghubungka dua atau lebih *input* dalam bentuk *normally open* secara seri
And Not		Menghubungka dua atau lebih *input* dalam bentuk *normally close* secara seri
Or		Menghubungka dua atau lebih *input* dalam bentuk *normally open* secara paralel
Or Not		Menghubungka dua atau lebih *input* dalam bentuk *normally open* dan *normally close* secara paralel
Output		Untuk menyalakan *Output*

b. *Function Block Diagram* (FBD)

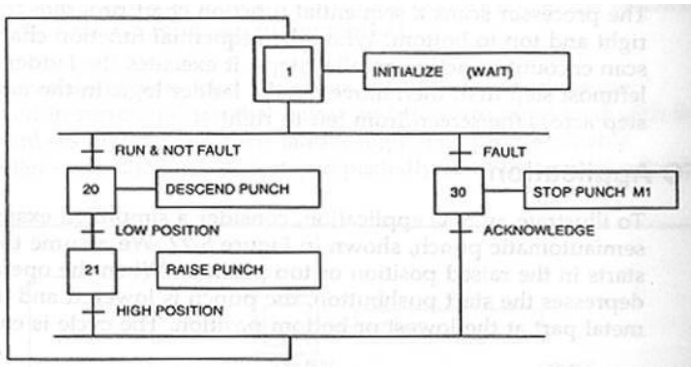
Merupakan penggunaan blok-blok fungsi standar maupun buatan pengguna sendiri yang digunakan untuk pemrogram PLC. FDB jarang digunakan karena mempunyai logika pemrograman yang rumit dan komplek. Gambar 2.14 merupakan contoh pemrograman FDB.



Gambar 2.14 *Function block*[9]

c. *Sequential Function Chart* (SFC)

SFC menggambarkan secara grafis aksi sekuensial dari sebuah kontrol proses. SFC terdiri dari *step* yang terhubung dengan blok aksi dan transisi. Masing-masing *step* merepresentasikan keadaan (*state*) tertentu dari sebuah sistem yang dikendalikan. Sebuah transisi berkenaan dengan sebuah kondisi, dimana jika benar akan menyebabkan *step* sebelumnya tidak aktif dan *step* selanjutnya aktif. *Step-step* yang terhubung ke blok aksi akan menjalankan aksi kontrol tertentu. Masing-masing elemen SFC dapat diprogram dengan sembarang bahasa IEC, termasuk SFC itu sendiri. Gambar 2.15 merupakan contoh pemrograman PLC menggunakan SFC.



Gambar 2.15 *Sequential function chart*[9]

d. *Instruction List* (IL)

PLC diprogram dengan serangkaian instruksi atau perintah dan tiap instruksi harus dimulai pada baris baru. Bahasa pemrograman IL sangat jarang digunakan dalam dunia industri karena kurang familiar. Gambar 2.16 merupakan contoh bahasa pemrograman IL.

```
Start : LD      IX1     (* load input IX1, start pushbutton *)
        ANDN    MX5     (* AND with NOT of MX5)
        ST      QX2     (* store output QX2 to start motor *)
```

Gambar 2.16 *Instruction list*[9]

e. *Structured Text* (ST)

Pemrograman PLC dengan menggunakan bahasa tingkat tinggi seperti PASCAL. Gambar 2.17 merupakan contoh bahasa pemrograman ST.

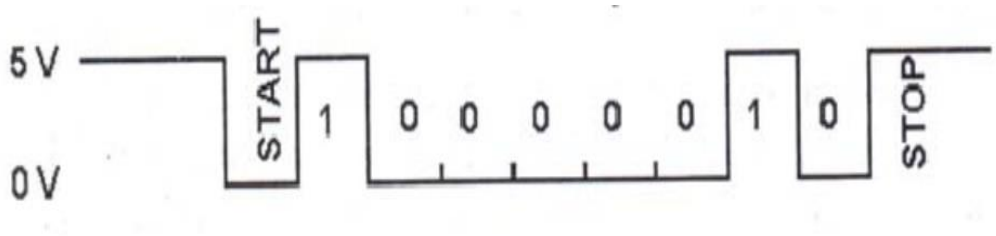
```
imax:=max_ite;
cond:=X12;
if not(cond(*alarm*))
then return;
end_if;

for i:=1 to max_ite do
if i<>2 then
SPcall();
end_if;
end_for;
```

Gambar 2.17 *Structured text*[9]

2.3 Komunikasi Data Serial

Komunikasi data secara serial dibedakan menjadi dua, yaitu komunikasi data serial secara sinkron dan komunikasi data secara asinkron. Pada komunikasi data serial sinkron, *clock* dikirimkan bersama-sama dengan data serial, sedangkan komunikasi data asinkron *clock* tidak dikirimkan bersama data serial, melainkan dibangkitkan secara sendiri-sendiri baik pada sisi pengirim (*transmitter*) maupun pada sisi penerima (*receiver*). Pada PC *compatibel port* serialnya termasuk jenis asinkron.

Kecepatan pengiriman data (*baud rate*) dan *fase clock* pada sisi *transmitter* dan *receiver* harus sinkron. Untuk itu diperlukan sinkronisasi antara *transmitter* dan *receiver*. Hal ini dilakukan oleh *bit Start* dan *bit Stop*. Ketika saluran transmisi dalam keadaan *idle*, *output Universal Asynchronous Receiver Trasmitter* (UART) adalah dalam keadaan logika '1'. Ketika *transmitter* ingin mengirimkan data, *output* UART akan diset lebih dahulu ke logika '0' untuk waktu 1 bit. Sinyal ini pada *receiver* akan dikenali sebagai sinyal '*Start*' yang digunakan untuk mensinkronkan *fase clock*-nya sehingga sinkron dengan *fase clock transmitter*. Selanjutnya data akan dikirimkan secara serial dari bit paling rendah (bit 0) sampai bit tertinggi bit (1). Selanjutnya akan dikirim sinyal '*Stop*' sebagai akhir dari pengiriman data serial. Cara pemberian kode yang disalurkan tidak ditetapkan secara pasti. Gambar 2.18 adalah contoh pengiriman huruf 'A' dalam format ASCII (41 Heksa atau 10000001 biner) tanpa bit paritas[19].



Gambar 2.18 Contoh pengiriman huruf "A" tanpa paritas bit[11]

Kecepatan transmisi (*baud rate*) dapat dipilih bebas dalam rentang tertentu. *Baud rate* yang umum dipakai adalah 110, 135, 150, 300, 600,1200, 2400 dan 9600 (bit/detik). Dalam komunikasi data serial, *baud rate* dari kedua alat yang berhubungan harus diatur pada kecepatan yang sama.

2.3.1 Komunikasi Data Serial RS 232

RS 232 merupakan *interface* paling umum yang digunakan pada komunikasi serial. RS 232 yang dikenalkan pada tahun 1962 ini sering digunakan diberbagai industri dan otomasi. Spesifikasi transmisi data dari *transmiter* ke *receiver* rata-rata lamban dan jarak tranmisinya pendek. RS 232 populer karena harganya yang murah, dan dapat menggunakan kabel yang lebih panjang dibandingkan komunikasi menggunakan parallel. *Channel-channel independent* dibuat untuk komunikasi dua arah (*full-duplex*). Sinyal RS 232 diwakili oleh tegangan sistem umum, dan menspesifikasikan protocol komunikasi, dimana dapat bekerja baik dalam komunikasi *point to point* pada rata-rata transmisi data rendah. Suatu perangkat dapat menggunakan *port* yang ada pada komputer, atau perlu tambahan *port*, namun kebanyakan PC mempunyai *interface* RS 232, dan *port* RS 232 pada PC merupakan *single device*. Sinyal RS 232 membutuhkan *ground* antara PC dan peralatan yang terhubung. Sedangakn jarak kabel harus dibatasi 1 sampai 200 ft pada data *asynchronous* dan sekitar 50 ft dengan data *synchronous*. Data *synchronous* mempunyai *transmite* dan *receive* yang membatasi jarak maksimum pada saat menggunakn suatu jalur data *synchronous*. *Port* RS 232 didesain untuk berkomunikasi dengan peralatan yang mendukung satu *transmiter* dan satu *receiver*. Pada umumnya serial RS 232 digunakan untuk komunikasi dua arah. Gambar 2.19 merupakan kabel konektor RS 232[12].

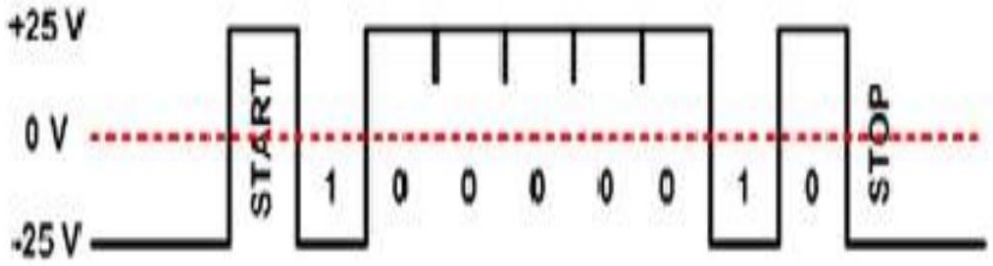
Gambar 2.19 Kabel konektor RS 232[12]

Duplex adalah suatu metode pengoperasian rangkaian komunikasi antara dua peralatan. *Full-duplex*, memungkinkan suatu unit untuk mengirim dan menerima informasi secara bersamaan. *Half-duplex*, memungkinkan suatu unit untuk mengirim informasi pada suatu saat meskipun sambungan mungkin mampu untuk melakukan transmisi dua arah, namun komunikasi terjadi secara bergantian.

2.3.2 Karakteristik Sinyal Port Serial

Standar sinyal komunikasi serial yang banyak digunakan adalah RS 232 yang dikembangkan oleh *Electronic Industry Association and the Telecommunication Industry Association* (EIA/TIA) yang pertama kali dipublikasikan pada tahu 1962. Hal ini terjadi jauh sebelum IC TTL popular, sehingga sinyal ini tidak ada hubunganya sama sekali dengan level tegangan IC TTL. Sinyal standar serial ini hanya menyangkut komunikasi data antara komputer (*Data Terminal Equipment* – DTE) dengan alat-alat perlengkapan komputer (*Data Circuit-Terminating equipment* – DCE) saja. Sinyal standar serial RS 232 memiliki ketentuan level tegangan sebagai berikut[13]:

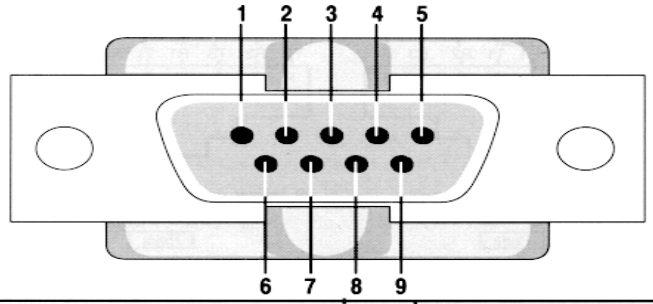
a. *Logic* "1" disebut *mark* terletak antara -3 volt hingga -25 volt.
b. *Logic* "0" disebut *space* terletak antara +3 volt hingga +25 volt.
c. Daerah tegangan antara -3 volt hingga +3 volt adalah *invalid level*, yaitu daerah tegangan yang tidak memiliki *level logic* pasti, sehingga harus dihindari. Demikian juga, *level* tegangan lebih negatif dari -25 volt atau lebih positif +25 volt juga harus dihindari karena *level* tegangan tersebut dapat merusak *line driver* pada saluran RS 232. Gambar 2.20 merupakan contoh level tegangan pada pengiriman huruf A ASCII (41 heksa dan 1000001 biner).



Gambar 2.20 Level tegangan RS 232 pada pengiriman huruf "A" [13]

2.3.3 Konfigurasi Port Serial

Konektor DB-9 pada bagian belakang computer adalah port serial RS 232 yang biasa dinamai dengan COM1 dan COM2. Gambar 2.21 menunjukkan konektor *female port serial* DB-9[13].



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Gambar 2.21 Konektor *female port serial* DB-9[13]

Keterangan mengenai fungsi saluran RS 232 pada konektor *female* dan konfigurasi pin serta nama sinyal konektor serial DB-9 dapat dilihat pada keterangan dan Tabel 2.3 dibawah ini:

a. *Received line signal detect*, dengan saluran ini DCE memberitahukan ke DTE bahwa pada terminal masukan ada data masuk.
b. *Receive Data*, digunakan DTE menerima data dari DCE.
c. *Transmit data*, digunakan DTE mengirimkan data ke DCE.
d. *Data Terminal Ready*, DTE memberitahukan kesiapan terminalnya.
e. *Signal ground*, saluran *ground*.
f. DCE *read*y, sinyal aktif pada saluran ini menunjukkan bahwa DCE sudah siap.
g. *Request to send*, dengan saluran ini DCE diminta mengirim data oleh DTE.
h. *Clear to send*, dengan saluran ini DCE memberitahukan bahwa DTE boleh mulai mengirim data.
i. *Ring indicator*, pada saluran ini DCE memberitahukan ke DTE bahwa sebuah stasiun menghendaki hubungan dengannya.

Tabel 2.3 Konfigurasi pin dan nama sinyal konektor serial DB-9[13]

Pin	Nama	Keterangan	Arah	Fungsi
1	DCD	*Data Carrier Detect*	Masuk	Saat mendeteksi suatu "*carrier*" maka sinyal ini aktif.
2	RxD	*Receive Data*	Masuk	Untuk penerimaan data serial.
3	TxD	*Transmit data*	Keluar	Untuk pengiriman data serial.
4	DTR	*Data Terminal Ready*	Keluar	Siap/tidak terjadi hubungan komunikasi.
6	DSR	*Data Set Ready*	Masuk	Siap/tidak melakukan pertukaran data.
7	RST	*Request to Send*	Keluar	Informasi siap/tidak pertukaran data.
8	CTS	*Clear to Send*	Masuk	Siap/tidak melakukan pertukaran data.
9	RI	*Ring Indicator*	Masuk	Aktif jika mendeteksi sinyal dering.

2.3.4 Transmisi Data Pada RS 232

Komunikasi pada RS 232 dengan PC adalah komunikasi asinkron. Pada komunikasi serial asinkron tidak diperlukan *clock* karena data dikirim dengan kecepatan tertentu yang sama baik pada pengirim atau penerima. Masing-masing data disinkronkan menggunakan *clock* internal pada tiap-tiap sisinya. Kecepatan transmisi (*baud rate*) dapat dipilih bebas dalam rentang tertentu. *Baud rate* yang umum dipakai adalah 110, 135, 150, 300, 600, 1200, 2400, dan 9600 (bit per detik). Dalam komunikasi data serial, *baud rate* dari kedua alat yang berhubungan harus diatur pada kecepatan yang sama. Selanjutnya harus ditentukan panjang data (6,7 atau 8 bit), paritas (genap, ganjil, atau tanpa paritas), dan jumlah bit "*stop*" (1, 1.5, atau 2 bit). Bentuk data *Asynchronuos*[13]:

a. *Logic* "1" disebut *mark* terletak antara -3 volt hingga -25 volt.
b. *Logic* "0" disebut *space* terletak antara +3 volt hingga +25 volt.
c. T (*bit time*): lama pengiriman 1 bit data.
d. *Start bit*: bit sebagai penanda mulainya pengiriman data (selalu *low*).

e. *Data bit*: bit-bit data

f. *Parity bit*: bit untuk mengecek kesalahan pengiriman data.

- *Odd parity* (bernilai 0) = jumlah data bit + *parity bit* = ganjil.
- *Even parity* (berbilai 1) = jumlah data bit + *parity bit* = genap.

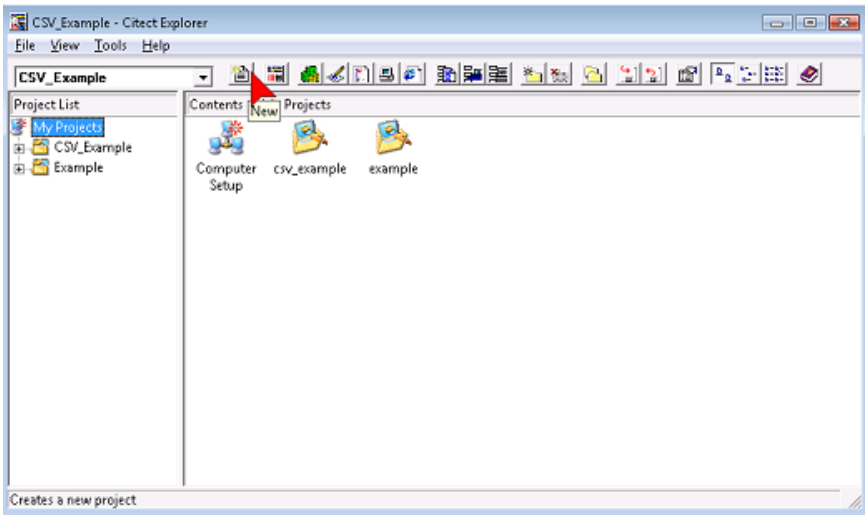
g. *Stop bit* atau *stop internal*: panjangnya waktu sebelum interval berikutnya, untuk menjamin kondisi kembali ke *MARK* (sebelum *start bit* karakter berikutnya memberi kondisi *SPACE*), dan besarnya adalah 1,/1.5,/2 kali dari T (*bit time*).

2.4 Vijeo Citect

Vijeo Citect ini merupakan salah satu software yang digunakan untuk membangun sistem SCADA, yang kegunaanya dapat memonitoring juga mengontrol sistem bahkan membuat *database* sebuah sistem. Secara umum *software* Vijeo Citect terdiri dari empat bagian utama yaitu[14]:

a. *Citect Explorer*

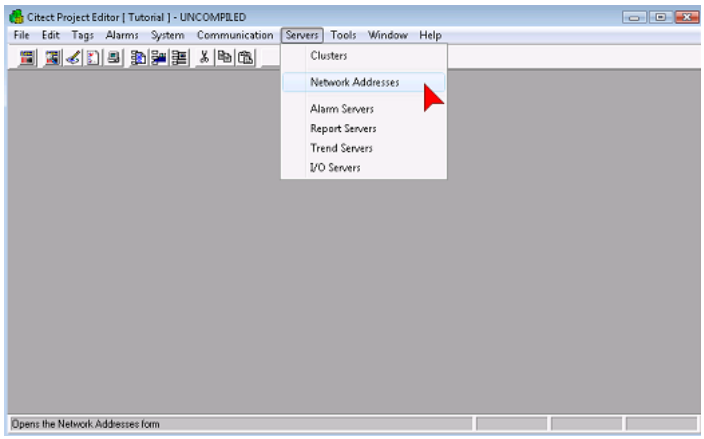
Berfungsi membuat *project* baru, memilih dan mengatur *project*, *backup* dan *restore project* dan menjalankan aplikasi lainya seperti *cicode editor*. Pada *citect explorer* juga terdapat semua *database* dari sistem yang telah dibuat. Gambar 2.22 merupakan tampilan dari *citect explorer*.



Gambar 2.22 *Citect explorer*[14]

b. *Citect Project Editor*

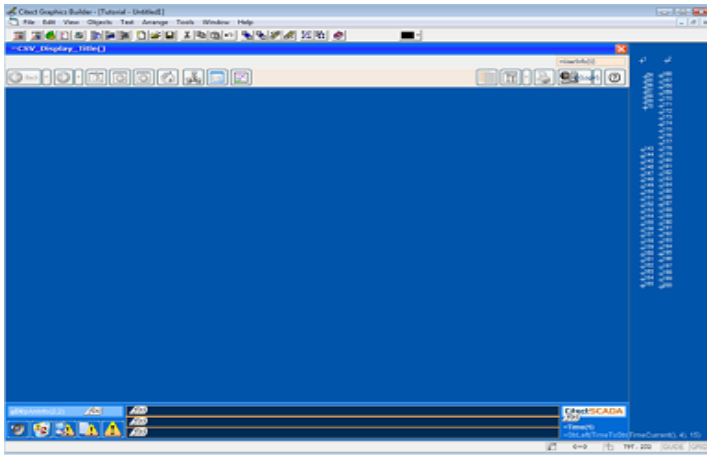
Berfungsi membuat dan mengatur *database* Vijeo Citect yang berisi informasi dari *project* Vijeo Citect. Pada *citect project editor* tidak dapat digunakan untuk membuat graphics. Gambar 2.23 merupakan tampilan dari *citect project editor*.



Gambar 2.23 *Citect project editor*[14]

c. *Citect Graphics Builder*

Befungsi untuk membuat gambar atau tampilan sistem yang akan dirancang, masukan *variable tags* atau fungsi untuk menjalankan perintah mengirim atau menerima data, dan untuk membuat *object display runtime*. Gambar 2.24 merupakan tampilan dari *citect graphics builder*.



Gambar 2.24 *Citect graphics builder*[14]

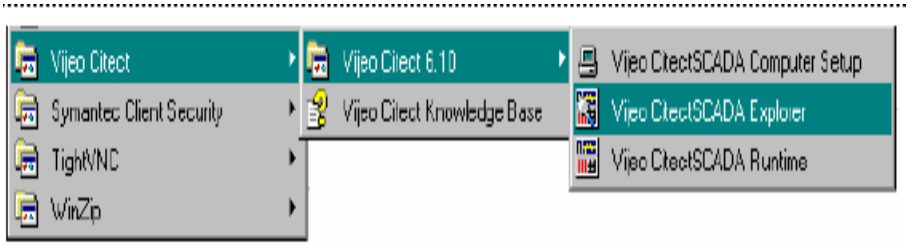
d. *Citect Runtime*

Untuk menjalankan semua *project* yang telah dibuat dan untuk membuka komunikasi serial *port* antara PLC dengan PC sehingga sistem SCADA tersebut dapat dioperasikan oleh operator. Gambar 2.25 merupakan tampilan dari *citect runtime*.

Gambar 2.25 *Citect runtime*[14]

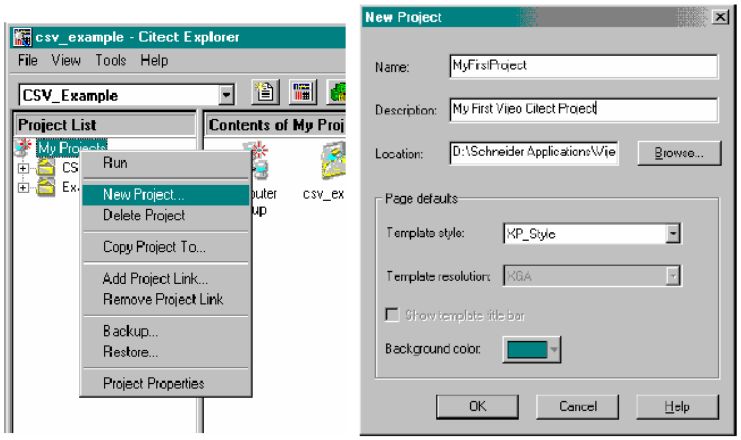
2.4.1 Membuat *Project* Baru

Untuk membuat suatu project baru dengan Vijeo Citect v7.1 yaitu dengan membuka *citect explorer* seperti ditunjukkan Gambar 2.26.



Gambar 2.26 Menjalankan Vijeo Citect[14]

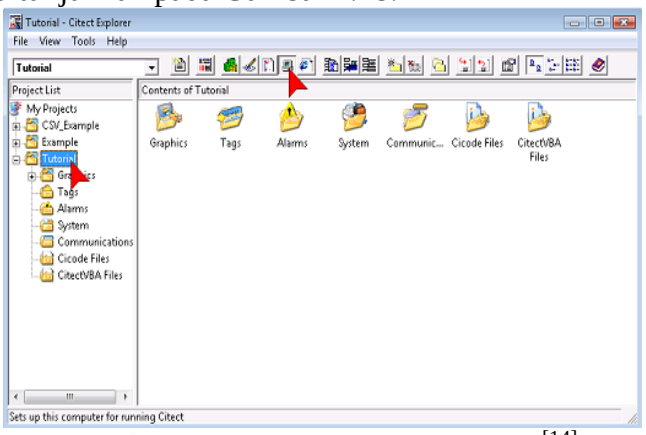
Kemudian klik kanan *my project* pilih *new project* maka akan muncul tampilan seperti Gambar 2.27. Pada gambar tersebut terdapat kolom-kolom yang harus kita isi sesuai dengan nama *project* yang dinginkan. Setelah selesai, klik *OK* maka Vijeo Citect akan membuat *database* yang berisi semua data-data yang akan digunakan oleh *project* yang dibuat tersebut.



Gambar 2.27 *New project*[14]

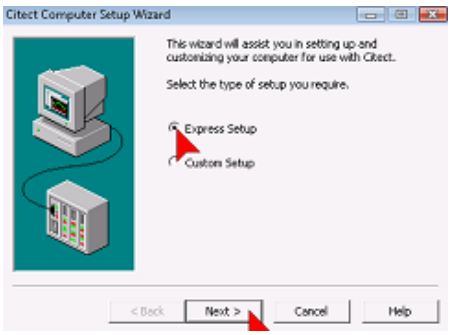
2.4.2 Membuat Konfigurasi *Computer Setup Wizard*

Tujuan membuat pengaturan komputer adalah untuk menjalankan *project* yang telah dibuat agar bisa dijalankan di komputer tersebut. Sebelum melakukan pengaturan computer untukmenjalankan *project* Vijeo citect, pastikan dahulu bahwa *project* sudah di-*compile* seperti ditunjukkan pada Gambar 2.28.



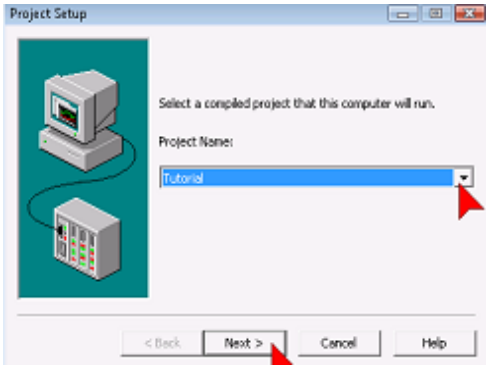
Gambar 2.28 *Computer setup wizard*[14]

a. Untuk melakukan pengaturan komputer, klik tombol *computer setup*, maka akan keluar tampilan berikut. Ikutilah pilihan yang ada pada Gambar 2.29, lalu klik *next*.



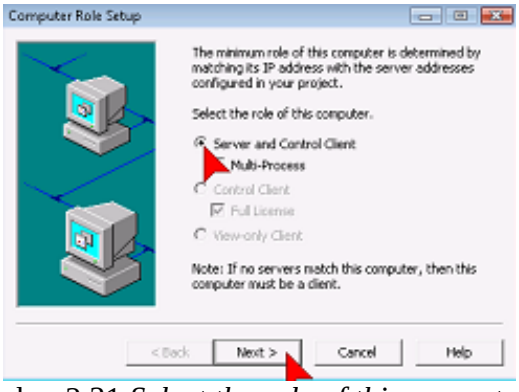
Gambar 2.29 Tipe *computer setup wizard*[14]

b. Akan keluar tampilan seperti Gambar 2.30 untuk memilih *project* yang akan dijalankan, pilih *project* kemudian klik *next*.



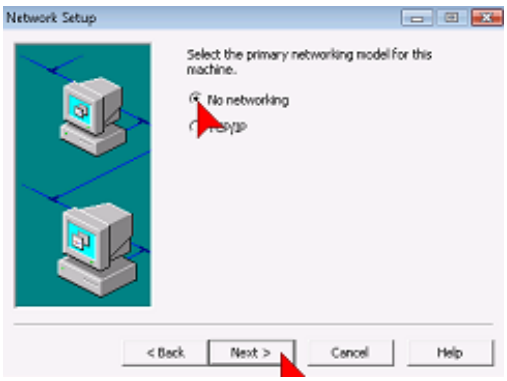
Gambar 2.30 *Select a compiled project*[14]

c. Setelah keluar seperti Gambar 2.31 kemudian pilih *server and display client*, kemudian klik *next*.



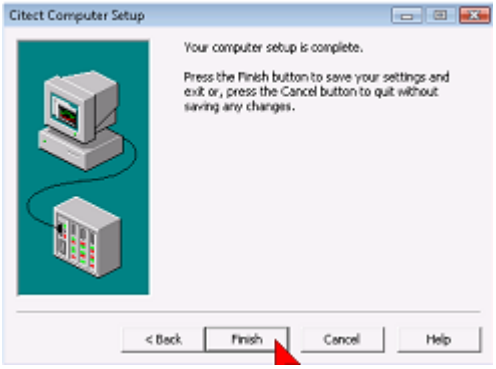
Gambar 2.31 *Select the role of this computer*[14]

d. Maka akan keluar seperti Gambar 2.32 selanjutnya pilih *no networking*, kemudian klik *next*.



Gambar 2.32 *Select the primary networking*[14]

e. Pilih *Finish*, untuk menyimpan semua pengaturan yang telah dilakukan dan keluar dari *citect computer setup wizard* seperti yang ditunjukkan pada Gambar 2.33.



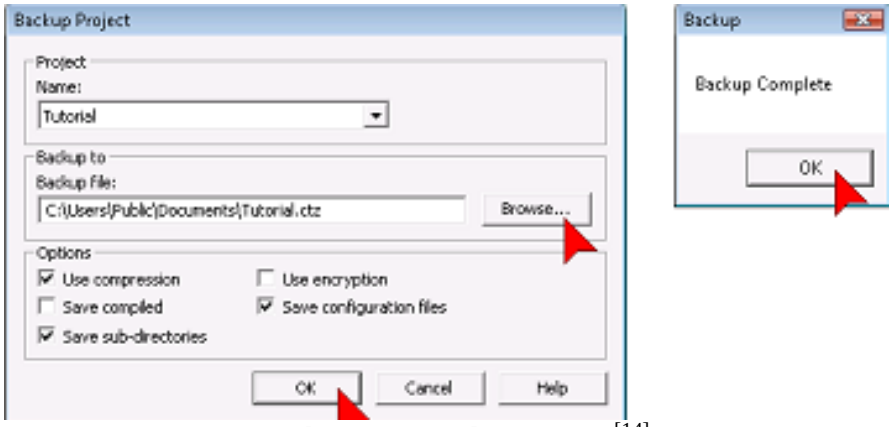
Gambar 2.33 *Computer setup is complete*[14]

2.4.3 Membuat *Backup* dan *Restore Project.*

a. *Backup Project*

Langkah-langkah membuat *backup project* yaitu pada *citect explorer*, pilih *project* yang akan di *backup* pada *project list*. Klik kanan pada *project* yang dipilih tersebut, pilih *backup*, akan keluar tampilan seperti pada Gambar 2.34. Pada gambar tersebut ada piihan nama *project* yang akan di *backup*. Selain itu ada pilihan folder untuk menyimpan *project* yang akan di *backup* dengan menggunakan

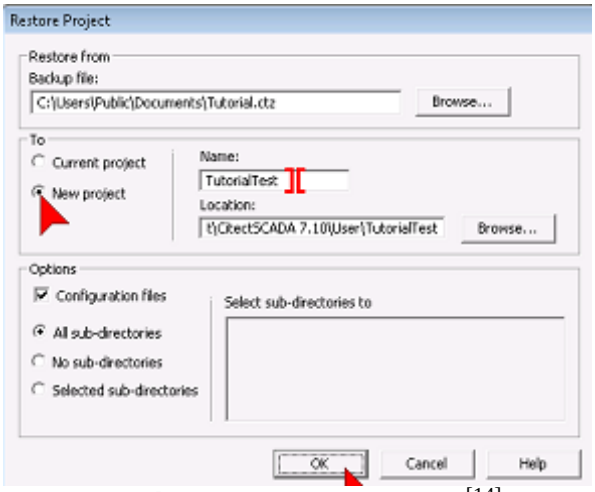
tombol *browse* untuk memilih tempat penyimpanan *file* yang di *backup*. Kemudian kita tekan *OK* untuk mengakhiri proses *backup* tersebut.



Gambar 2.34 *Backup project*[14]

b. *Restore Backup Project*

Langkah-langkah untuk membuat *restore project* yaitu pada *citect explorer*, pilih *my project* pada *project list*. Kemudian klik kanan pada *my project* yang dipilih tersebut, pilih *restore*, akan keluar tampilan seperti Gambar 2.35. Kolom *backup file* merupakan tempat *project* yang akan di *restore* dengan menggunakan tombol *browse* untuk menemukan lokasi *file*-nya. Kemudian pilih *new project* untuk memasukkan *project* yang akan di *restore* sebagai *project* baru di *citect explorer*. Setelah selesai klik *OK* untuk mengakhiri proses *restore project*.



Gambar 2.35 *Restore project*[14]

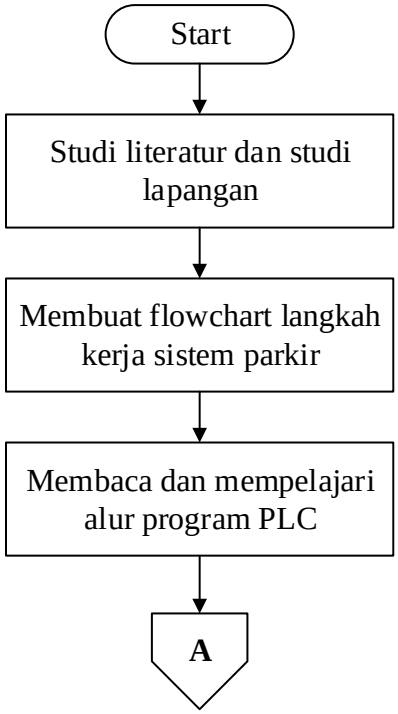
BAB III

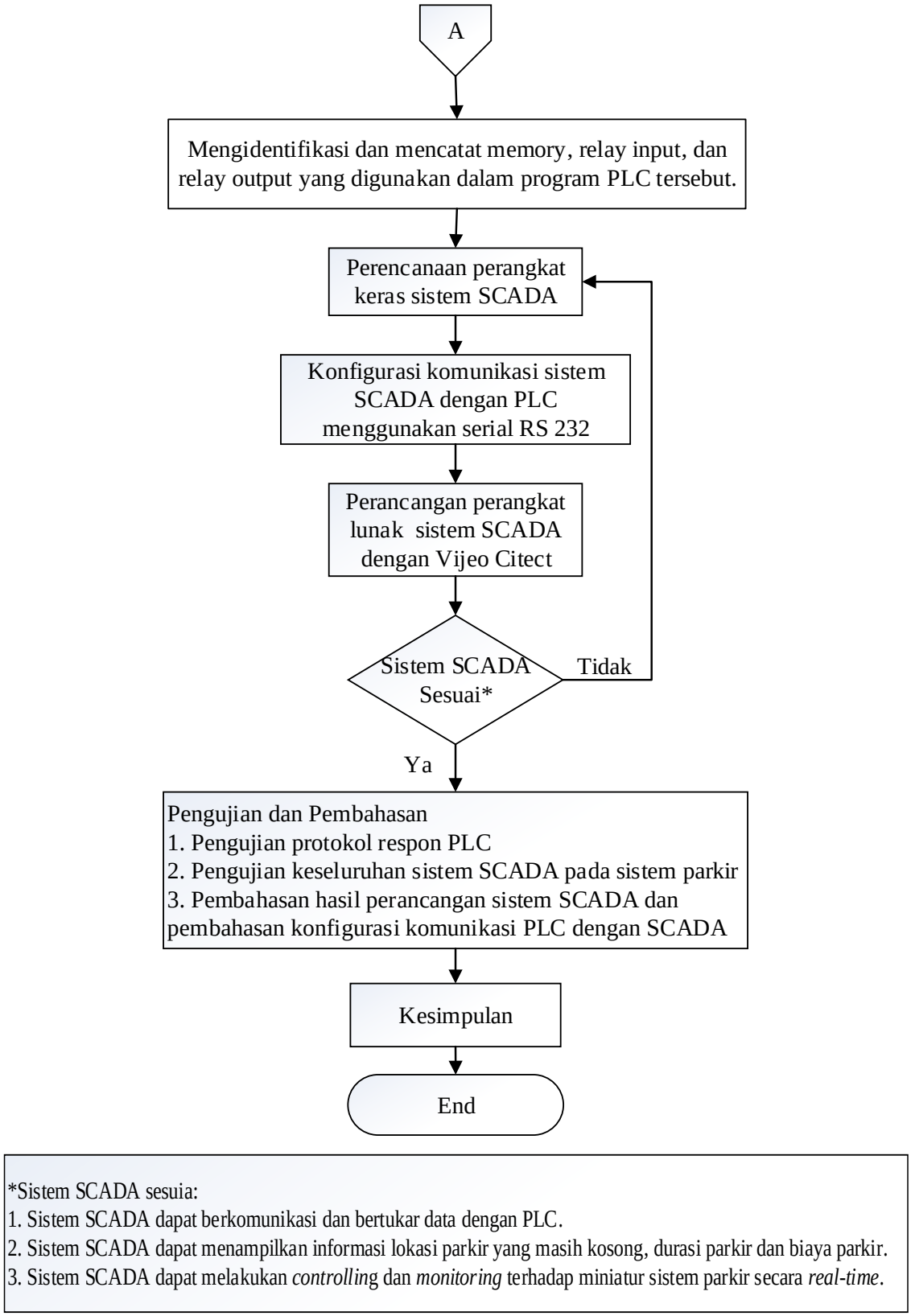
PERANCANGAN SISTEM SCADA

Pada Bab ini akan dijelaskan mengenai perancangan sistem SCADA dengan menggunakan program Vijeo Citect secara keseluruhan. Sistem SCADA pada Tugas Akhir ini berfungsi untuk mengontrol dan memonitoring sistem parkir ramah lingkungan berbasis PLC yang memberikan informasi lokasi parker kepada pemilik kendaraan yang akan parkir, dimana simulator (miniatur) dan program PLC sistem parkir tersebut telah dibuat oleh Ronny Cahyadi Utomo.

3.1 Diagram Alir Penelitian

Agar mendapatkan hasil sesuai dengan yang diharapkan, maka pada penelitian ini dibuat suatu alur kegiatan yang akan dilaksanakan. Alur kegiatan yang dimaksud adalah diagram alir penelitian. Diagram alir ini berisi urutan kerja yang akan dilakukan untuk penelitian ini mulai dari awal sampai akhir. Gambar 3.1 merupakan diagram alir penelitian desain sistem SCADA pada sistem parkir ramah lingkungan berbasis PLC yang mampu memberikan informasi lokasi parkir yang masih kosong.

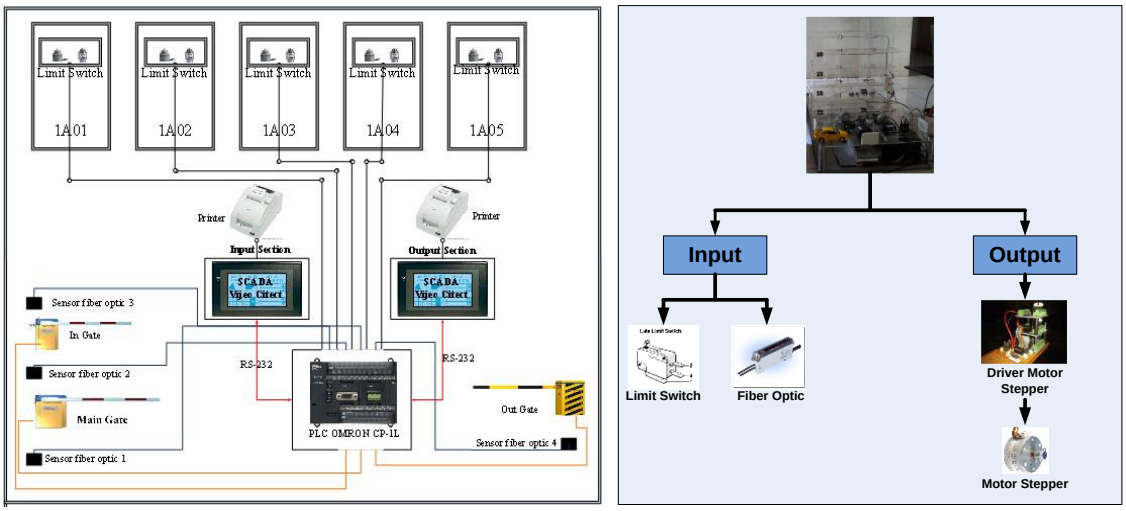




Gambar 3.1 Diagram alir penelitian

3.2 Perancangan Perangkat Keras Sistem SCADA

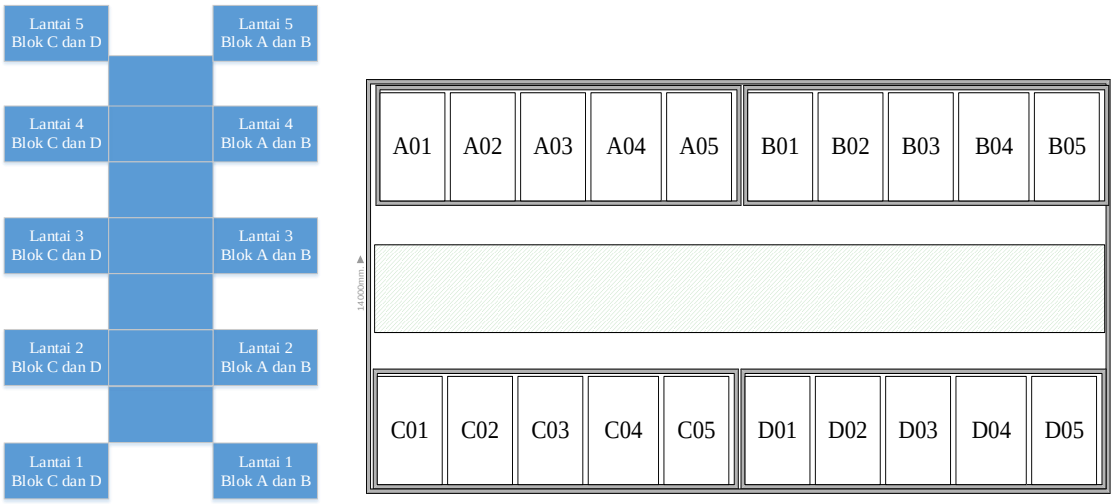
Gambar 3.2 merupakan gambar skema sistem SCADA yang akan dirancang:



a b

Gambar 3.2 a. Blok diagram perangkat keras sistem parkir secara garis besar
b. Blok diagram perangkat keras sistem parkir yang dikontrol PLC

Pada Gambar 3.2a dapat dilihat bahwa perangkat keras sistem parkir diatas digerakkan oleh PC dan PLC. Sedangkan Gambar 3.2b merupakan blok diagram perangkat keras sistem parkir yang dikontrol oleh PLC. Fungsi dari PC ini adalah untuk mengontrol dan memonitoring dari miniatur parkir mobil secara otomatis menggunakan program Vijeo Citect. Sedangkan komunikasi antara PLC dengan SCADA menggunakan serial RS-232. Fungsi dari PLC sendiri adalah untuk menjalankan perintah dari sistem SCADA atau perintah dari perangkat keras yang digunakan pada miniatur sistem parkir, baik *input* maupun *output*. *Input* pada PLC ini digunkan untuk sensor *fiber optic* dan *limit switch*. Sensor *fiber optic* akan digunakan untuk mendeteksi keberadaan mobil saat masuk parkir dan untuk membuka dan menutup gate parkir. Sedangkan *limit switch* digunakan untuk indikator keberadaan mobil pada setiap lokasi parkir. *Output* pada PLC digunakan untuk menggerakkan motor *stepper* melalui *perantara driver* motor *stepper*. Pada sistem parkir ini juga digunakan printer yang dikonfigurasikan dengan sistem SCADA untuk mencetak karcis parkir.



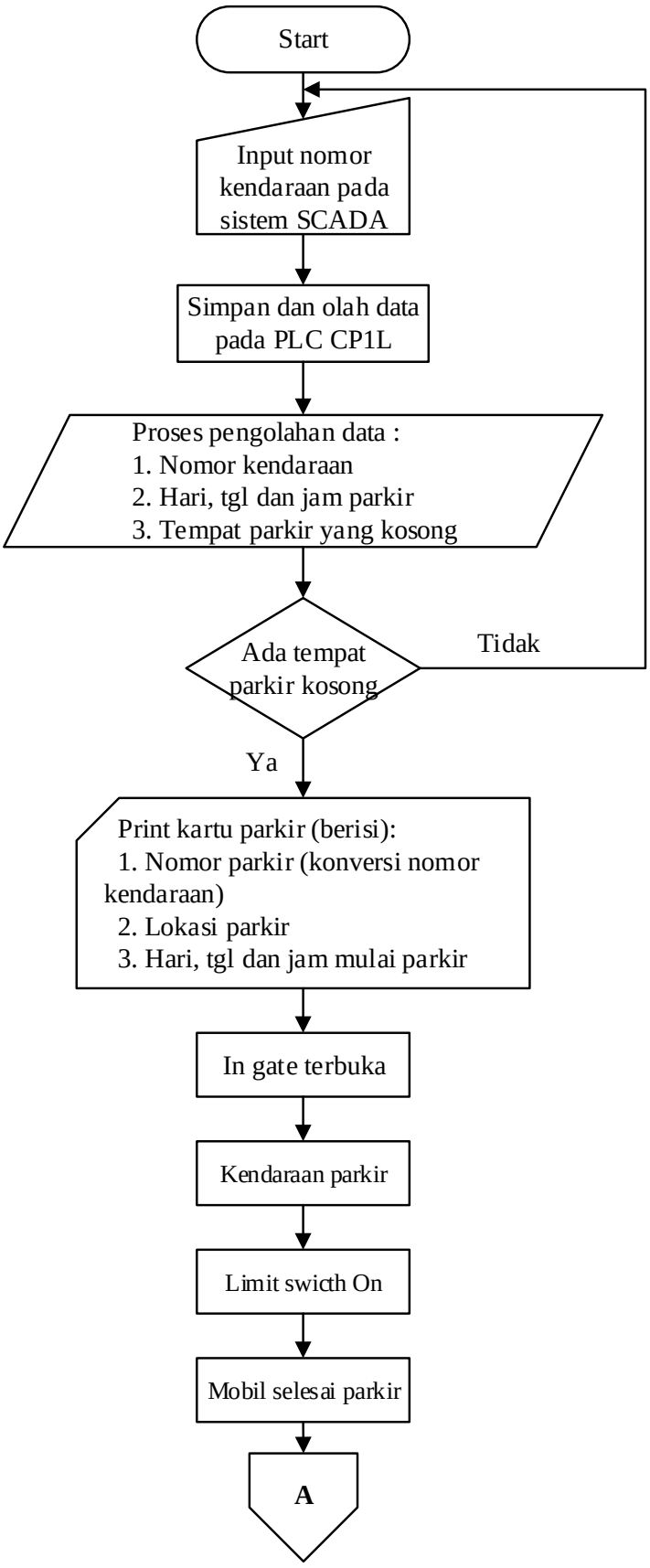
Gambar 3.3 Skema miniatur sistem parkir

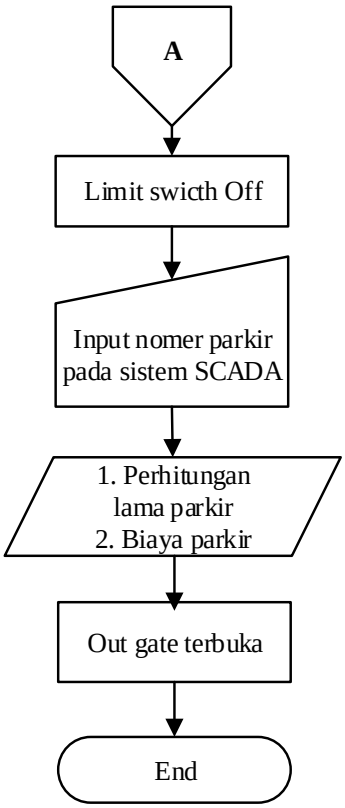
Gambar 3.3 merupakan skema miniatur sistem parkir yang dirancang yang terdiri dari lima lantai, dimana pada setiap lantai terdapat empat blok yaitu blok A-D. Dari masing-masing blok tersebut terdapat lima lokasi parkir, sehingga secara keseluruhan terdapat seratus lokasi parkir. Pada perancangan yang akan dilakukan hanya mengambil beberapa sampel lokasi parkir di setiap lantainya. Sampel lokasi parkir yang akan digunakan sebagai sampel diantaranya yaitu, lantai satu (1A01-1A04, 1B01, 1C01, 1D01), lantai dua (2A01, 2A02, 2B01, 2C01, 2D01), lantai tiga (3A01, 3B01, 3C01), lantai empat (4A01, 4B01, 4C01), dan untuk lantai lima (5A01, 5B01).

3.3 Diagram Alir Keseluruhan Sistem Parkir

Untuk menjalankan program secara keseluruhan, dalam perancangan sistem SCADA sistem parkir ini diperlukan alur dari program yang akan dibuat. Alur program tersebut adalah *flow chart* sistem. Dengan adanya *flow chart* tersebut maka program dapat berjalan sesuai dengan konsep yang direncanakan. *Flow chart* tersebut dijalankan sesuai urutan proses program dari atas ke bawah mengikuti tanda panah yang ada.

Gambar 3.4 adalah *flow chart* program secara keseluruhan untuk mengontrol dan memonitor miniatur sistem parkir mobil tersebut.


Start
Input nomor kendaraan pada sistem SCADA
Simpan dan olah data pada PLC CP1L
Proses pengolahan data :
1. Nomor kendaraan
2. Hari, tgl dan jam parkir
3. Tempat parkir yang kosong
Ada tempat parkir kosong
Tidak
Ya
Print kartu parkir (berisi):
1. Nomor parkir (konversi nomor kendaraan)
2. Lokasi parkir
3. Hari, tgl dan jam mulai parkir
In gate terbuka
Kendaraan parkir
Limit swicth On
Mobil selesai parkir
A



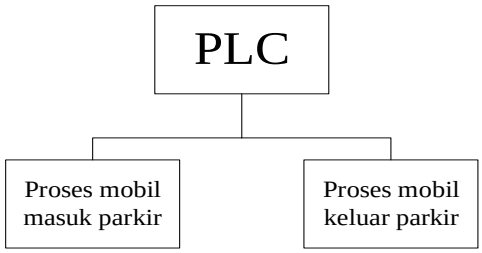
Gambar 3.4 Diagram alir keseluruhan sistem parkir

Berikut ini adalah urutan langkah kerja sistem parkir:

a. Mobil masuk lokasi parkir.
b. Operator menginput plat nomor kendaraan pada sistem SCADA kemudian diproses dalam PLC.
c. Bila ada lokasi parkir yang kosong maka akan tampil kemudian diprint yang terdiri dari nomor parkir, lokasi parkir, hari, tanggal, dan jam mulai parkir.
d. *In gate* membuka.
e. Kendaraan masuk ke lokasi parkir sesuai dengan lokasi yang telah tertera dikarcis.
f. Setelah sampai dilokasi *limit switch* dalam posisi *on*.
g. Selesai parkir mobil meninggalkan lokasi tempat parkir *limit switch* dalam posisi *off*.
h. Menyerahkan kartu parkir ke operator dan operator akan menginput nomer parkir pada sistem SCADA.

i. Setelah diinput akan diproses dalam PLC sehingga akan mengetahui lama parkir dan biaya parkir yang kemudian dicetak. Selanjutnya o*ut gate* terbuka dan mobil meninggalkan lokasi parkir.

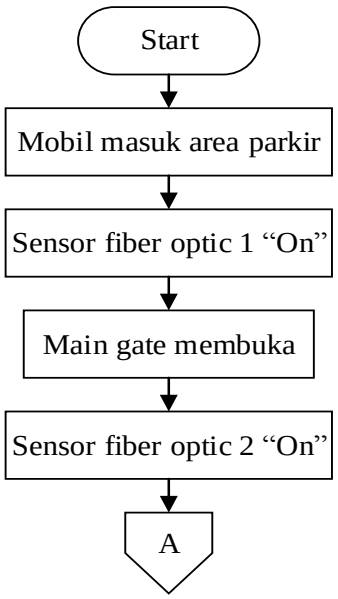
Program PLC yang sudah dibuat digunakan untuk menjalankan proses kerja pada sistem parkir tersebut. Ada dua proses kerja dalam sistem parkir, yaitu proses pada saat mobil masuk area parkir dan pada saat mobil keluar area parkir. Gambar 3.5 adalah blok diagram sistem pada PLC.

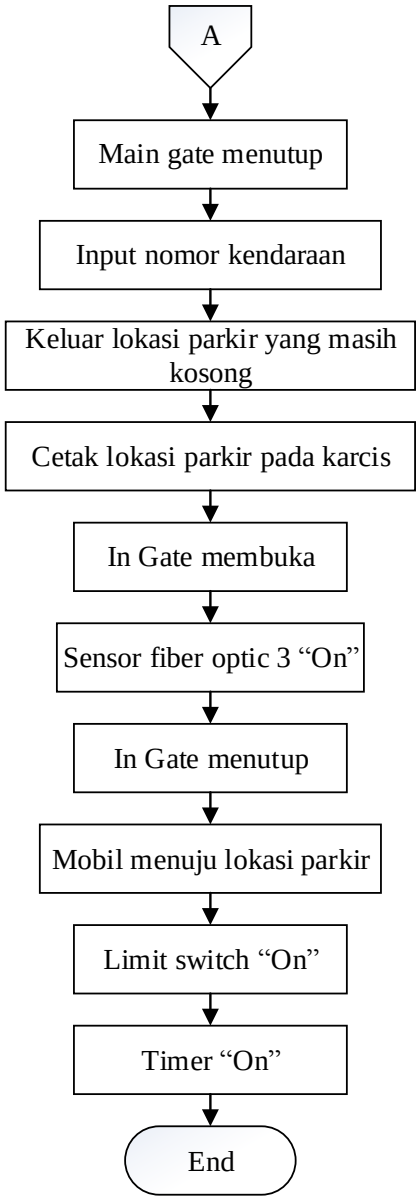


Gambar 3.5 Blok diagram sistem kerja pada PLC

3.3.1 Diagram Alir Proses Mobil Masuk Parkir

Gambar 3.6 di bawah ini merupakan diagram alir pada program PLC dan SCADA dalam melakukan kerja pada saat proses mobil masuk area parkir:





Gambar 3.6 Diagram alir proses mobil masuk parkir

Berikut ini adalah urutan langkah kerja mobil masuk area parkir:

a. Urutan kerja sistem parkir saat mobil masuk area parkir.
   1. Keberadaan mobil tersebut kemudian dideteksi dengan sensor *fiber optic* 1 kemudian sinyal tersebut dikirim ke PLC.
   2. Sinyal dari sensor *fiber optic* 1 ke PLC tersebut digunakan untuk menghidupkan motor *stepper* agar bergerak berlawanan arah jarum jam. Motor *stepper* tersebut digunakan untuk membuka *main gate*.

3. Setelah *main gate* terbuka mobil masuk area *in gate* dan dideteksi sensor *fiber optic* 2 kemudian sinyal tersebut dikirim ke PLC.
4. Sinyal dari sensor *fiber optic* 2 ke PLC tersebut digunakan untuk menghidupkan motor *stepper* agar bergerak searah jarum jam. Motor *stepper* tersebut digunakan untuk menutup *main gate*.

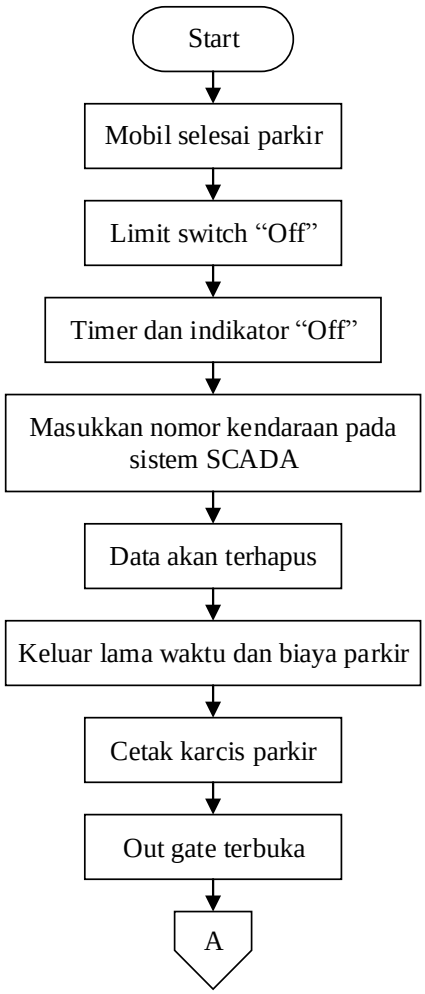
b. Urutan kerja sistem parkir pada saat kendaraan melakukan registrasi sebelum parkir.
   1. Sinyal dari sensor *fiber optic* 2 ke PLC tersebut juga digunakan untuk mengumpan agar data dapat dimasukkan dari sistem SCADA ke PLC. Data yang dimaksud pada sistem parkir ini adalah dengan menginputkan nomor kendaraan ke sistem SCADA kemudian data tersebut dikirim dan disimpan ke memori PLC.
   2. Data dalam memori PLC tersebut kemudian diolah oleh PLC yang telah diprogram (*ladder diagram*) sehingga dapat memberikan informasi data lokasi parkir yang masih kosong. Lokasi-lokasi tempat parkir tersebut dibuat semacam database di dalam memori *internal* PLC.
   3. Data informasi lokasi parkir yang masih kosong dari memori PLC tersebut kemudian dikirim kembali ke sistem SCADA yang kemudian dicetak pada karcis parkir.
   4. Data nomor kendaraan yang dimasukkan dari sistem SCADA ke PLC tersebut juga digunakan untuk membuat umpan menghidupkan motor *stepper* berputar berlawanan arah jarum jam. Motor *stepper* tersebut digunakan untuk membuka *in gate*.

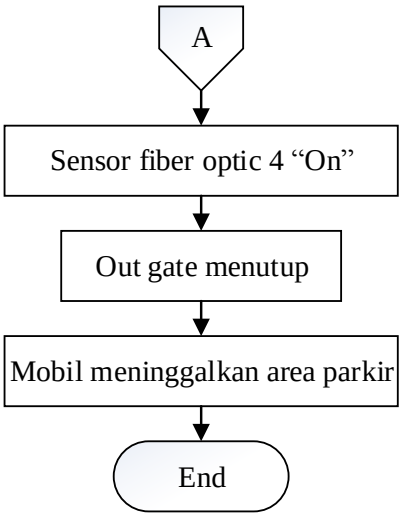
c. Urutan kerja sistem parkir pada saat kendaraan akan menuju lokasi parkir.
   1. Registrasi data dan cetak lokasi parkir selesai, mobil menuju lokasi parkir sesuai dengan yang diinformasikan tersebut. Mobil tersebut dideteksi oleh sensor *fiber optic* 3 yang kemudian sinyal tersebut dikirim ke PLC.
   2. Sinyal dari sensor *fiber optic* 3 tersebut digunakan untuk menghidupkan motor *stepper* agar bergerak searah jarum jam. Motor *stepper* tersebut digunakan untuk menutup *in gate*.

3. Mobil sampai pada lokasi parkir sesuai dengan yang diinformasikan yang ditandai dengan menginjak *limit switch*. Karena terinjak oleh roda mobil tersebut maka *limit switch* pada lokasi parkir tersebut menyala.
4. Sinyal dari *limit switch* tersebut kemudian dikirim ke PLC, dimana sinyal tersebut digunakan untuk menghidupkan program *timer* dalam memori PLC. *Timer* tersebut digunakan untuk mengetahui berapa lama mobil tersebut parkir.
5. Sinyal *limit switch* dan *timer* dari memori PLC tersebut kemudian dikirim ke sistem SCADA yang digunakan untuk me-*monitoring* sistem parkir tersebut.

3.3.2 Diagram Alir Proses Mobil Keluar Parkir

Gambar 3.7 merupakan diagram alir pada program PLC dan SCADA dalam melakukan kerja pada saat proses mobil keluar area parkir:





Gambar 3.7 Diagram alir mobil keluar area parkir

Berikut ini adalah urutan langkah kerja mobil keluar area parkir :

a. Urutan kerja sistem parkir pada saat kendaraan selesai parkir.
    1. Mobil meninggalkan lokasi parkir.
    2. Sehingga *limit switch* pada lokasi parkir tersebut menjadi tidak aktif karena sudah tidak terinjak roda mobil, kemudian *limit switch* mengirimkan sinyal tersebut ke PLC.
    3. Karena *limit switch* tidak aktif maka timer dan indikator pada PLC juga otomatis akan mati. *Data timer* tersebut disimpan dalam memori PLC untuk selanjutnya dikirim ke sistem SCADA.
b. Urutan kerja sistem parkir pada saat kendaraan melakukan registrasi untuk keluar parkir.
    1. Mobil akan keluar tempat parkir, selanjutnya registrasi agar mobil bisa keluar dengan cara memasukkan nomor kendaraan mobil tersebut pada sistem SCADA yang kemudian dikirim ke PLC.
    2. Data masukkan tersebut akan diproses dalam PLC yang telah terprogram, jika data tersebut sesuai dengan data yang ada pada database pada memori PLC maka data tersebut otomatis akan terhapus.

3. Setelah data tersebut terhapus data timer pada memori PLC dikalkulasi untuk menunjukkan biaya sewa parkir mobil tersebut secara otomatis dan kembali disimpan di memori PLC.
4. Data timer dan biaya parkir dari memori PLC selanjutnya dikirim ke sistem SCADA yang kemudian ditampilkan dan dicetak pada karcis parkir.

c. Urutan kerja sistem parkir pada saat kendaraan akan menuju lokasi parkir
1. Data registrasi diatas juga digunakan untuk mengumpan agar motor stepper pada out gate berputar berlawanan arah jarum jam. Motor stepper tersebut digunakan untuk membuka out gate.
2. Mobil meninggalkan tempat parkir yang kemudian mobil tersebut melewati dan dideteksi sensor fiber optic 4. Sinyal dari fiber optic 4 kemudian dikirim ke PLC dan selanjutnya digunakan PLC untuk menghidupkan motor stepper agar berputar searah jarum jam.
3. Motor stepper tersebut digunakan untuk menutup out gate.

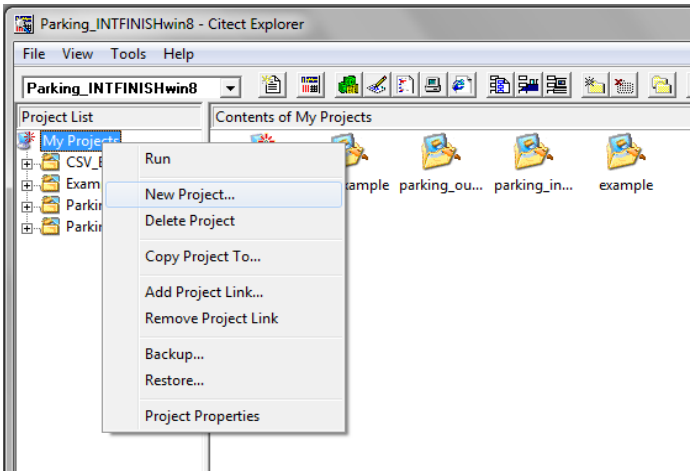
3.4 Perancangan Perangkat Lunak Sistem SCADA dengan *Software* Vjieo Citect

Dalam perancangan sistem SCADA pada penelitian Tugas Akhir ini *software* yang digunakan adalah Vijeo Citect v7.10 r2 (*Demo Version*). Selain program Vijeo Citect, program PLC (*ladder diagram*) juga diperlukan untuk mengontrol dan mengolah data sistem parkir tersebut. Dimana program PLC dan program pada sistem SCADA akan mempunyai keterkaitan yaitu sistem SCADA akan memberikan perintah sedangkan PLC akan menjalankan perintah tersebut dan mengumpan balik perintah tersebut ke sistem SCADA.

3.4.1 Membuat *Project File* Sistem Parkir Pada *Sofware* Vijeo Citect

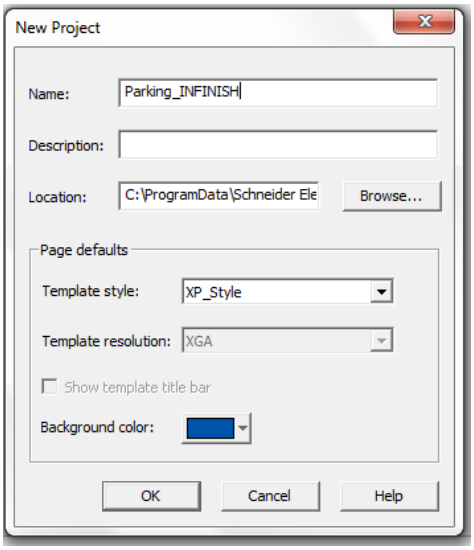
Berikut ini adalah langkah-langkah membuat *project file* sistem parkir dengan menggunakan *software* Vijeo Citect:

a. Membuat *New Project*
1. Untuk membuat *project* baru langkah yang pertama adalah membuka *citect explorer*, kemudian klik kanan *my project* selanjutnya pilih tombol *new project*. Seperti yang ditampilkan pada Gambar 3.8.



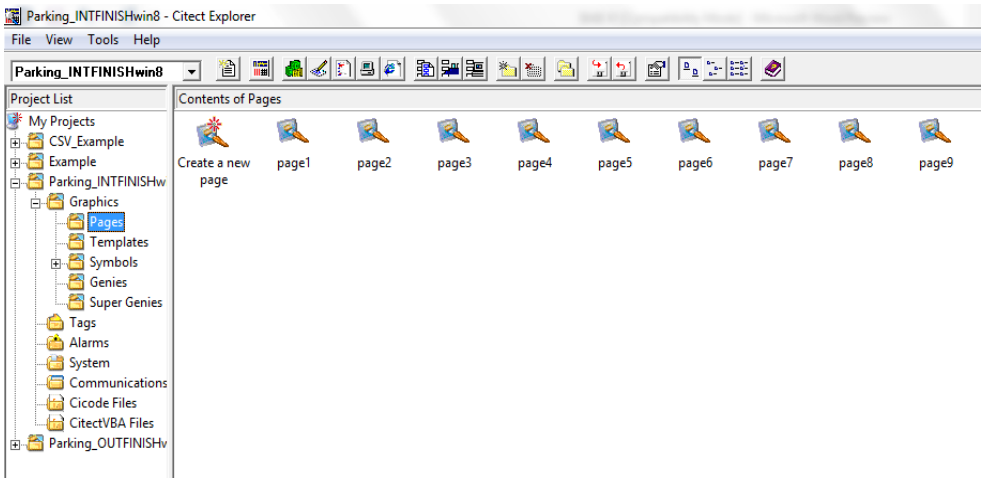
Gambar 3.8 Membuat *new project*

2. Selanjutnya akan muncul menu seperti terlihat pada Gambar 3.9, kemudian melengkapi kolom-kolom yang tertera. Mengisi nama *project* dengan *Parking_INFINISH*, setelah itu memilih *template XP_Style* lalu klik *OK*.



Gambar 3.9 Membuat registrasi *new project*

3. Setelah selesai maka Vijeo Citect akan membuat *data base* yang berisi semua data-data yang akan digunakan oleh *project file* denagan nama *Parking_INFINISH*. Seperti ditunjukkan pada Gambar 3.10.

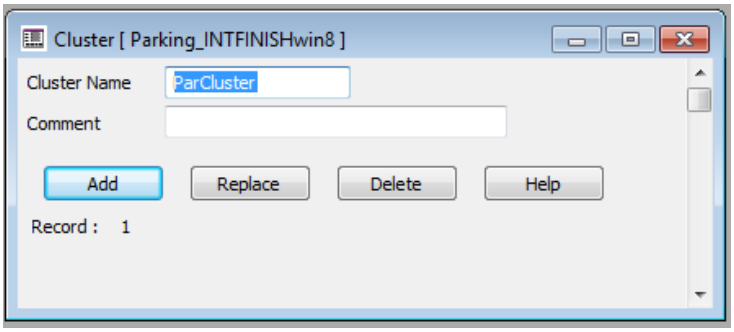


Gambar 3.10 *Database project file Parking_INFINISH*

b. Membuat Konfigurasi *Cluster* dan *Server*

Setiap *project file* yang dibuat harus mempunyai sustu *I/O server*, *alarm*, *report* & *trend server*, dan *display client*. Sedangkan *cluste*r digunakan untuk mengelompokkan *project file* yang telah dibuat. Berikut ini adalah langkah-langkah membuat konfigurasi *cluster* dan *server*.

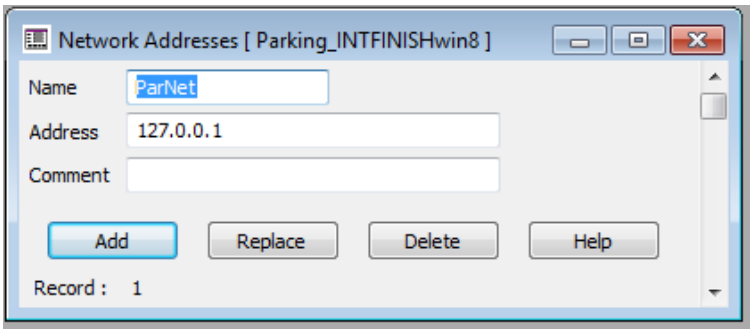
1. Langkah pertama yaitu membuka *citect project editor*, lalu memilih *server* kemudian *cluster* dari menu. Akan keluar tampilan seperti Gambar 3.11 di bawah ini, selanjutnya melengkapi kolom *cluster name* dengan *ParCluster*, setelah itu klik *add*.



Gambar 3.11 Konfigurasi *cluster*

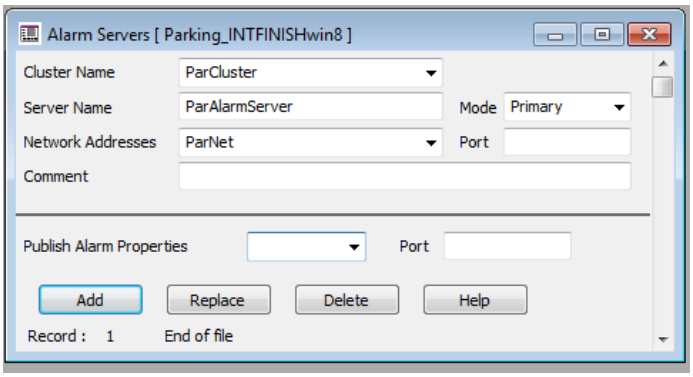
2. Langkah kedua yaitu konfigurasi *network addresses* dengan cara membuka *server* pada *citect project editor* kemudian pilih *network addresses*. Sehingga

muncul seperti Gambar 3.12. Mengisi *name* pada kolom dengan nama *ParNet* dan *address* diisi dengan alamat 127.0.0.1. *Address* pada kolom ini sudah *default* dari program Vijeo Citect, setelah itu klik *add*.



Gambar 3.12 Konfigurasi *network addresses*

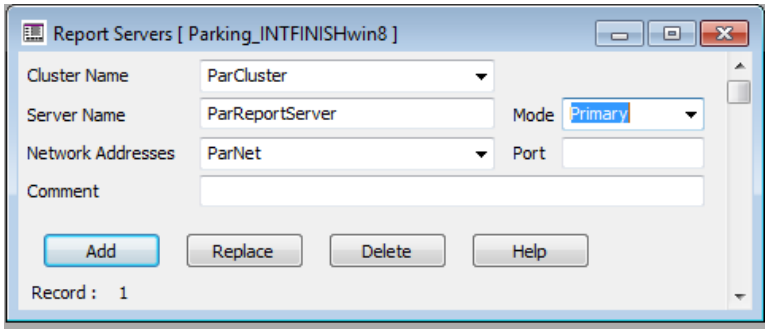
3. Langkah yang ketiga yaitu buka *server* kemudian pilih *alarm server* sehingga muncul Gambar 3.13. *Alarm server* berfungsi untuk mendeteksi kerusakan atau kegagalan dari sistem yang dibuat. Berikut ini adalah cara membuat konfigurasi *alarm server*. Mengisikan *cluster name* dengan *ParCluster*, *server name* diisi dengan *ParAlarmServer*, *network addresses* diisi dengan *Parnet*, dan *mode* diisi *Primary*, setelah itu klik *add*.



Gambar 3.13 Konfigurasi *alarm server*

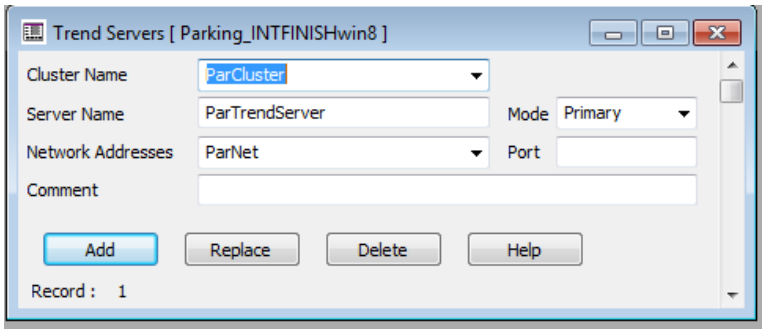
4. Langkah keempat yaitu konfigurasi *report server* dengan cara membuka *server* pada menu *citect project editor* kemudian pilih *report server*. Maka akan muncul tampilan seperti Gambar 3.14. Selanjutnya mengisikan kolom *cluster*

name dengan *ParCluster*, *server name* diisi *ParReportServer*, *network addresses* diisi *ParNet*, dan *mode* diisi *Primary*, setelah itu klik *add*.



Gambar 3.14 Konfigurasi *report server*

5. Langkah kelima yaitu konfigurasi *trend server* dengan cara membuka *server* pada menu *citect project editor* kemudian pilih *trend server*. Sehingga akan muncul tampilan seperti Gambar 3.15. Kemudaian kita lengkapi kolom-kolomnya mulai dari *cluster name* diisi dengan *ParCluster*, *server name* diisi *ParTrendServer*, *network addresses* diisi *ParNet*, dan *mode* diisi *Primary*, setelah itu klik *add*.



Gambar 3.15 Konfigurasi *trend server*

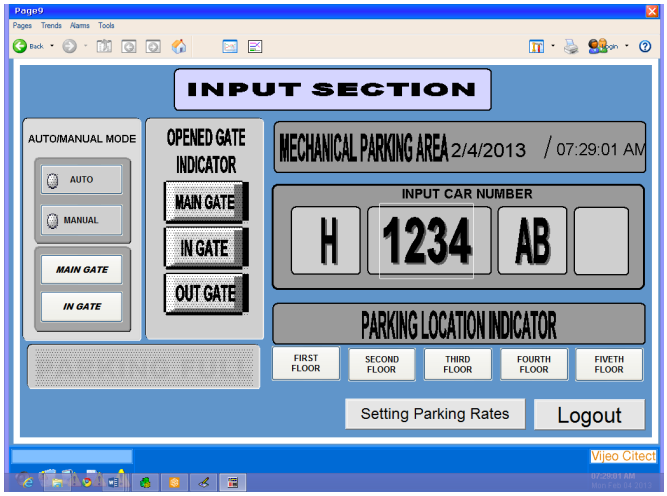
3.4.2 Desain Tampilan *Graphics* Program Sistem SCADA

Desain tampilan *graphics* pada program sistem SCADA memiliki cukup banyak indikator yang dapat mempermudah proses *monitoring* terhadap miniatur sistem parkir tesebut. Selain terdapat banyak indikator, proses cara inputan dari programnya cukup mudah dan tidak membingungkan operator, karena telah didesain sesuai dengan alur

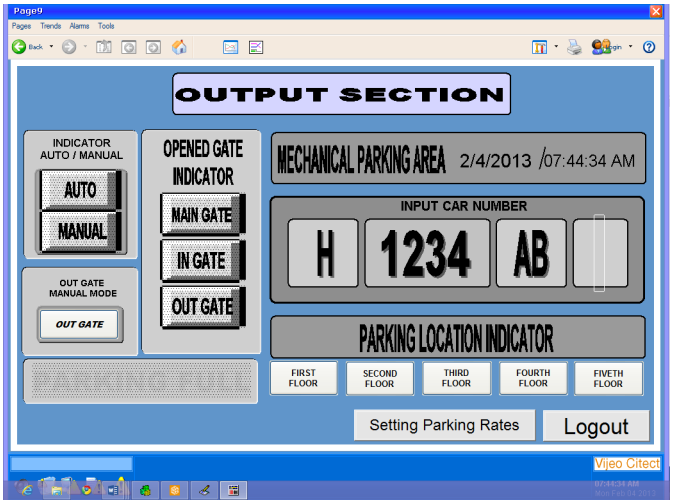
program dan cara mengontrol miniatur sistem parkir yang ada. Jadi operator dapat memasukkan inputan dengan tepat, sehingga program dapat berjalan sesuai dengan alur yang dikehendaki. Semua desain tampilan *graphics* dibuat pada *citect graphics builder* yang terdapat pada *software* Vijeo Citect dengan menggunakan *tools* yang telah disediakan pada *software* tersebut. Gambar 3.16 adalah desain tampilan-tampilan *graphics* sistem SCADA.

Gambar 3.16 Halaman *user registration* sistem SCADA

Pada Gambar 3.16 dapat dilihat bahwa pada tampilan awal sistem SCADA terdapat beberapa pilihan menu yang nantinya digunakan operator untuk menjalankan sistem tersebut. Dimulai dari menu *login*, menu ini digunakan operator untuk registrasi dimana operator harus memasukkan *username* dan *password* untuk bisa masuk halaman beriktnya. Menu yang kedua yaitu *user create*, menu ini berfungsi untuk melakukan registrasi jika operator belum memiliki *username* dan *password*. Untuk mendapatkan *username* dan *password* operator harus mengisi semua kolom registrasi yang disediakan. Menu selanjutnya yaitu *user edit form*, berfungsi untuk menghapus *username* dan *password* yang sudah terdapftar pada sistem SCADA. Selain itu menu ini juga dapat digunakan untuk mengganti *password*. Dan yang terakhir adalah menu *shutdown*, berfungsi untuk mengakhiri semua proses atau untuk menonaktifkan sistem SCADA.



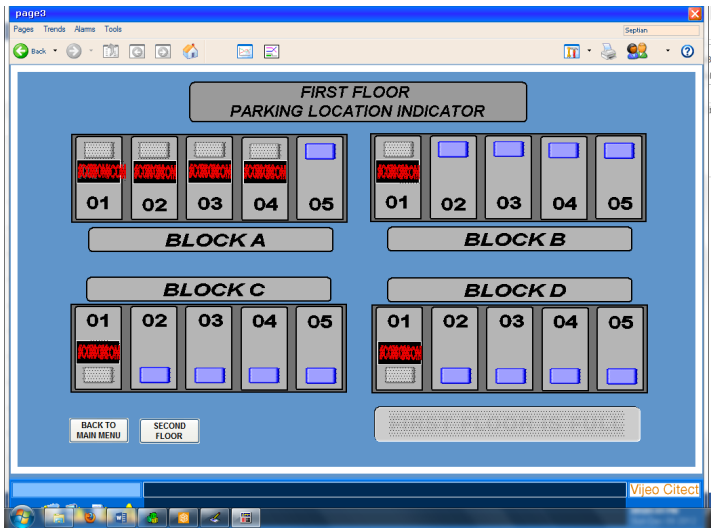
Gambar 3.17 Desain *interface* sistem SCADA *input section*



Gambar 3.18 Desain *interface* sistem SCADA *output section*

Gambar 3.17 dan Gambar 3.18 adalah desain *human machine interface* (HMI) dari sistem SCADA yang berfungsi untuk memberikan perintah dan mengontrol miniatur sistem parkir. HMI dari sistem SCADA ini memiliki banyak menu, yang pertama yaitu *auto manual mode* berfungsi memilih *mode* untuk mengontrol sistem *gate*. Jika *mode auto* maka *gate* akan bekerja secara otomatis, sedangkan jika pada *mode manual* maka *gate* akan bekerja secara *manual* dengan cara operator menekan tombol (*main*, *in* atau *out gate*). Menu yang kedua yaitu *opened gate indicator*, menu ini berfungsi bagi operator sebagai indikator ketika *gate* dalam keadaan membuka atau

menutup maka lampu indikator tersebut akan menyala. Menu yang ketiga yaitu *input car number* yang digunakan operator untuk memasukkan nomor kendaraan. Menu yang keempat yaitu *parking location indicator* berfungsi bagi operator untuk melihat posisi parkir mobil dari lantai satu sampai lantai lima. Jika semua lantai parkir sudah penuh maka *display parking full* akan menyala. Dan yang terakhir adalah menu *logout* yang berfungsi untuk keluar dari halaman HMI dan kembali pada halaman awal sistem SCADA.



Gambar 3.19 Desain sistem monitoring sistem parkir

Pada Gambar 3.19 merupakan bagian dari sistem SCADA parkir yang berfungsi untuk memudahkan operator dalam memonitoring keadaan lokasi parkir pada setiap lantai. Ada beberapa menu yang terdapat pada desain sistem monitoring lokasi parkir yaitu lampu indikator dan *timer* pada setiap lokasi parkir serta indikator jika lantai tersebut penuh. Jika terdapat mobil yang parkir pada posisi parkir tersebut maka lampu indikator dan *timer* akan menyala. Jika semua lokasi parkir sudah terisi maka akan ada teks pemberitahuan bahwa parkir sudah penuh. Menu *back to main menu* berfungsi sebagai navigasi operator kembali pada halaman menu utama (HMI) dari sistem SCADA.

a b

Gambar 3.20 a. Desain *print out karcis* sistem parkir pada *input section*
b. Desain *print out karcis* sistem parkir pada *output section*

Pada Gambar 3.20a dan 3.20b merupakan tampilan *print out* karcis pada sistem SCADA yang berfungsi untuk memberitahukan informasi pada pengguna parkir. Untuk Gambar 3.20a adalah tampilan karcis pada saat mobil masuk parkir yang terdiri dari beberpa informasi diantaranya nomer karcis, nama operator, nomor kendaraan, tanggal dan waktu, serta informasi lokasi tempat parkir yang masih kosong. Sedangkan Gambar 3.20b merupakan tampilan karcis pada saat mobil keluar parkir berisi nomor karcis, nama operator, tanggal&waktu, lama durasi parkir dan yang terkhir yaitu biaya yang harus dibayar pengguna parkir.

Pada Gambar 3.21 merupakan bagian dari sistem SCADA yang berfungsi untuk mengatur tarif dasar sewa parkir dan biaya sewa parkir setelah satu jam.

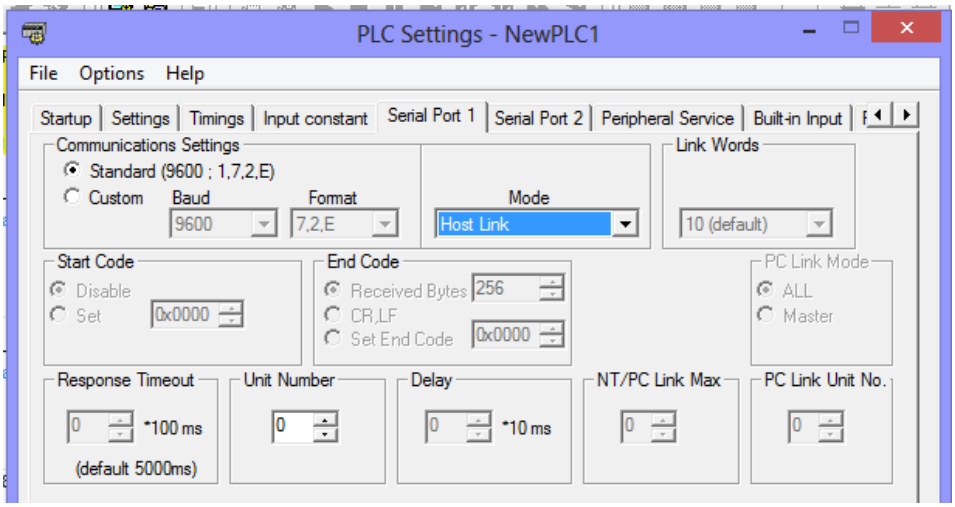
Gambar 3.21 Tampilan *setting parking rates*

3.4.3 Konfigurasi Komunikasi Serial Port

Konfigurasi komunikasi *serial port* berfungsi untuk mensingkronkan antara PLC dengan sistem SCADA (PC) agar dapat berkomunikasi dua arah, *transfer* dan *receiver data*. Pada penilitian Tugas Akhir ini komunikasi yang digunakan antara sistem SCADA dengan PLC adalah menggunakan kabel serial RS-232. Dimana pada komunikasi serial RS-232 ada beberapa parameter yang harus disingkronkan baik pada PLC maupun pada sistem SCADA. Parameter yang harus dikonfigurasi diantaranya *baut rate*, *start bits*, *data bits*, *parity*, dan *stop bits*.

3.4.3.1 Konfigurasi Komunikasi Serial Port Pada PLC

Untuk dapat memerintahkan PLC melakukan sesuatu, atau untuk dapat membaca kondisi PLC, ataupun melakukan banyak hal pada PLC, maka perlu *protocol* komunikasi khusus yang sesuai dengan jenis PLCnya. PLC yang digunakan pada peneliian Tugas Akhir ini adalah PLC Omron Sysmac CP-1L, dimana *protocol* komunikasinya masih terbuka untuk umum, jadi dapat diperintah dan dibaca dengan mudah. Ada beberapa *mode* komunikasi pada PLC omron, pada penelitian ini mode komunikasi yang diguankan adalah *Host link unit*. *Host link unit* berfungsi untuk menjebatani PC dalam memonitor status pengoperasian dan lokasi data dari PLC. Dalam hal ini kita menggunakan PLC Omron CP-1L yang memiliki komunikasi standar seperti Gambar 3.22.



Gambar 3.22 Konfigurasi komunikasi *port serial* PLC omron CP-1L

Dari Gambar 3.22 dapat dilihat bahwa parameter komunikasi pada PLC Omron CP-1L sudah ter-*setting* standar yaitu:

- *Baud rate* = 9.600 (*bit per second*)
- Nomor *start bits (address)* = 0
- Panjang data (*data bits*) = 7 bits
- *Event (vertical) parity* = 1 *bits*
- Nomor *stop bits* = 2

3.4.3.2 Konfigurasi Komunikasi Serial Port Pada Sistem SCADA

Agar dapat berkumunikasi dengan PLC, sehingga PC dapat mengontrol dan memonitor PLC, PC harus membuka komunikasi dengan PLC terlebih dahulu dengan cara mengkonfigurasi *serial port* pada PC agar sesuai dengan konfigurasi *serial port* pada PLC. Setelah komunikasi *serial port* terbuka, maka PC dapat mengontrol dan memonitor PLC dengan leluasa. Dalam penilitian Tugas Akhir ini software SCADA yang digunakan ialah Vijeo Citect v7.1, jadi seluruh komunikasi menggunakan parameter komunikasi data sesuai program dari Vijeo Citect.

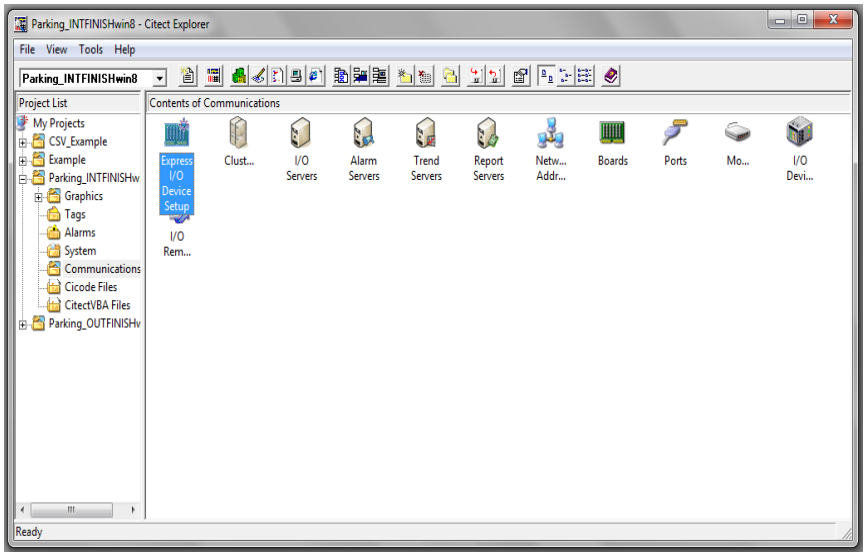
Berikut ini adalah langkah – langkah untuk membuat konfigurasi komunikasi *serial port* pada software Vijeo Citect agar dapat berkomunikasi dengan PLC:

a. Membuat Konfigurasi *External Express I/O Device Setup*

   Konfigurasi *external express I/O device setup* berfungsi untuk membuat komunikasi dengan modul *input* dan *output device* pada PLC agar sistem SCADA dapat dengan mudah menerima dan mengambil data dari PLC. Didalam *external express I/O device setup* ini terdapat bermacam-macam pilihan komunikasi dengan merk PLC yang sudah tersedia.

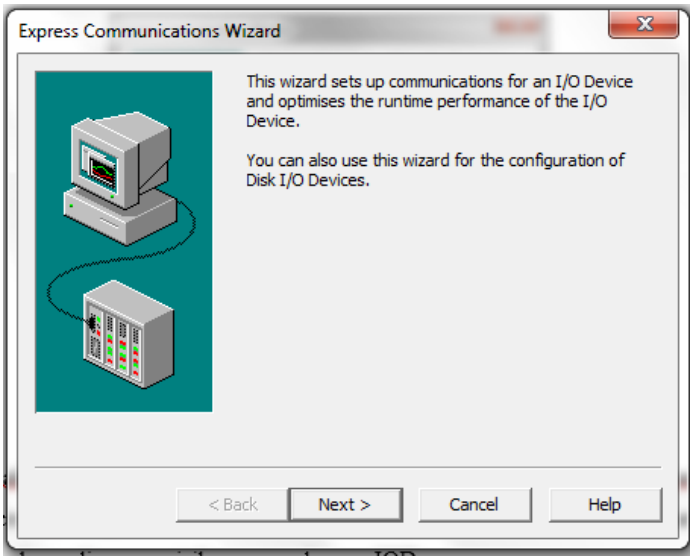
   Berikut ini adalah langkah-langkah untuk membuat konfigurasi *express I/O device setup* pada *software* Vijeo Citect :

   1. Untuk membuat konfigurasi *external express I/O device setup* langkah pertama yaitu membuka *citect explorer* kemudian pilih *project file* yang telah dibuat. Selanjutnya pilih *communication* setelah keluar tampilan gambar seperti gambar 3.23 dibawah ini lalu pilih *express I/O device setup*.



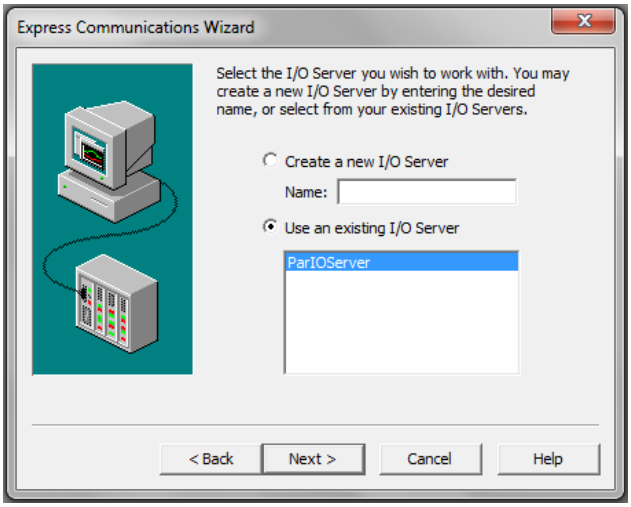
Gambar 3.23 *Communication citect explorer*

2. Setelah kita meng-klik *express I/O device setup* maka akan keluar tampilan seperti Gambar 3.24, kemudian klik *next*.



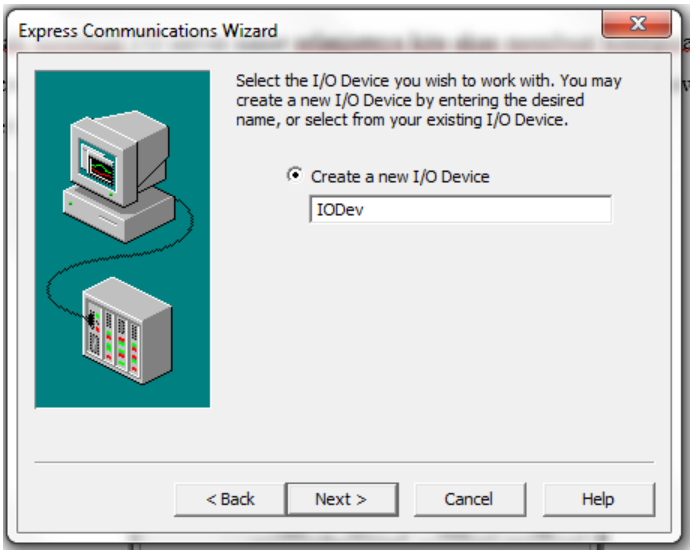
Gambar 3.24 Membuat *external express I/O device setup*

3. Akan mucul tampilan seperti Gambar 3.25, kemudian kita pilih *use an exitsting I/O server* dan selanjutnya isi kolom dengan nama ParIOserver lalu tekan *next*.



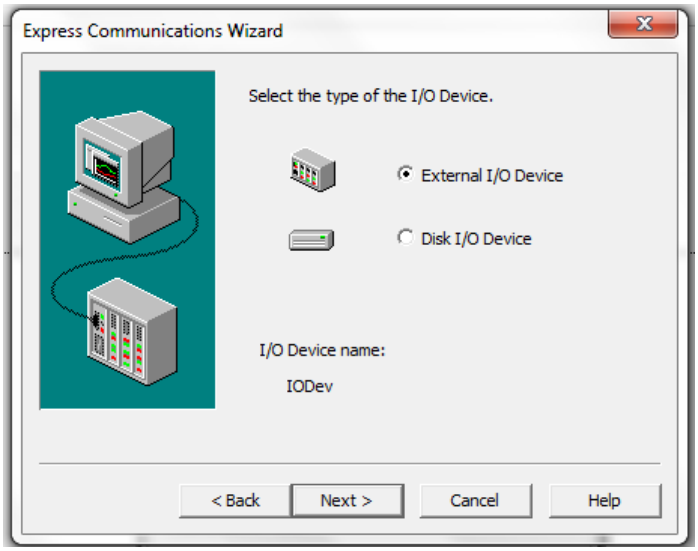
Gambar 3.25 Membuat *I/O server name*

4. Setelah membuat *I/O server name* selanjutnya kita akan membuat konfigurasi *I/O device* seperti Gambar 3.26 dengan cara memilih *create a new I/O server* kemudian mengisikan nama dengan IODev kemudian klik *next*.



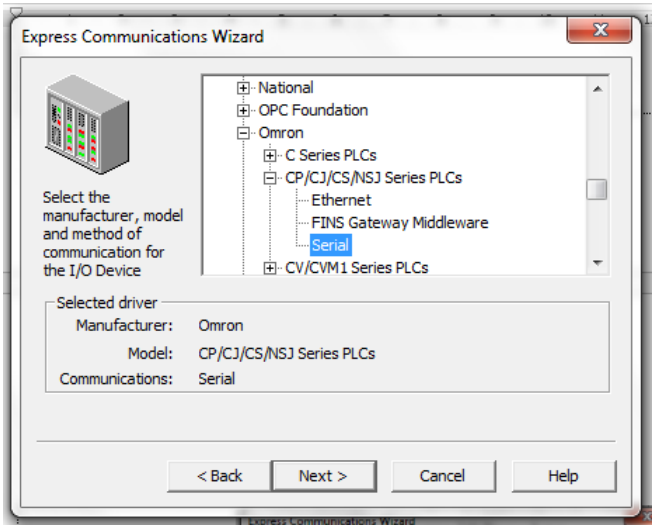
Gambar 3.26 Membuat *I/O device*

5. Selanjutnya membuat konfigurasi tipe dari *I/O device* seperi Gambar 3.27. Karena *I/O device* yang akan dibaca merupakan *I/Odevice* dari PLC maka yang digunakan adalah *external I/O device*. Setelah selesai kemudian klik *next*.



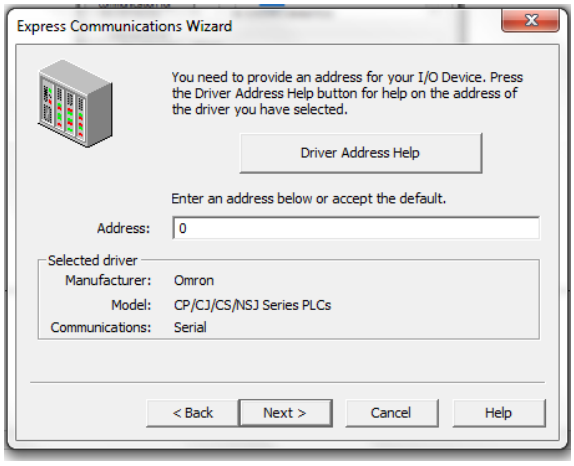
Gambar 3.27 Tipe *I/O device*

6. Langkah selanjutnya membuat konfigurasi model PLC yang akan digunakan seperti pada Gambar 3.28. Karena PLC yang digunakan adalah merk Omron maka pilih omron. Kemudian memilih tipe dari plc yang digunakan, PLC yang digunakan pada penelitian ini adalah tipe CP-1L maka pilih *CP/CJ/CS/NSJ Series PLCs*. Maka muncul pilihan jenis komunikasi yang ingin diguanakan. Komunikasi yang digunakan pada penelitian ini adalah komunikasi Serial. Seteleh selesai lalu klik *next*.



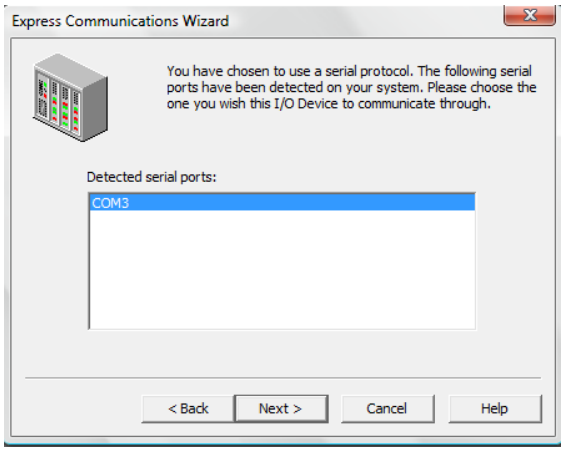
Gambar 3.28 Memilih jenis PLC dan komunkasinya

7. Kemudian muncul tampilan seperti Gambar 3.29 untuk mengisi *address* (alamat) yang akan digunakan. Alamat ini harus sesuai dengan alamat pada PLC. Pada PLC alamat yang digunakan adalah "*0*" maka pada sistem SCADA ini alamat yang harus digunakan adalah "*0*", setelah diisikan lalu klik *next*.



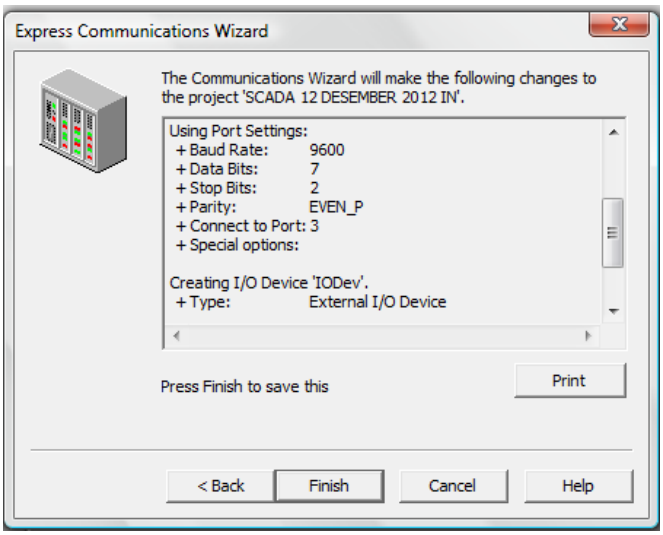
Gambar 3.29 *Addres I/O device*

8. Selanjutnya yaitu memilih *port serial* bertujuan untuk mengkonfigurasi komunikasi pada SCADA dengan PC. Untuk konfigurasi *port serial* setiap komputer mempunyai konfigurasi yang berbeda-beda dan dapat diubah-ubah sesuai *COM* yang dideteksi oleh PC seperti Gambar 3.30. Setelah memilih *COM* kemudian klik *next*.



Gambar 3.30 Konfigurasi *port serial*

9. Kemudian akan muncul tampilan yang berisi informasi konfigurasi yang telah dibuat. Informasi itu diantaranya nilai dari memori *I/O device external* dengan nama *IODev*, *baut rate*, *start bits*, *data bits*, *parity*, dan *stop bits* seperti yang ditunjukkan pada Gambar 3.31. Proses konfigurasi *express I/O device setup* telah selesai, lalu klik *finish*.



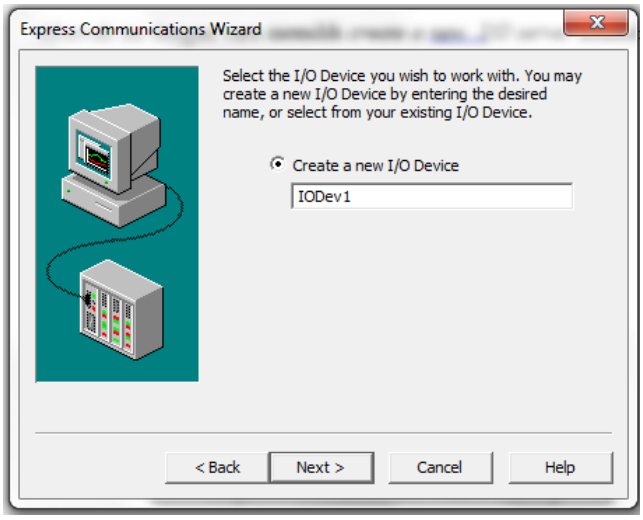
Gambar 3.31 Konfigurasi *I/O device external* selesai

b. Membuat Konfigurasi *Internal Express I/O Device Setup*

Konfigurasi *internal express I/O device setup* berfungsi untuk membuat komunikasi dengan modul *input* dan *output device* pada memori Vijeo Citect agar sistem SCADA dapat dengan mudah menerima dan mengambil data kemudian menampilkan data tersebut.

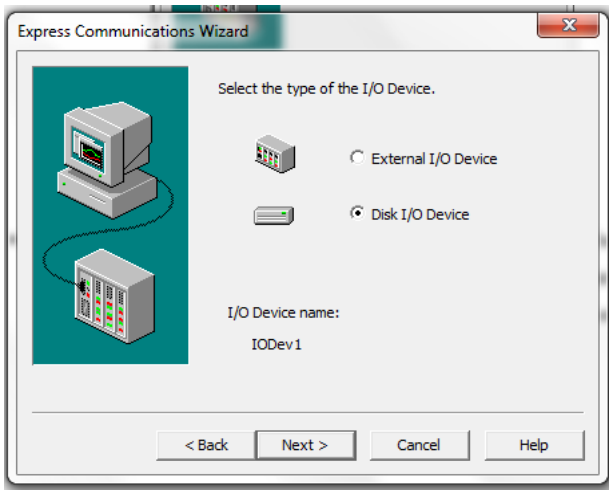
Berikut ini adalah langkah-langkah untuk membuat konfigurasi *internal express I/O device setup* pada *software* Vijeo Citect:

1. Untuk membuat konfigurasi *internal express I/O device setup* langkah pertama sampai langkah ketiga sama dengan pada konfigurasi *external express I/O device setup*.
2. Selanjutnya kita akan membuat konfigurasi *I/O device* seperti Gambar 3.32 di bawah ini dengan cara memilih *create a new I/O server* kemudian mengisikan nama dengan *IODev1* kemudian klik *next*.



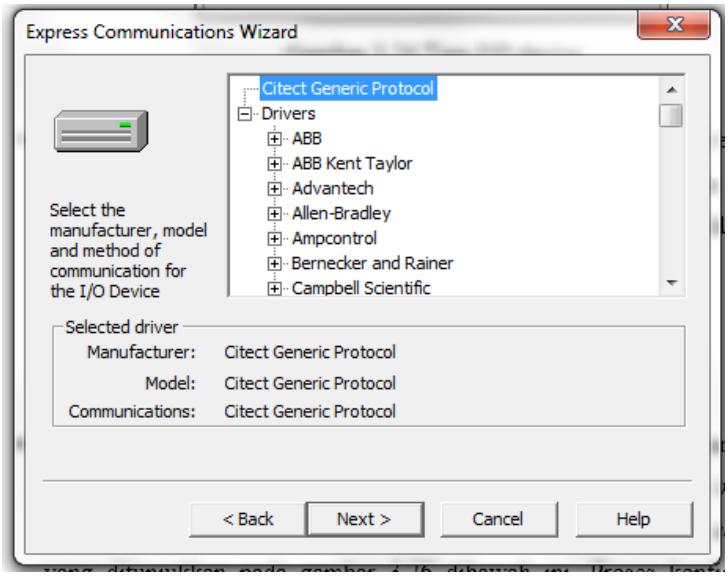
Gambar 3.32 Membuat nama *internal I/O device*

3. Selanjutnya membuat konfigurasi tipe dari *I/O device* seperti Gambar 3.33. Karena *I/O device* yang akan dibaca merupakan *I/O device* dari memori Vijeo Citect maka yang digunakan adalah *internal I/O device*. Setelah selesai kemudian klik *next*.



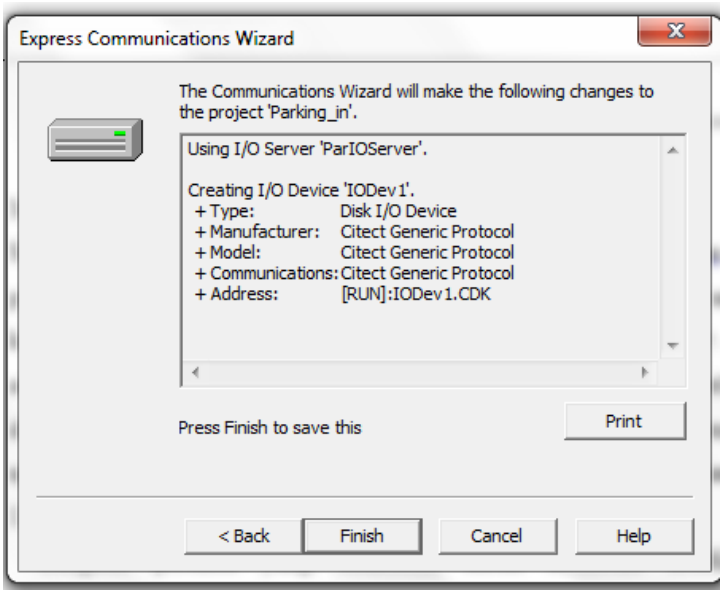
Gambar 3.33 Tipe *I/O device*

4. Langkah selanjutnya membuat konfigurasi memori Vijeo Citect yang akan digunakan seperti tampilan pada Gambar 3.34. Pada konfigurasi ini memori yang digunakan adalah *citect generic protocol* kemudian klik *next*.



Gambar 3.34 Memori Vijeo Citect

5. Kemudian akan muncul tampilan yang berisi informasi konfigurasi yang telah dibuat. Informasi itu diantaranya mulai dari memori *I/O device internal* dengan nama *IODev1*, *type*, *manufacturer*, *model*, *communications*, dan *addreses* seperti yang ditunjukkan pada Gambar 3.35. Proses konfigurasi *internal express I/O device setup* telah selesai, lalu klik *finish*.



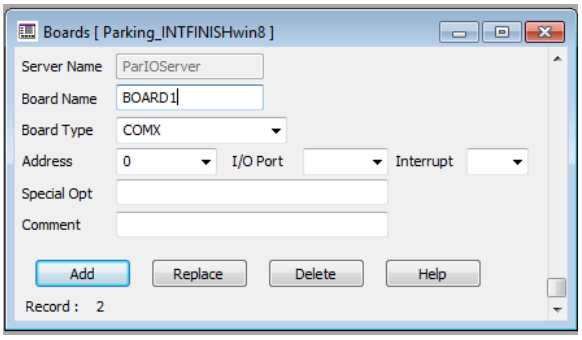
Gambar 3.35 Konfigurasi *internal express I/O device setup* selesai

c. Membuat Konfigurasi *Citect Communication*

Setelah konfigurasi *express I/O device setup* selesai maka akan kita dapatkan parameter komunikasinya yang selanjutnya akan digunakan untuk membuat konfigurasi *citect communication*. Fungsi dari konfigurasi *citect communication* adalah untuk mensingkronkan komunikas sistem SCADA yang telah dibuat agar dapat dibaca oleh PC dan PLC. Ada beberapa parameter yang harus dikonfigurasi agar sistem SCADA ini dapat berkomunikasi dengan PC dan PLC diantaranya:

1. *Boards Communication*

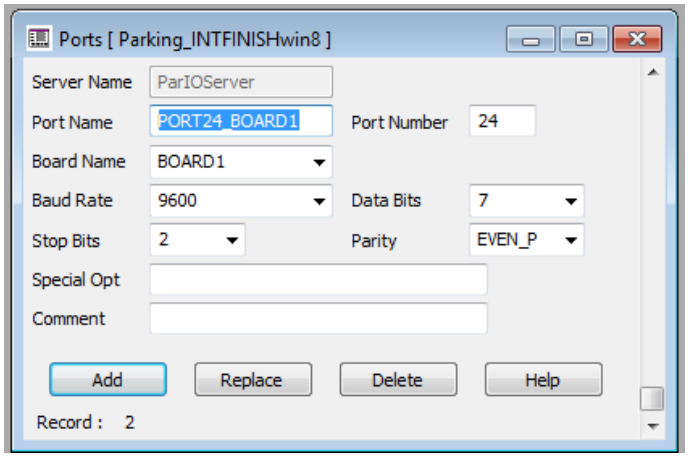
   Langkah pertama yang dilakukan untuk membuat konfigurasi *boards communication* yaitu dengan membuka *toolbar communication* pada *citect project editor* komudian pilih *boards* maka akan muncul tampilan seperti Gambar 3.36. Kemudian mengisi kolom mulai dari *server name* biasanya sudah ter*setting default*, mengisi *board name* dengan nama *Board1*, memilih *board type* dengan tipe *COMX* karena jenis komunikasi yang digunakan adalah serial, *address* diisi dengan *0* sesuai dengan *address* pada PLC, setelah itu klik *add*.



Gambar 3.36 Konfigurasi *boards communication*

2. *Ports Communication*

   Setelah konfigurasi *boards communication* langkah selanjutnya yaitu konfigurasi *ports communication* dengan cara membuka *toolbar communication* pada *citect project editor* kemudian pilih *port*, maka akan muncul tampilan seperti pada Gambar 3.37. Untuk selanjutnya melengkapi kolom untuk konfigurasi mulai dari:



Gambar 3.37 Konfigurasi *ports communication*

- *Port Name*

  Untuk *port name* diisi dengan domain *PORT…_BOARD…*, nomor *port* bisa berubah-ubah tergantung pada konfigurasi yang dibuat oleh komputer. Sedangkan nomor *board* harus sesuai dengan *board name* pada konfigurasi *boards communication.* Pada penelitian ini nomor *port* yang digunakan adalah *PORT24* sedangkan nomor boardnya adalah *BOARD1*.

- *Board Name*

  Board name diisikan dengan BOARD1 sesuai dengan *board name* hasil konfigurasi *boards communication.*

- *Baud rate*, *Stop bits*, *Data bits*, dan *Parity*

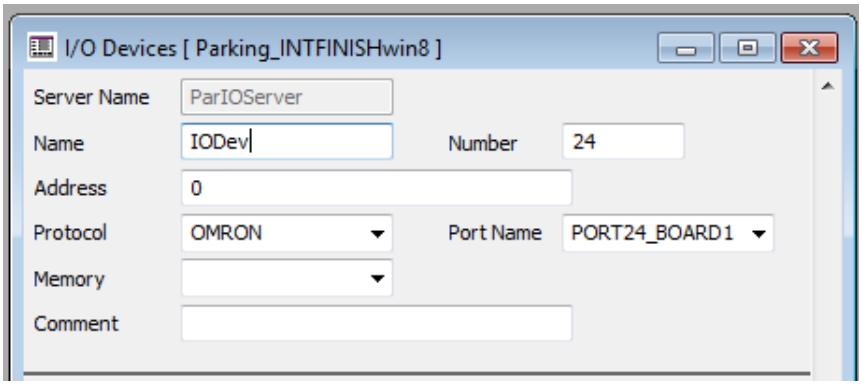
  Untuk nilai dari *baud rate*, *stop bits*, *data bits*, dan *parity* harus disamakan dengan nilai *baud rate*, *stop bits*, *data bits*, dan *parity* dari hasil konfigurasi komunikasi *serial port* pada PLC. Dimana nilai dari *baud rate* "*9600*", *stop bits* "*2*", *data bits* "*7*", *parity* "*Even_P*".

3. *I/O Device Communication*

   Setelah konfigurasi *ports communication* langkah selanjutnya yaitu konfigurasi *I/O device communication* dengan cara membuka *toolbar communication* pada *citect project editor* kemudian pilih *I/O device.* Ada dua konfigurasi yang dibuat yaitu *external I/O device* dan *internal I/O device.*

- *External I/O Device Communication*

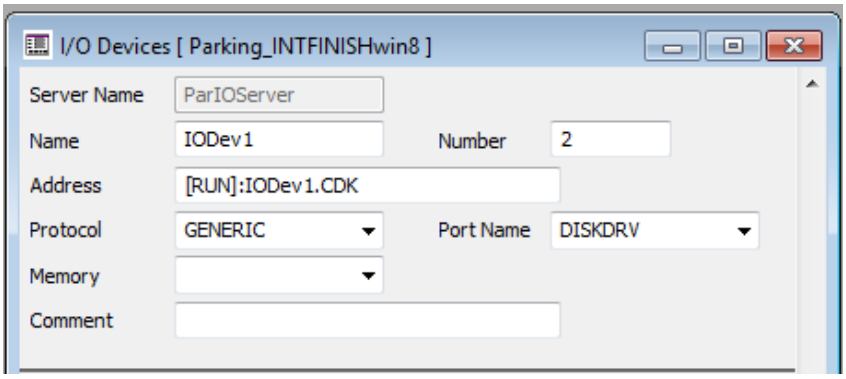
  Untuk parameter yang digunakan pada *external I/O device* ini adalah parameter hasil dari konfigurasi *external express I/O device setup.* Parameternya yaitu *name* "*IODev*", *address* "*0*", *protocol* "*omron*", *number* "*24*", dan *port name* "*PORT24_BOARD1".* Seperti ditunjuukan pada Gambar 3.38.



Gambar 3.38 Konfigurasi *external I/O device communication*

- *Internal I/O device Communication*

  Untuk parameter yang digunakan pada *internal I/O device* ini adalah parameter hasil dari konfigurasi *internal express I/O device setup.* Parameternya yaitu *name* "*IODev1*", *address* "*[RUN]:IODev1.CDK*", *protocol* "*GENERIC*", *number* "*2*", dan *port name* "*DISKDRV".* Seperti ditunjuukan pada Gambar 3.39.



Gambar 3.39 Konfigurasi *internal I/O device communication*

3.4.4 Konfigurasi *Variable Tags*

Sebelum membuat program sistem SCADA pada *software* Vijeo Citect, terlebih dahulu harus membuat *variable tags*. Fungsi dari *variable tags* adalah untuk menggabungkan parameter-parameter yang akan digunakan untuk membuat persamaan. Dimana persamaan tersebut akan digunakan untuk menjalankan fungsi kontrol pada sistem SCADA tersebut. *Variable tags* ini akan dimasukkan pada setiap *graphics* yang dibuat pada *citect graphic builder*. Sebelum membuat *variable tags*, langkah pertama yang harus dilakukan adalah membuat data entri parameter yang akan digunakan diantaranya adalah *I/O device* dan *address* (*memori*) dari memori PLC dan SCADA. Dibawah ini adalah data entri dari *I/O device* dan *address* (*memori*) yang akan digunakan seperti yang ditunjukkan pada Tabel 3.1 dan Tabel 3.2 di bawah ini.

Tabel 3.1 *I/O device*

Nama *I/O Device*	Fungsi
IODev	Untuk membuat konfigurasi memori atau address PLC agar dapat dibaca sistem SCADA.
IODev1	Untuk membuat konfigurasi memori internal sistem SCADA.

Tabel 3.2 Pengalamatan memori pada PLC

Memori	Alamat	Fungsi
Input	0.00	Sebagai *sensor fiber optic* untuk membuka *main gate*
	0.01	Sebagai *sensor fiber optic* untuk menutup *main gate*
	0.02	Sebagai *sensor fiber optic* untuk menutup *in gate*
	0.03	Sebagai *sensor fiber optic* untuk menutup *out gate*
	0.04	Sebagai *input limit switch* untuk posisi 2B01
	0.05	Sebagai *input limit switch* untuk posisi 2A02
	0.06	Sebagai *input limit switch* untuk posisi 2A01
	0.07	Sebagai *input limit switch* untuk posisi 2C01
	0.08	Sebagai *input limit switch* untuk posisi 5B01
	0.09	Sebagai *input limit switch* untuk posisi 3B01

Tabel 3.2 pengalamatan memori pada PLC (lanjutan)

Memori	Alamat	Fungsi
Input	0.10	Sebagai *input limit switch* untuk posisi 5A01
	0.11	Sebagai *input limit switch* untuk posisi 3B01
	1.00	Sebagai *input limit switch* untuk posisi 4B01
	1.01	Sebagai *input limit switch* untuk posisi 3A01
	1.02	Sebagai *input limit switch* untuk posisi 4A01
	1.03	Sebagai *input limit switch* untuk posisi 2C01
	1.04	Sebagai *input limit switch* untuk posisi 4C01
	1.05	Sebagai *input limit switch* untuk posisi 2D01
	1.06	Sebagai *input limit switch* untuk posisi 3C01
	1.07	Sebagai *input limit switch* untuk posisi 1A01
	1.08	Sebagai *input limit switch* untuk posisi 1A02
	1.09	Sebagai *input limit switch* untuk posisi 1A03
	1.10	Sebagai *input limit switch* untuk posisi 1A04
	1.11	Sebagai *input limit switch* untuk posisi 1B01
Output	100.00	Untuk *output motor stepper* 1 data 1 pada *main gate*
	100.01	Untuk *output motor stepper* 1 data 2 pada *main gate*
	100.02	Untuk *output motor stepper* 1 data 3 pada *main gate*
	100.03	Untuk *output motor stepper* 1 data 4 pada *main gate*
	100.05	Untuk *output motor stepper* 2 data 1 pada *in gate*
	100.06	Untuk *output motor stepper* 2 data 2 pada *in gate*
	100.07	Untuk *output motor stepper* 2 data 3 pada *in gate*
	101.00	Untuk *output motor stepper* 2 data 4 pada *in gate*
	101.01	Untuk *output motor stepper* 3 data 1 pada *out gate*
	101.02	Untuk *output motor stepper* 3 data 2 pada *out gate*
	101.03	Untuk *output motor stepper* 3 data 3 pada *out gate*
	101.04	Untuk *output motor stepper* 3 data 4 pada *out gate*
	210.01	Sebagai *output* untuk mengaktifkan posisi parkir 1A01
	210.02	Sebagai *output* untuk mengaktifkan posisi parkir 1A02

Tabel 3.2 pengalamatan memori pada PLC (lanjutan)

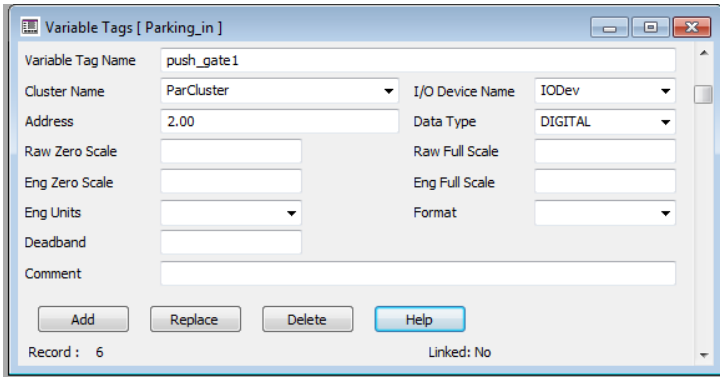
Memori	Alamat	Fungsi
Output	210.03	Sebagai *output* untuk mengaktifkan posisi parkir 1A03
	210.04	Sebagai *output* untuk mengaktifkan posisi parkir 1A04
	210.05	Sebagai *output* untuk mengaktifkan posisi parkir 1B01
	210.06	Sebagai *output* untuk mengaktifkan posisi parkir 1C01
	210.07	Sebagai *output* untuk mengaktifkan posisi parkir 1D01
	211.01	Sebagai *output* untuk mengaktifkan posisi parkir 2A01
	211.02	Sebagai *output* untuk mengaktifkan posisi parkir 2A02
	211.03	Sebagai *output* untuk mengaktifkan posisi parkir 2B01
	211.04	Sebagai *output* untuk mengaktifkan posisi parkir 2C01
	211.05	Sebagai *output* untuk mengaktifkan posisi parkir 2D01
	212.01	Sebagai *output* untuk mengaktifkan posisi parkir 3A01
	212.02	Sebagai *output* untuk mengaktifkan posisi parkir 3B01
	212.03	Sebagai *output* untuk mengaktifkan posisi parkir 3C01
	213.01	Sebagai *output* untuk mengaktifkan posisi parkir 4A01
	213.02	Sebagai *output* untuk mengaktifkan posisi parkir 4B01
	213.03	Sebagai *output* untuk mengaktifkan posisi parkir 4C01
	214.01	Sebagai *output* untuk mengaktifkan posisi parkir 5A01
	214.02	Sebagai *output* untuk mengaktifkan posisi parkir 5B01
IR (*Internal Relay*)	280.11	Sebagai *output* untuk mengaktifkan print posisi parkir 1A01
	280.12	Sebagai *output* untuk mengaktifkan print posisi parkir 1A02
	280.13	Sebagai *output* untuk mengaktifkan print posisi parkir 1A03
	280.14	Sebagai *output* untuk mengaktifkan print posisi parkir 1A04
	280.15	Sebagai *output* untuk mengaktifkan print posisi parkir 1B01
	290.00	Sebagai *output* untuk mengaktifkan print posisi parkir 1C01
	290.11	Sebagai *output* untuk mengaktifkan print posisi parkir 1D01
	380.11	Sebagai output untuk mengaktifkan print posisi parkir 2A01
	380.12	Sebagai *output* untuk mengaktifkan print posisi parkir 2A02
	380.13	Sebagai *output* untuk mengaktifkan print posisi parkir 2B01

Tabel 3.2 pengalamatan memori pada PLC (lanjutan)

Memori	Alamat	Fungsi
IR (*Internal Relay*)	380.14	Sebagai output untuk mengaktifkan print posisi parkir 2C01
	380.15	Sebagai *output* untuk mengaktifkan print posisi parkir 2D01
	480.11	Sebagai *output* untuk mengaktifkan print posisi parkir 3A01
	480.12	Sebagai *output* untuk mengaktifkan print posisi parkir 3B01
	480.13	Sebagai *output* untuk mengaktifkan print posisi parkir 3C01
	480.00	Sebagai *output* untuk mengaktifkan print posisi parkir 4A01
	480.01	Sebagai *output* untuk mengaktifkan print posisi parkir 4B01
	480.02	Sebagai *output* untuk mengaktifkan print posisi parkir 4C01
	480.03	Sebagai *output* untuk mengaktifkan *print* posisi parkir 5A01
	480.04	Sebagai *output* untuk mengaktifkan *print* posisi parkir 5B01
DM (*Data Memory*)	D0	Untuk menyimpan data nomor kendaraan (ASCII) pada saat mobil masuk parkir.
	D1	
	D2	
	D3	
	D5	Untuk menyimpan data nomor kendaraan (ASCII) pada saat mobil keluar parkir.
	D6	
	D7	
	D8	
	D10	Untuk menyimpan lama waktu parkir untuk data detik.
	D11	Untuk menyimpan lama waktu parkir untuk data menit.
	D12	Untuk menyimpan lama waktu parkir untuk data jam.
	D15	Untuk menyimpan data biaya parkir.
	D17	Untuk informasi biaya parkir per menit.
	D19	Untuk informasi tarif dasar sewa parkir.

Setelah membuat data entri dari parameter yang akan digunakan, langkah selanjutnya yaitu membuat konfigurasi *variable tags* dengan cara membuka *menu tag*

pada *toolbar citect project editor* kemudian memilih menu *variable tag* maka akan muncul tampilan seperti pada Gambar 3.40.



Gambar 3.40 Konfigurasi *variable tags*

Selanjutnya melengkapi kolom-kolom tersebut sesuai dengan parameter yang akan digunakan. Pada Gambar 3.40 adalah *variable tags* untuk membuat *push butto*n pada *in gate* parkir. Parameter yang dimasukkan adalah *variable tags name* "*push_gate1*", *cluster name* "*ParCluster*", *address* "*2.00*", *I/O device name* "*IODev*", dan *data type* "*DIGITAL*". Setelah parameternya selesai diisi selanjutnya klik *add.* Dengan cara yang sama seperti langkah di atas maka didapatkan variable tags yang lainnya seperti yang ditunjukkan pada Table 3.3:

Tabel 3.3 *Variable tags* pada SCADA sistem parkir

No	*Variable Tags Name*	*Data Type*	*I/O Device*	*Address*
1	man	DIGITAL	IODev	2.04
2	auto_on	DIGITAL	IODev	2.03
3	push_gate1	DIGITAL	IODev	2.00
4	push_gate2	DIGITAL	IODev	2.01
5	push_gate3	DIGITAL	IODev	2.03
6	gate1	BYTE	IODev	50.01
7	gate2	DIGITAL	IODev	50.02
8	gate3	DIGITAL	IODev	50.03
9	in_s	STRING	IODev	D0

Tabel 3.3 *Variable tags* pada SCADA sistem parker (lanjutan)

No	*Variable Tags Name*	*Data Type*	*I/O Device*	*Address*
10	in_s1	STRING	IODev1	S1
11	in_r	INT	IODev	D1
12	in_r1	INT	IODev1	I1
13	in_sr	STRING	IODev	D2
14	in_sr1	STRING	IODev1	S2
15	in_srf	STRING	IODev	D3
16	in_srf1	STRING	IODev1	S3
17	full	BYTE	IODev	200.00
19	penuh1	BYTE	IODev	200.01
20	penuh2	BYTE	IODev	200.02
21	penuh3	BYTE	IODev	200.03
22	penuh4	BYTE	IODev	200.04
23	penuh5	BYTE	IODev	200.05
24	posisi_1a01	BYTE	IODev	210.01
25	posisi_1a02	BYTE	IODev	210.02
26	posisi_1a03	BYTE	IODev	210.03
27	posisi_1a04	BYTE	IODev	210.04
28	posisi_1b01	BYTE	IODev	210.05
29	posisi_1c01	BYTE	IODev	210.06
30	posisi_1d01	BYTE	IODev	210.07
31	posisi_2a01	BYTE	IODev	211.01
32	posisi_2a02	BYTE	IODev	211.02
33	posisi_2b01	BYTE	IODev	211.03
34	posisi_2c01	BYTE	IODev	211.04
35	posisi_2d01	BYTE	IODev	211.05
36	posisi_3a01	BYTE	IODev	212.01
37	posisi_3b01	BYTE	IODev	212.02
38	posisi_3c01	BYTE	IODev	212.03

Tabel 3.3 *Variable tags* pada SCADA sistem parkir (lanjutan)

No	*Variable Tags Name*	*Data Type*	*I/O Device*	*Address*
39	posisi_4a01	BYTE	IODev	213.01
40	posisi_4b01	BYTE	IODev	213.02
41	posisi_4c01	BYTE	IODev	213.03
42	posisi_5a01	BYTE	IODev	214.01
43	posisi_5b01	BYTE	IODev	214.02
44	posisi_print1a01	DIGITAL	IODev	280.11
45	posisi_print1a02	DIGITAL	IODev	280.12
46	posisi_print1a03	DIGITAL	IODev	280.13
47	posisi_print1a04	DIGITAL	IODev	280.14
48	posisi_print1b01	DIGITAL	IODev	280.15
49	posisi_print1c01	DIGITAL	IODev	290.00
50	posisi_print1d01	DIGITAL	IODev	290.01
51	posisi_print2a01	DIGITAL	IODev	380.11
52	posisi_print2a02	DIGITAL	IODev	380.12
53	posisi_print2b01	DIGITAL	IODev	380.13
54	posisi_print2c01	DIGITAL	IODev	380.14
55	posisi_print2d01	DIGITAL	IODev	380.15
56	posisi_print3a01	DIGITAL	IODev	480.11
57	posisi_print3b01	DIGITAL	IODev	480.12
58	posisi_print2c01	DIGITAL	IODev	380.14
59	posisi_print2d01	DIGITAL	IODev	380.15
60	posisi_print3a01	DIGITAL	IODev	480.11
61	posisi_print3b01	DIGITAL	IODev	480.12
62	posisi_print3c01	DIGITAL	IODev	480.13
63	posisi_print4a01	DIGITAL	IODev	480.00
64	posisi_print4b01	DIGITAL	IODev	480.01
65	posisi_print4c01	DIGITAL	IODev	480.02
66	posisi_print5a01	DIGITAL	IODev	480.03

Tabel 3.3 *Variable tags* pada SCADA sistem parkir (lanjutan)

No	*Variable Tags Name*	*Data Type*	*I/O Device*	*Address*
67	posisi_print5b01	DIGITAL	IODev	480.04
68	in_d1a01	INT	IODev	D1001
69	in_m1a01	INT	IODev	D1201
70	in_j1a01	INT	IODev	D1301
71	in_d1a02	INT	IODev	D1002
72	in_d1a03	INT	IODev	D1003
73	in_d1a04	INT	IODev	D1004
74	in_d1c01	INT	IODev	D1006
75	in_d1d01	INT	IODev	D1007
76	in_m1a02	INT	IODev	D1202
77	in_m1a03	INT	IODev	D1203
78	in_m1a04	INT	IODev	D1204
79	in_m1b01	INT	IODev	D1205
80	in_m1c01	INT	IODev	D1206
81	in_m1d01	INT	IODev	D1207
82	in_j1a02	INT	IODev	D1302
83	in_j1a03	INT	IODev	D1303
84	in_j1a04	INT	IODev	D1304
85	in_j1b01	INT	IODev	D1305
86	in_j1c01	INT	IODev	D1306
87	in_j1d01	INT	IODev	D1307
88	in_d1b01	INT	IODev	D1005
89	in_d2a01	INT	IODev	D2001
90	in_d2a02	INT	IODev	D2002
91	in_d2b01	INT	IODev	D2003
92	in_d2c01	INT	IODev	D2004
93	in_d2d01	INT	IODev	D2005
94	in_m2a01	INT	IODev	D2201

Tabel 3.3 *Variable tags* pada SCADA sistem parkir (lanjutan)

No	*Variable Tags Name*	*Data Type*	*I/O Device*	*Address*
95	in_m2a02	INT	IODev	D2202
96	in_m2b01	INT	IODev	D2203
97	in_m2c01	INT	IODev	D2204
98	in_m2d01	INT	IODev	D2205
99	in_j2a01	INT	IODev	D2301
100	in_j2a02	INT	IODev	D2302
101	in_j2c01	INT	IODev	D2304
102	in_j2d01	INT	IODev	D2305
103	in_j2b01	INT	IODev	D2303
104	in_j3a01	INT	IODev	D3301
105	in_j3b01	INT	IODev	D3302
106	in_j3c01	INT	IODev	D3303
107	in_m3a01	INT	IODev	D3201
108	in_m3b01	INT	IODev	D3202
109	in_m3c01	INT	IODev	D3203
110	in_d3a01	INT	IODev	D3001
111	in_d3b01	INT	IODev	D3002
112	in_d3c01	INT	IODev	D3003
113	in_d4a01	INT	IODev	D4001
114	in_d4b01	INT	IODev	D4002
115	in_d4c01	INT	IODev	D4003
116	in_m4a01	INT	IODev	D4201
117	in_m4b01	INT	IODev	D4202
118	in_m4c01	INT	IODev	D4203
119	in_j4a01	INT	IODev	D4301
120	in_j4b01	INT	IODev	D4302
121	in_j4c01	INT	IODev	D4303
122	in_j5a01	INT	IODev	D5301

Tabel 3.3 *Variable tags* pada SCADA sistem parkir (lanjutan)

No	*Variable Tags Name*	*Data Type*	*I/O Device*	*Address*
123	in_j5b01	INT	IODev	D5302
124	in_m5a01	INT	IODev	D5201
125	in_m5b01	INT	IODev	D5202
126	in_d5a01	INT	IODev	D5001
127	in_d5b01	INT	IODev	D5002
128	in_fee	INT	IODev	D17
129	in_feecash	INT	IODev	D19
130	in_fee1	INT	IODev1	I6
131	in_feecash1	INT	IODev1	I8

3.4.5 Perancangan Program Sistem SCADA

Pada perancangan perangkat lunak sistem SCADA pada *input* dan *output section* terdiri dari 4 kategori program yaitu program *user registration*, program HMI, program untuk *monitoring*, dan program *printout karcis* parkir. Pada program *user registration* terdiri dari program untuk *login user*, *user create*, *user edit form*, dan *shutdown*. Dilanjutkan pada program HMI yang terdiri dari tuju bagian yaitu kendali *gate*, *indicator gate*, *input car number*, *parking location indicator*, *full parking indicator*, *log out*, dan *input tarif* dasar parkir. Pada program untuk *monitoring* terdiri dari indikator lokasi parkir, indikator jika parkir penuh dan *timer* setiap lokasi parkir dari lantai satu sampai lantai lima. Selanjutnya merupakan program untuk *printout* karcis parkir, dimana pada *input section* halaman karcis parkir terdiri dari informasi nama operator, nomor kendaraan yang masuk, tanggal serta jam parkir, dan terakhir lokasi parkir. Sedangkan pada *output section* halaman karcis parkir terdiri dari informasi nama operator, tanggal dan jam, waktu lama parkir, serta biaya sewa parkir.

3.4.5.1 Perancangan Program Sistem SCADA Pada *Input Section*

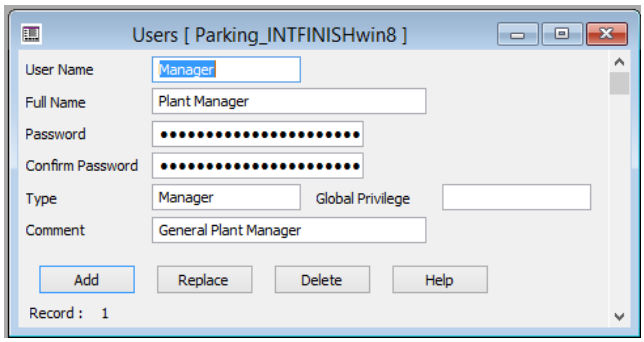
3.4.5.1.1 Perancangan Program *User Registration*

Gambar 3.41 merupakan desain tampilan *graphic* dari program *user registration* yang telah dirancang. Dimana ada empat bagian program utama yaitu *login user*, *user create*, *user edit form* dan *shutdown*. Program-program tersebut berjalan secara independen tanpa terpengaruh koneksi dari PLC.

Gambar 3.41 *User Registration*

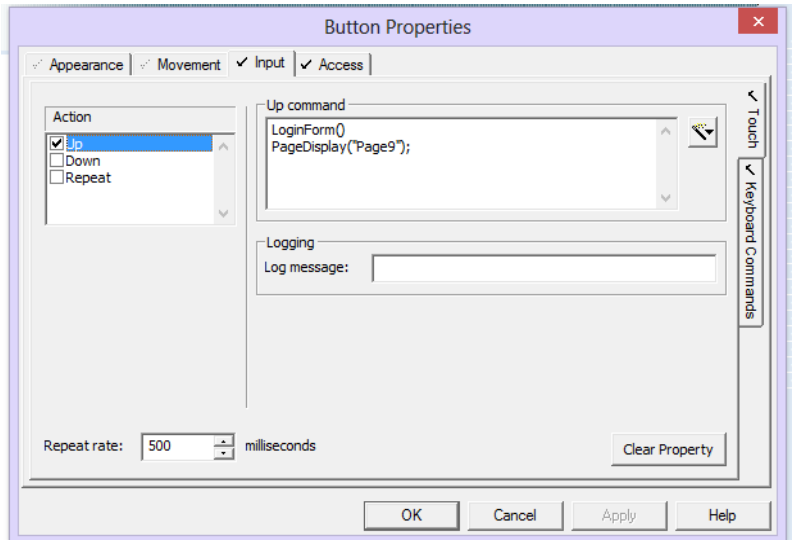
a. Perancangan program *login user*

Untuk membuat program *user login* langkah pertama yang dilakukan adalah membuat *database username* dan *password* pada sistem tersebut dengan cara membuka menu sistem pada *toolbar citect project editor* kemudian pilih *user*. Dimana *username* dan *password* pada *database* yang telah dibuat digunakan sebagai sampel data. Gambar 3.42 merupakan *database* yang telah dibuat.

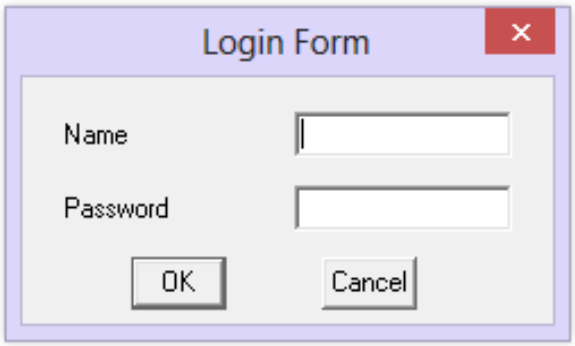


Gambar 3.42 *Database username* dan *password*

Setelah membuat database langkah selanjutnya yaitu memasukkan fungsi yang akan digunakan untuk menjalankan program tersebut. Ada dua fungsi yang digunakan untuk membuat program *user login*, yaitu yang pertama fungsi "*LoginForm()*" berfungsi untuk menjalankan *username* dan *password* yang telah dimasukkan kemudian dicocokan dengan *username* dan *password* yang ada pada *database*, bila nilainya sama maka *login* berhasil. Fungsi yang kedua yaitu "*PageDisplay("Page9')*" yang berfungsi untuk masuk ke halaman HMI jika login tersebut berhasil. Gambar 3.43 ini merupakan fungsi program yang digunakan untuk membuat program tersebut. Sedangkan Gambar 3.44 merupakan tampilan *user login* setelah dijalankan.



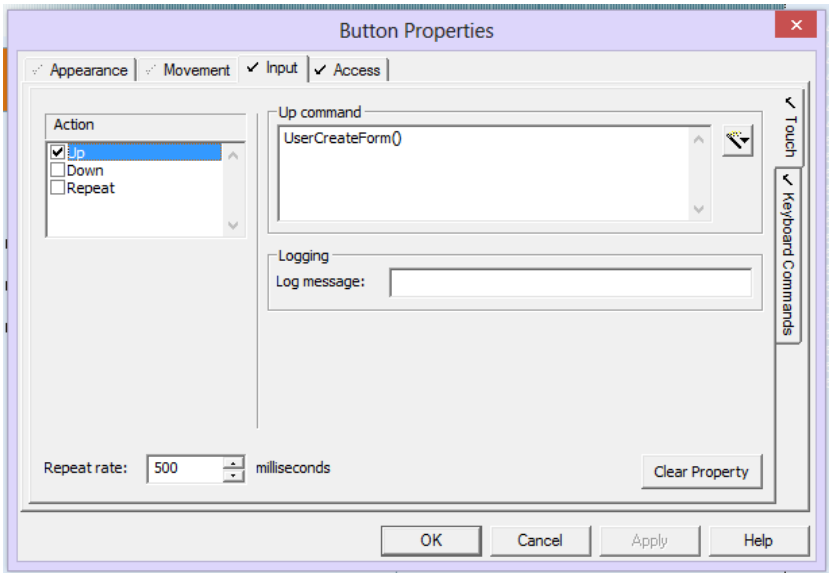
Gambar 3.43 Program *user login*



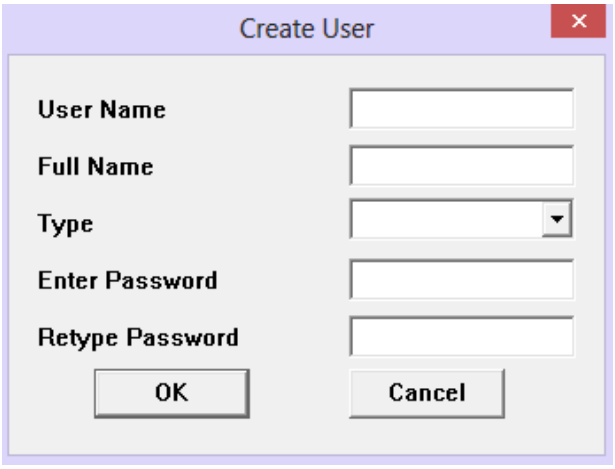
Gambar 3.44 Tampilan *login use*r setelah dijalankan

b. Perancangan program *user create*

User create ini berfungsi untuk membuat *username* dan *password* yang akan digunakan untuk login pada sistem SCADA. Fungsi yang digunakan untuk membuat program ini yaitu "*UserCreateForm*", fungsi ini akan menjalankan program *user create* dengan cara mengirim *username* dan *password* yang telah dibuat ke *database*. Gambar 3.45 merupakan fungsi program yang digunakan pada *user create*, sedangkan Gambar 3.46 merupakan hasil program *user create* setelah dijalankan.



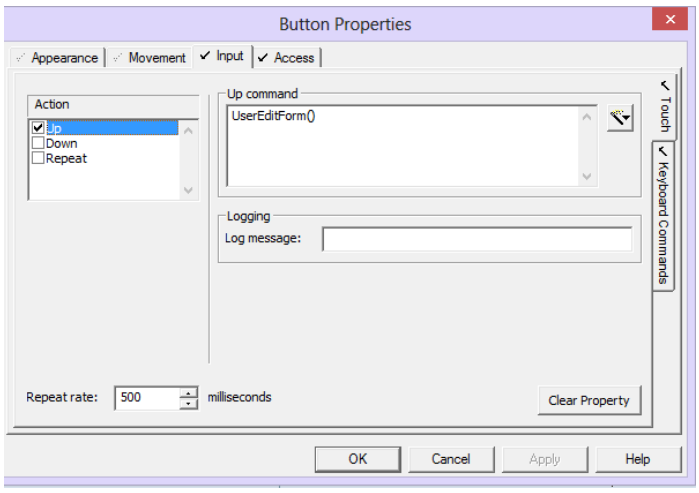
Gambar 3.45 Program *user create*



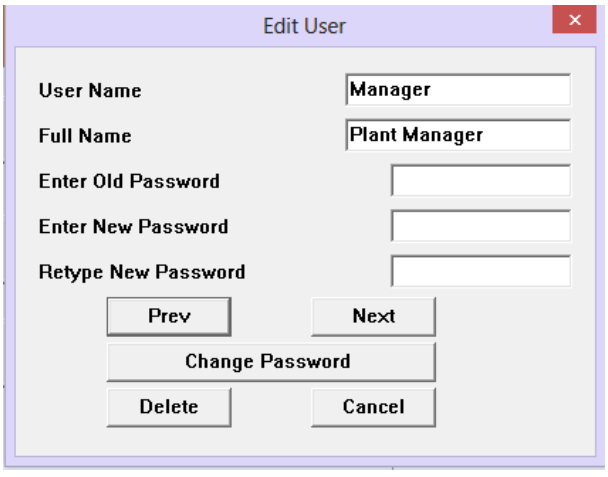
Gambar 3.46 Tampilan program *user create* setelah dijalankan

c. Perancangan program *user edit form*

User edit form ini berfungsi untuk menghapus *username* dan *password* yang sudah terdapftar pada sistem SCADA. Selain itu menu ini juga dapat digunakan untuk mengganti *password*. Fungsi yang digunakan untuk membuat program ini yaitu "*UserEditForm*", fungsi ini akan menjalankan program user edit form dengan cara memunculkan entri data *username* dan *password* yang telah dibuat di *database* setelah program ini dijalankan. Gambar 3.47 merupakan fungsi program yang digunakan pada user edit form, sedangkan Gambar 3.48 merupakan hasil program *user edit form* setelah dijalankan.



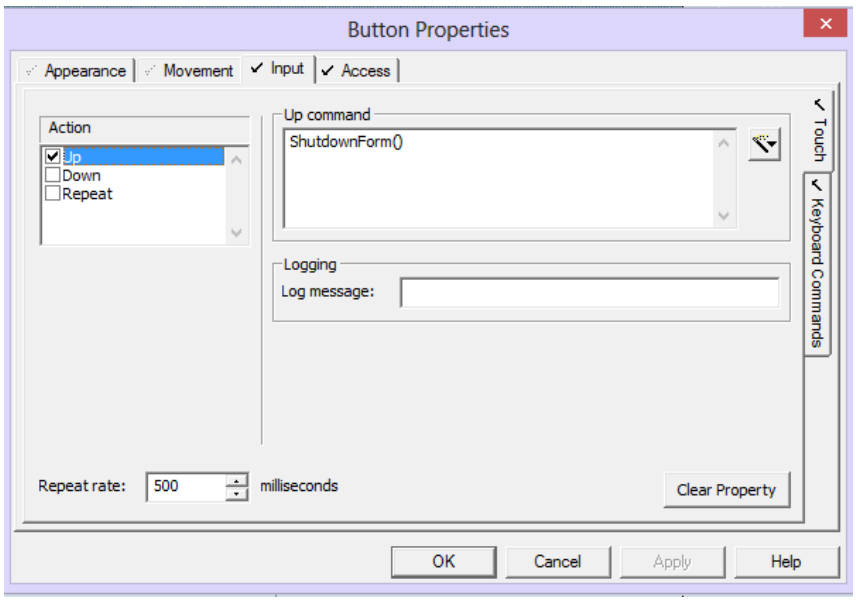
Gambar 3.47 Program *user edit form*



Gambar 3.48 Tampilan program *user edit form* setelah dijalankan

d. Perancangan program *shutdown*

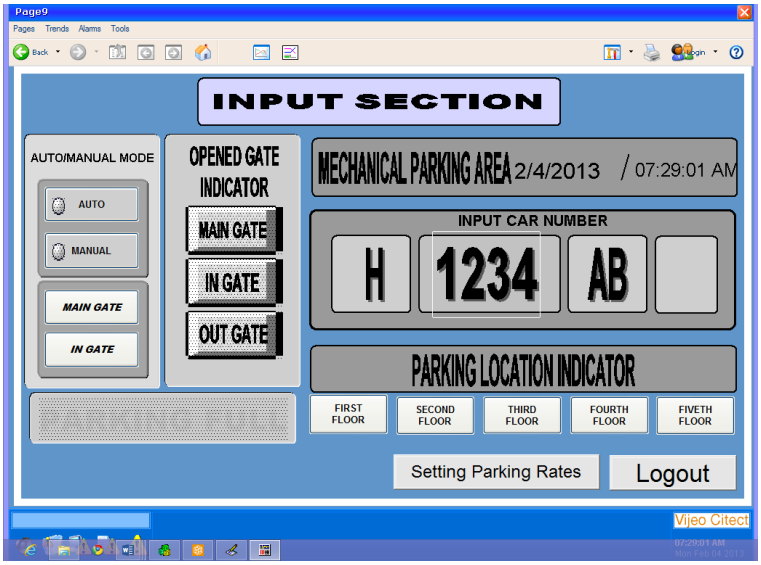
Menu *shutdown* ini berfungsi untuk mengakhiri semua proses atau untuk menonaktifkan sistem SCADA. Fungsi perintah yang digunakan untuk membuat program ini yaitu "*ShutdownForm*", fungsi ini akan menjalankan program shutdown dengan cara menonaktifkan semua sistem SCADA yang telah dijalankan pada *citect runtime manager*. Gambar 3.49 merupakan fungsi program yang digunakan pada program *shutdown*.



Gambar 3.49 Program *shutdown*

3.4.5.1.2 Perancangan Program *Human Machine Interface* (HMI)

Gambar 3.50 dibawah ini merupakan desain tampilan *graphic* dari program HMI yang akan dirancang. Dimana ada tujuh bagian utama program yang dibuat yaitu kendali *gate*, *indicator gate*, *input car number*, *parking location indicator*, *full parking indicator*, *log out*, dan *parking fee* seperti yang ditunjukkan pada Gambar 3.50. Program-program tersebut hanya dapat berjalan ketika sistem SCADA terkoneksi dengan PLC. Dimana program yang dibuat tersebut akan memerintahkan kerja pada PLC dan selanjutnya PLC akan menjalankan perintah tersebut untuk kemudian mengirimkan data yang telah didapat ke sistem SCADA.

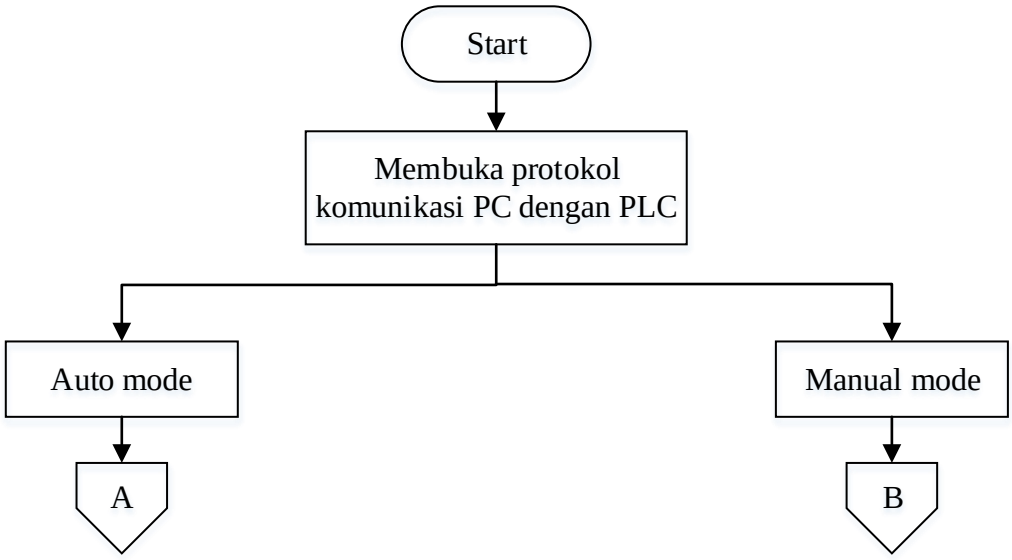


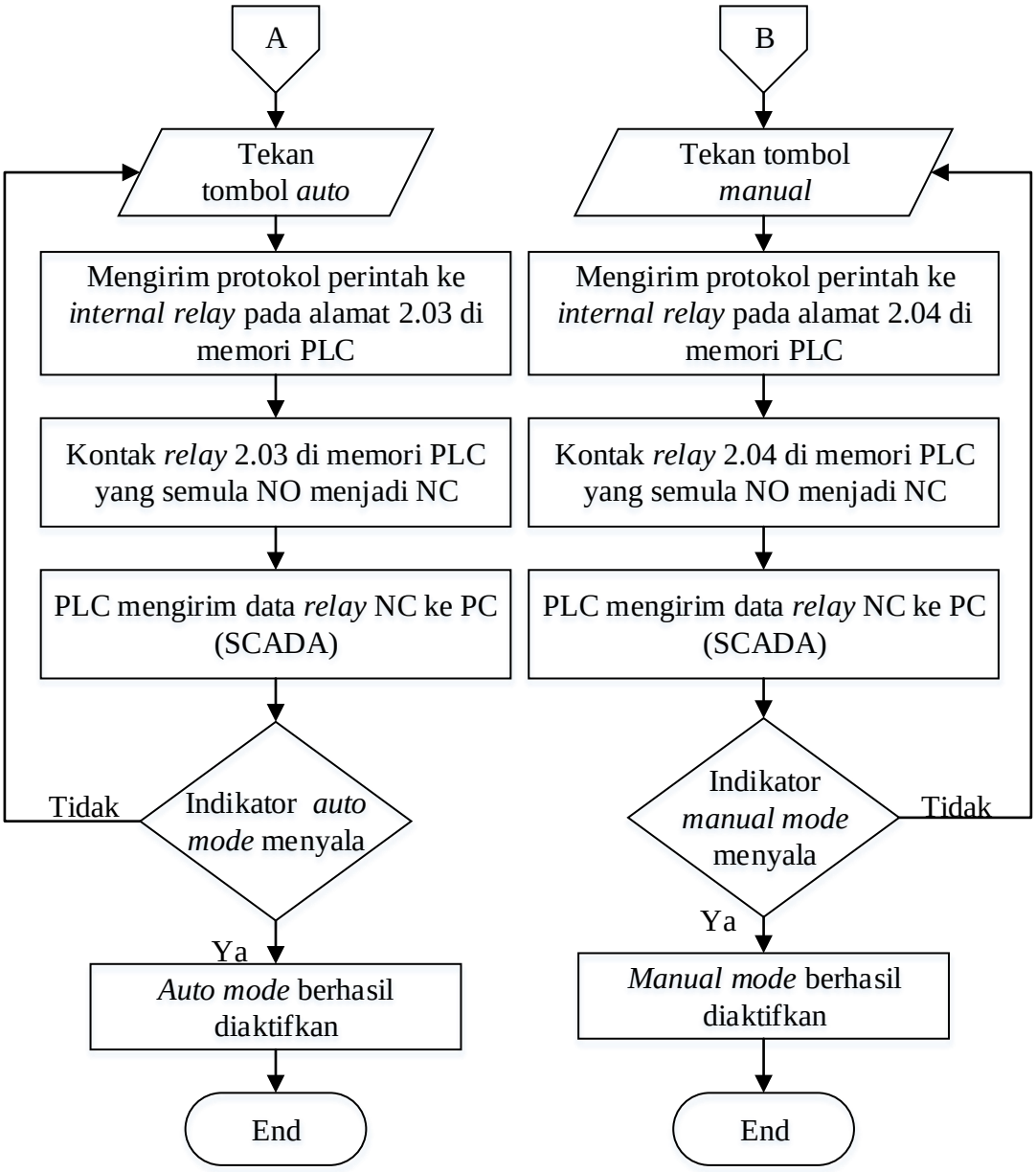
Gambar 3.50 Tampilan HMI *input section*

Berikut ini merupakan program-program yang telah dibuat pada HMI untuk mengendalikan sistem parkir:

a. Program kendali *auto mode* dan *manual mode gate* parkir

Untuk memerintahkan PLC melakukan proses kendali *gate* parkir, maka PC akan mengirimkan *protocol* perintah ke PLC untuk selanjutnya PLC mengirimkan datanya tersebut ke PC. Gambar 3.51 adalah diagram alir dari program proses kendali *auto manual gate* parkir.





Gambar 3.51 Proses kendali *gate* parkir *auto mode* dan *manual mode*

Berikut ini adalah urutan langkah kerja proses kendali *gate* parkir pada *auto mode*:

1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Setelah PC dan PLC terkoneksi, selanjutnya memilih mode otomatis (*auto*) dengan cara menekan *push button auto mode*.
3. Kemudian PC (SCADA) akan mengirim protokol perintah ke memori PLC, dimana alamat yang dituju adalah kontak *relay* 2.03 pada *ladder diagram* yang telah dibuat.

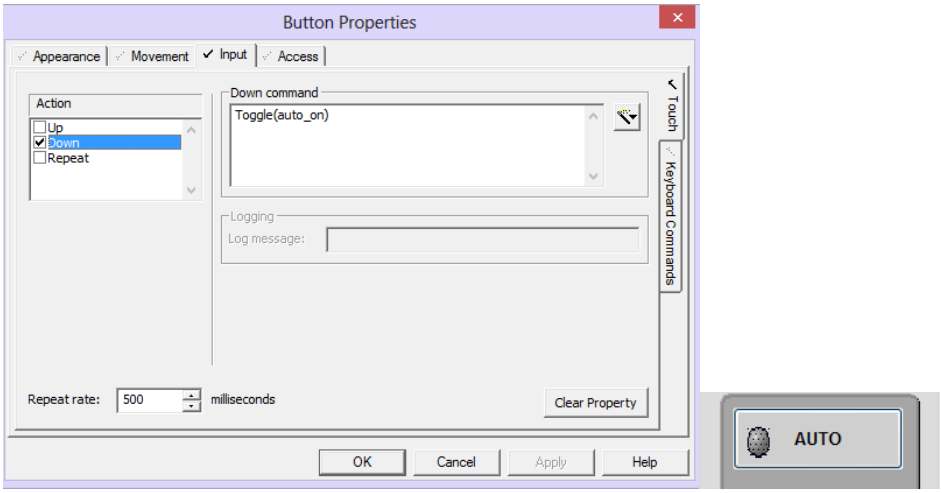
4. PLC akan menjalankan perintah tersebut sehingga kontak relay 2.03 *pada ladder diagram* yang semula NO berubah menjadi NC.
5. Perubahan data kontak relay 2.03 tersebut selanjutnya dikirim PLC ke PC (SCADA).
6. Jika data kontak relay 2.03 tersebut sampai ke SCADA dan sesuai dengan alamat yang dituliskan pada indikator, maka indikator mode auto akan menyala.
7. Setelah indikator auto mode menyala, maka proses untuk mengaktifkan kendali gate mode otomatis selesai. Sehingga gate parkir akan membuka dan menutup secara otomatis menurut perintah dari sensor fiber optic.

Berikut ini adalah urutan langkah kerja proses kendali *gate* parkir pada *auto mode*:

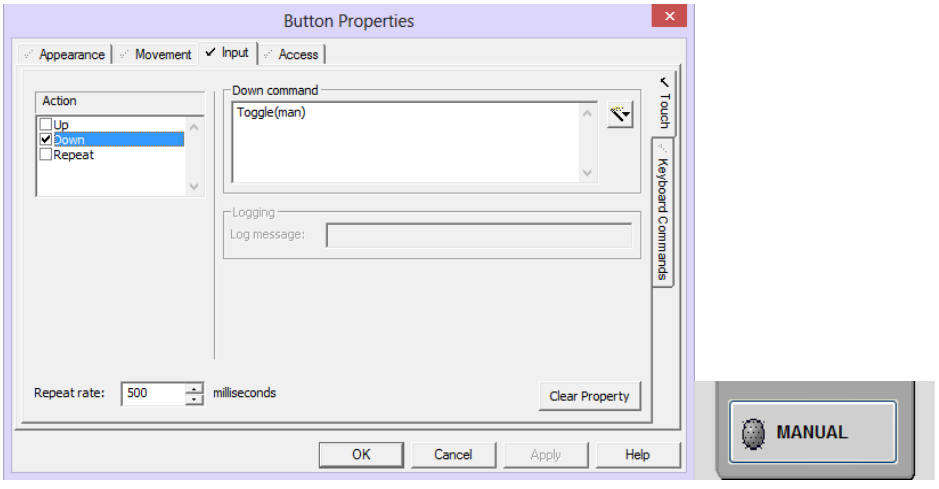
1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Setelah PC dan PLC terkoneksi, selanjutnya memilih mode manual dengan cara menekan *push button manual mode*.
3. Kemudian PC (SCADA) akan mengirim protokol perintah ke memori PLC, dimana alamat yang dituju adalah kontak relay 2.04 pada *ladder diagram* yang telah dibuat.
4. PLC akan menjalankan perintah tersebut sehingga kontak relay 2.04 pada *ladder diagram* yang semula NO berubah menjadi NC.
5. Perubahan data kontak relay 2.0 tersebut selanjutnya dikirim PLC ke PC (SCADA).
6. Jika data kontak relay 2.04 tersebut sampai ke SCADA dan sesuai dengan alamat yang dituliskan pada indikator, maka indikator *manual mode* akan menyala.
7. Setelah indikator *manual mode* menyala, maka proses untuk mengaktifkan kendali *gate manual mode* selesai. Sehingga *gate* parkir akan membuka dan menutup secara manual menurut perintah *push button gate* pada sistem SCADA yang ditekan oleh operator.

8. Jika kendali *gate* parkir *manual mode* diaktifkan maka secara otomatis kendali *gate* parkir *auto mode* akan tidak aktif, begitu pula sebaliknya jika *auto mode* aktif maka *manual mode* juga akan tidak aktif.

Gambar 3.52 dan 3.53 merupakan program untuk melakukan proses kendali *gate* secara otomatis dan manual.



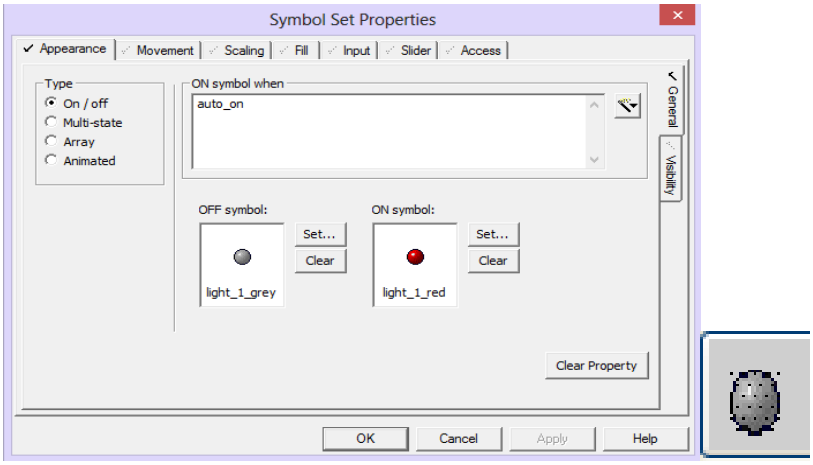
Gambar 3.52 Program *push button* kendali *gate* mode otomatis



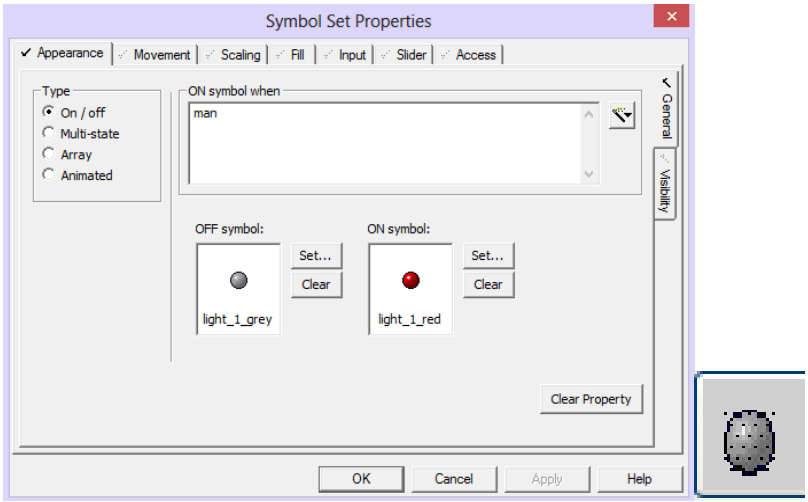
Gambar 3.53 Program *push button kendali gate* mode manual

Gambar 3.52 dan 3.53 merupakan program yang digunakan untuk mengatur mode kendali gate secara otomatis maupun secara manual. Gambar 3.52 adalah program

untuk mengontrol kendali gate secara otomatis dengan perintah fungsi "*Toggle()*" yang terdapat pada perangkat lunak Vijeo Citect kemudian dilanjutkan dengan memasukkan *variable tags* "*auto_on*" dari Table 3.3. Sedangkan pada Gambar 3.53 merupakan program kendali *gate* secara manual yang juga menggunakan perintah fungsi "*Toggle()*" tetapi menggunakan nama variable tags "*man*". Perintah "*Toggle()*" berfungsi untuk mengaktifkan *variable tag* yang terdapat didalam fungsi perintah "*Toggle()*" tersebut. Gambar 3.54 dan 3.55 adalah program yang digunakan untuk membuat indikator kendali *gate* tersebut.



Gambar 3.54 Program indikator kendali *gate* mode otomatis

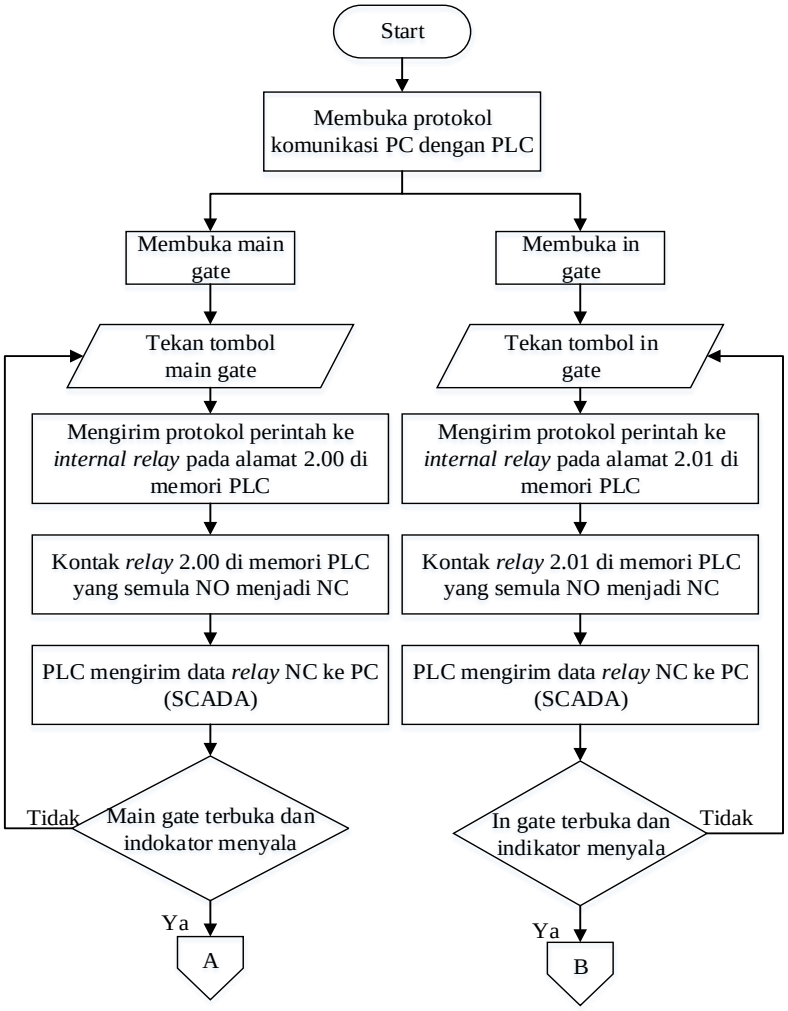


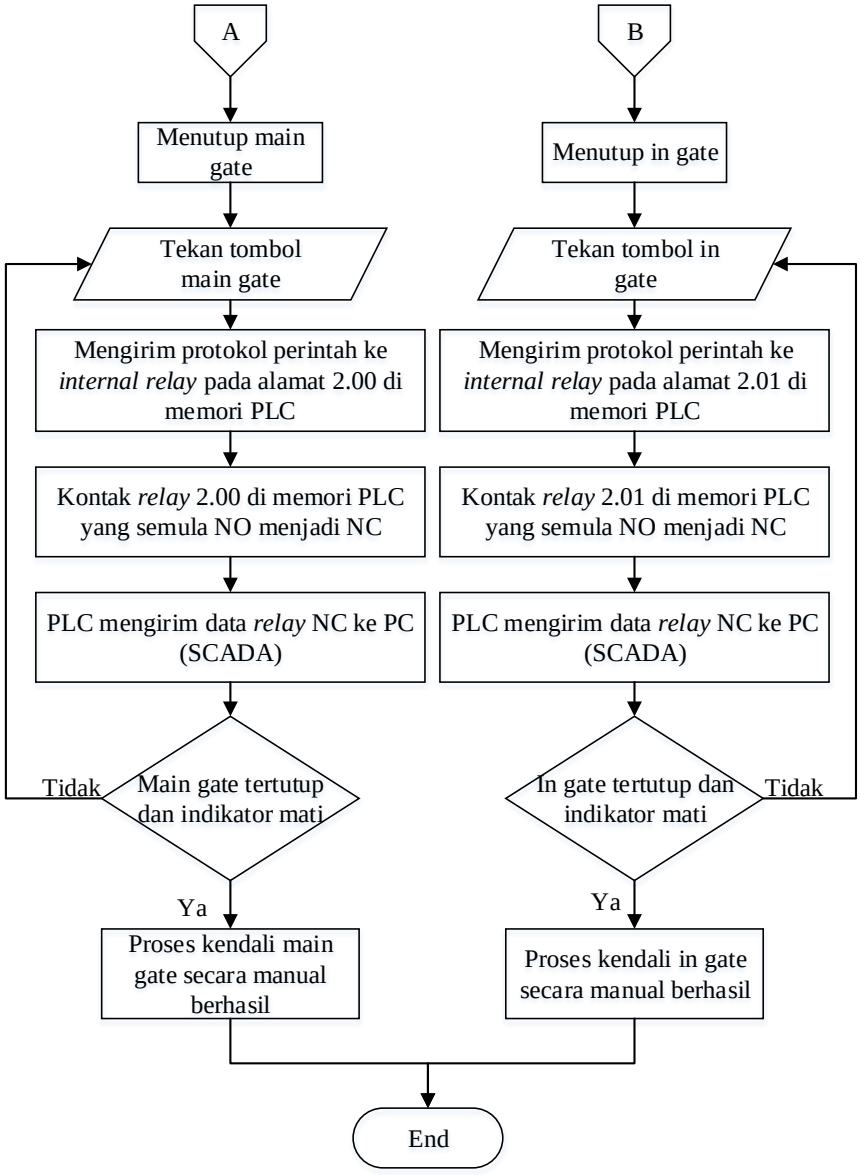
Gambar 3.55 Program indikator kendali *gate* mode manual

Pada Gambar 3.54 dan 3.55 merupakan program indikator kendali *gate* untuk mode otomatis dan manual, perintah fungsi yang digunakan ialah dengan memasukkan *variable tags* "*auto_on*" untuk mode otomatis dan *variable tags* "*man*" untuk mode manual. Kedua indikator tersebut menggunakan simbol lampu berwarna silver pada keadaan non aktif sedangkan simbol lampu berwarna merah pada saat keadaan aktif.

b. Program kendali gate secara manual (*push button main gate* dan *in gate*)

Untuk memerintahkan PLC melakukan proses kendali *gate* parkir secara manual dengan menggunakan *push button main gate* dan *in gate*, maka PC akan mengirimkan protocol perintah ke PLC dan selanjutnya PLC mengirimkan datanya tersebut ke PC. Gambar 3.56 adalah diagram alir dari program proses kendali *gate* secara manual.





Gambar 3.56 Diagram alir program proses kendali *gate* secara manual

Berikut ini adalah urutan langkah kerja proses kendali *main gate* parkir secara manual:

1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Setelah PC dan PLC terkoneksi, selanjutnya untuk membuka *main gate* secara manual dengan cara menekan *push button main gate*.

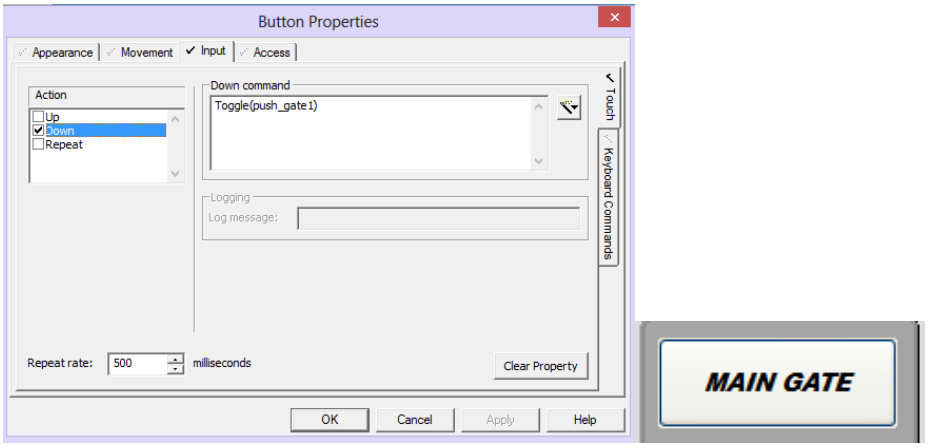
3. Kemudian PC (SCADA) akan mengirim protokol perintah ke memori PLC, dimana alamat yang dituju adalah kontak relay 2.00 pada *ladder diagram* yang telah dibuat pada memori PLC.
4. PLC akan menjalankan perintah tersebut sehingga kontak relay 2.00 pada *ladder diagram* yang semula NO berubah menjadi NC.
5. Perubahan data kontak relay 2.00 tersebut selanjutnya dikirim PLC ke PC (SCADA).
6. Jika data kontak relay 2.00 tersebut sampai ke SCADA dan sesuai dengan alamat yang dituliskan pada indikator, maka indikator *main gate* akan menyala dan *main gate* akan terbuka.
7. Untuk selanjutnya menutup *main gate* secara manual dengan cara menekan kembali *push button main gate*.
8. PC (SCADA) akan mengirim protokol perintah ke memori PLC, dimana alamat yang dituju adalah kontak relay 2.00 pada *ladder diagram* yang telah dibuat pada memori PLC.
9. PLC akan menjalankan perintah tersebut sehingga kontak relay 2.00 pada *ladder diagram* yang semula NC berubah menjadi NO.
10. Perubahan data kontak relay 2.00 tersebut selanjutnya dikirim PLC ke PC (SCADA).
11. Jika data kontak relay 2.00 tersebut sampai ke SCADA dan sesuai dengan alamat yang dituliskan pada indikator, maka indikator *main gate* akan mati dan *main gate* tertutup.

Berikut ini adalah urutan langkah kerja proses kendali *in gate* parkir secara manual:

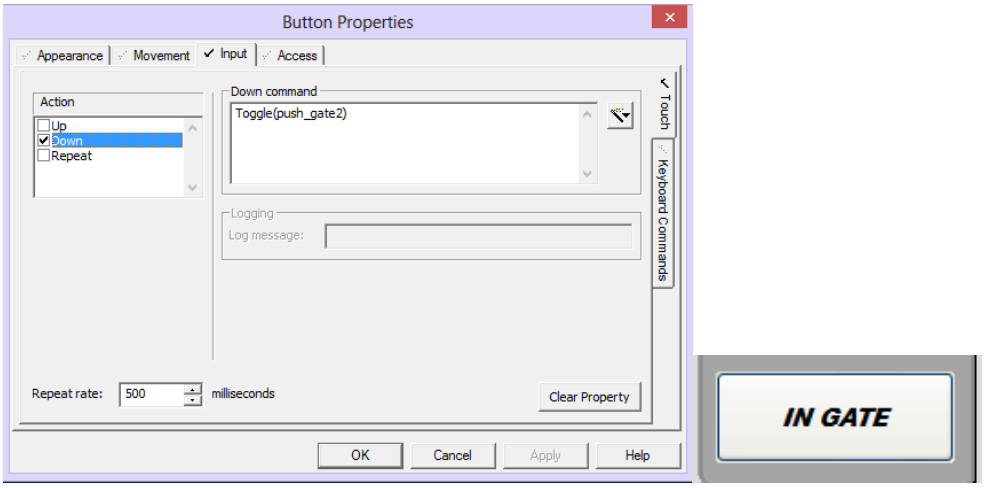
1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Setelah PC dan PLC terkoneksi, selanjutnya membuka *in gate* secara manual dengan cara menekan *push button in gate*.
3. Kemudian PC (SCADA) akan mengirim protokol perintah ke memori PLC, dimana alamat yang dituju adalah kontak relay 2.01 pada *ladder diagram* yang telah dibuat pada memori PLC.

4. PLC akan menjalankan perintah tersebut sehingga kontak relay 2.01 pada *ladder diagram* yang semula NO berubah menjadi NC.
5. Perubahan data kontak relay 2.01 tersebut selanjutnya dikirim PLC ke PC (SCADA).
6. Jika data kontak relay 2.01 tersebut sampai ke SCADA dan sesuai dengan alamat yang dituliskan pada indikator, maka indikator *in gate* akan menyala dan *in gate* akan terbuka.
7. Untuk selanjutnya menutup *in gate* secara manual dengan cara menekan kembali *push button in gate* .
8. PC (SCADA) akan mengirim protokol perintah ke memori PLC, dimana alamat yang dituju adalah kontak relay 2.01 pada *ladder diagram* yang telah dibuat pada memori PLC.
9. PLC akan menjalankan perintah tersebut sehingga kontak relay 2.01 pada *ladder diagram* yang semula NO berubah menjadi NC.
10. Perubahan data kontak relay 2.01 tersebut selanjutnya dikirim PLC ke PC (SCADA).
11. Jika data kontak relay 2.01 tersebut sampai ke SCADA dan sesuai dengan alamat yang dituliskan pada indikator, maka indikator *in gate* akan mati dan *in gate* akan tertutup.

Gambar 3.57 dan 3.58 merupakan program *push button* untuk melakukan proses kendali *main gate* dan *in gate* secara manual.



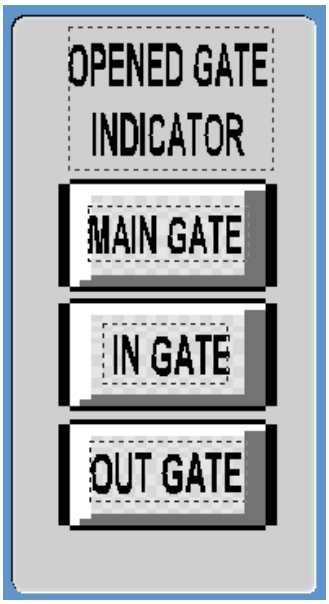
Gambar 3.57 Program *push button* untuk kendali *main gate* secara manual



Gambar 3.58 Program *push button* untuk kendali *in gate* secara manual

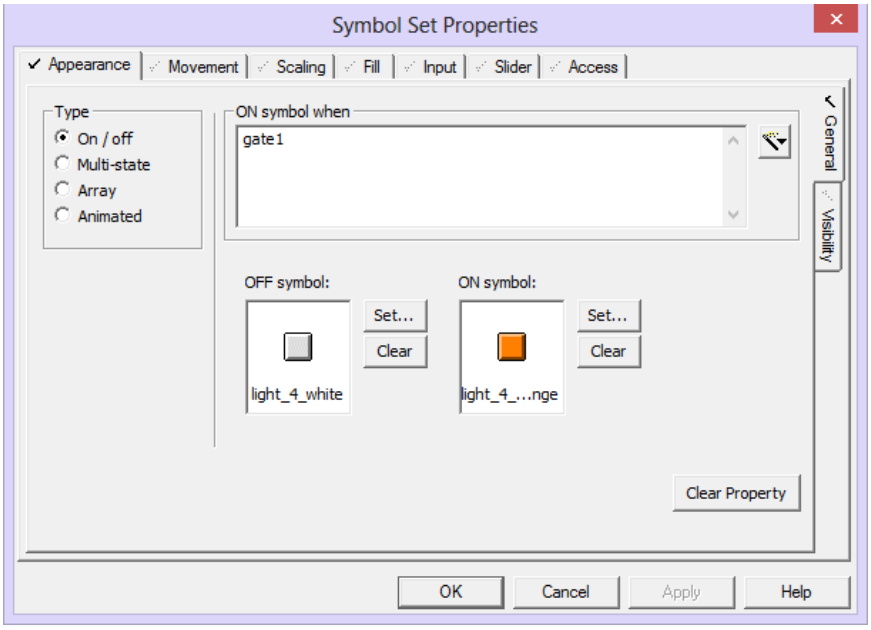
c. Program indikator jika *gate* parkir terbuka (*opened gate indicator*)

Opened gate indicator berfungsi untuk memudahkan operator memonitoring keadaan *gate* parkir dalam keadaan terbuka maupun tertutup. Ada tiga indikator yang terdapat pada program tersebut yaitu indikator *main gate*, indikator *in gate*, dan indikator *out gate* seperti yang ditunjukkan pada Gambar 3.59. Lampu indikator tersebut akan menyala jika *gate* parkir dalam keadaan terbuka sedangkan lampu indikator akan mati jika *gate* parkir dalam keadaan tertutup.

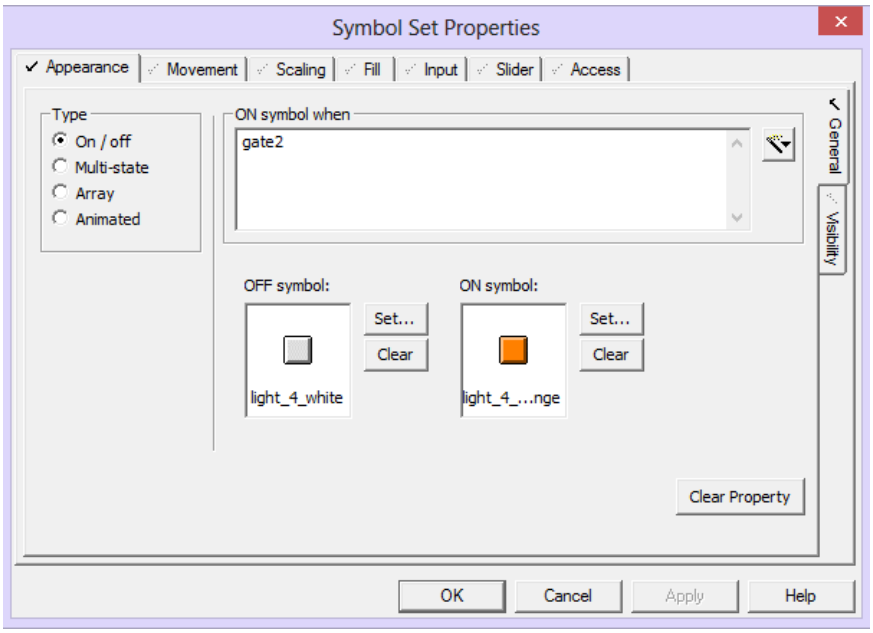


Gambar 3.59 *Opened gate indicator*

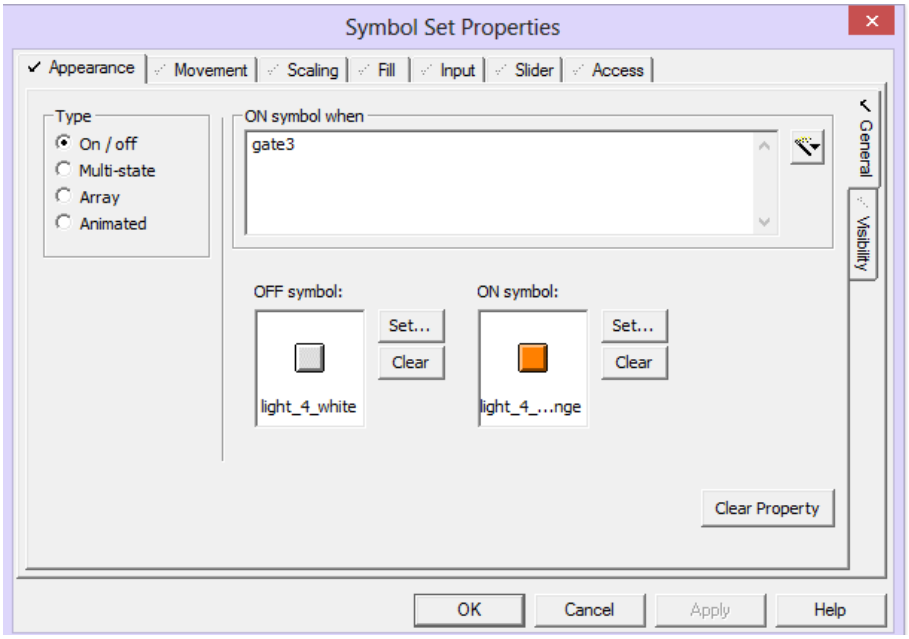
Gambar 3.60, 3.61, dan 3.62 merupakan program yang digunakan untuk membuat *opened gate indicator* tersebut.



Gambar 3.60 Program indikator *main gate*



Gambar 3.61 Program indikator *in gate*

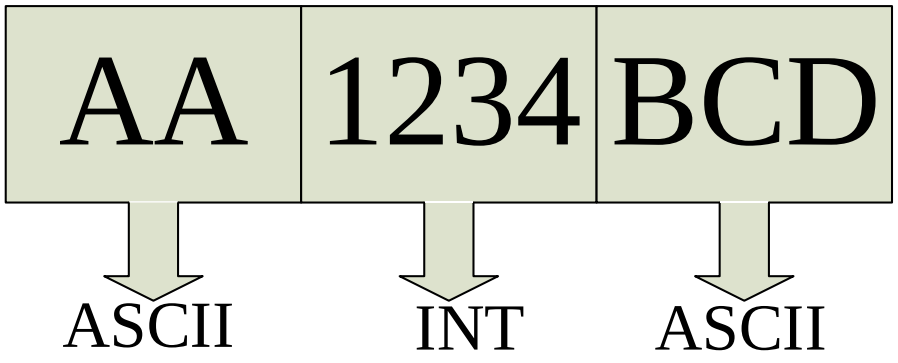


Gambar 3.62 Program indikator *out gate*

Pada Gambar 3.60, 3.61 dan 3.62 merupakan program indikator *gate* parkir, fungsi perintah yang digunakan untuk menjalankan indikator tersebut ialah dengan memasukkan *variable tags* "*gate1*" untuk indikator *main gate* dan *variable tags* "*gate2*" untuk indikator *in gate*, sedangkan untuk indikator *out gate variable tags* yang digunkanan yaitu "*gate 3*". Ketiga indikator tersebut menggunakan simbol lampu berwarna silver jika gate dalam keadaan tertutup sedangkan simbol lampu berwarna oranye jika gate dalam keadaan terbuka.

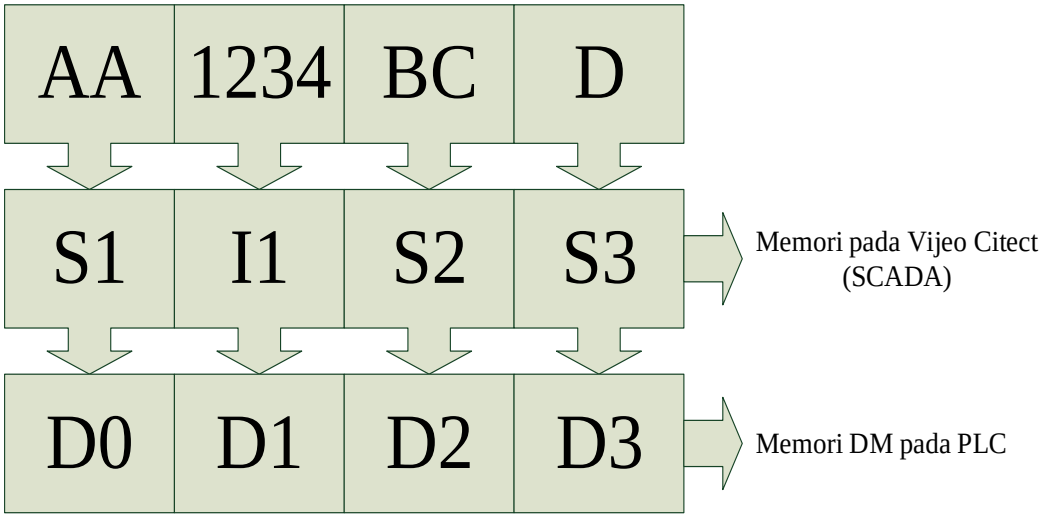
d. Program untuk memasukkan nomor kendaraan (*input car number*)

Pada sistem parkir ini media atau data yang digunakan untuk registrasi saat mobil akan masuk maupun keluar lokasi parkir ialah nomor kendaraan. Nomor kendaraan tersebut dimasukkan pada sistem SCADA dan kemudian dikirim ke PLC yang kemudian di simpan pada memori DM. Dalam perencanaan program ini, nomor kendaraan tersebut dikategorikan dalam dua karakter, yaitu untuk huruf dimasukkan dalam kategori karakter ASCII (*American Standart Code Information Interchange*) sedangkan untuk nomor dimasukkan kategori INTERGER (INT) seperti yang ditunjukkan pada Gambar 3.63.



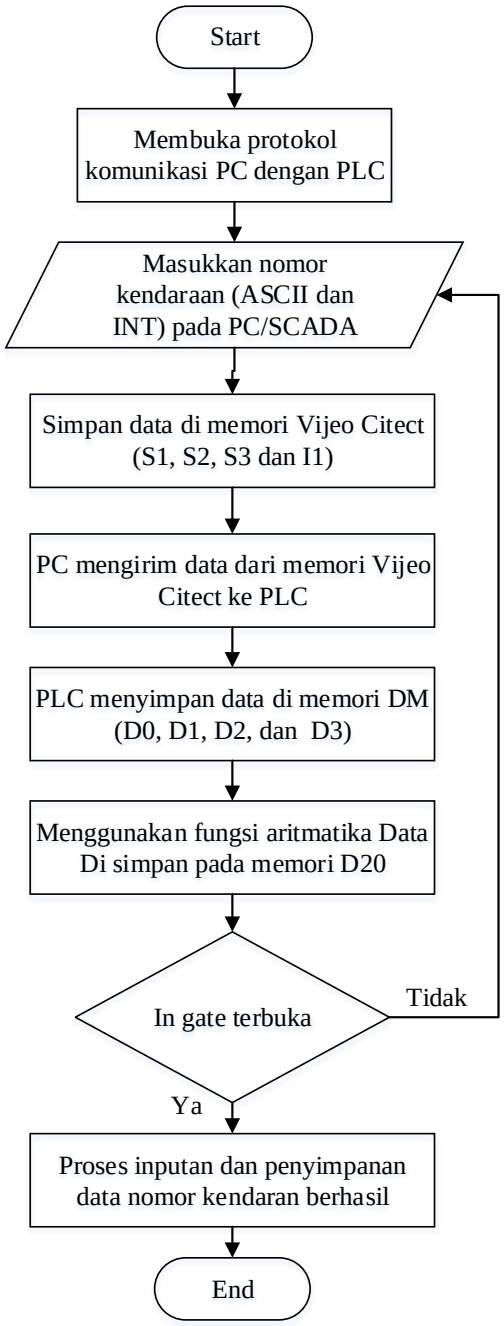
Gambar 3.63 Pengelompokan karakter kode bilangan data nomor kendaraan

Dari Gambar 3.63 di atas dapat dilihat bahwa ada dua kelompok kategori karakter ASCII dan satu kelompok kategori karakter INT. Pada karakter ASCII yang terdiri dari dua digit huruf depan dan tiga digit huruf belakang akan dibagi menjadi tiga bagian, karena dalam satu alamat memori DM pada PLC hanya bisa menyimpan dua digit karakter ASCII. Sedangkan untuk karakter INT yang terdiri dari empat digit angka dikelompokan menjadi satu, karena satu alamat memori DM pada PLC dapat menyimpan empat digit karakter bilangan INT. Gambar 3.64 merupakan pengelompokkan karakter ASCII dan INT tersebut serta pengalamatan memori pada SCADA dan PLC.



Gambar 3.64 Pengalamatan data nomor kendaraan pada memori SCADA dan PLC

Untuk memerintahkan PLC melakukan proses penyimpaman data seperti yang ditunjukkan pada Gambar 3.64, maka PC akan mengirimkan data nomor kendaraan ke PLC dan selanjutnya PLC menyimpan data tersebut ke memori. Gambar 3.65 adalah diagram alir dari program proses inputan dan penyimpanan data nomor kendaraan pada sistem SCADA dan PLC.

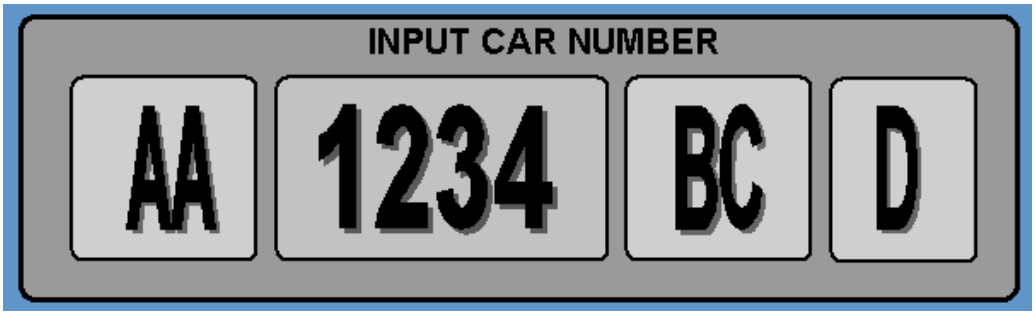


Gambar 3.65 Diagram alir proses inputan dan penyimpanan data nomor kendaraan

Berikut ini adalah urutan langkah kerja proses kendali *gate* parkir pada *auto mode*:

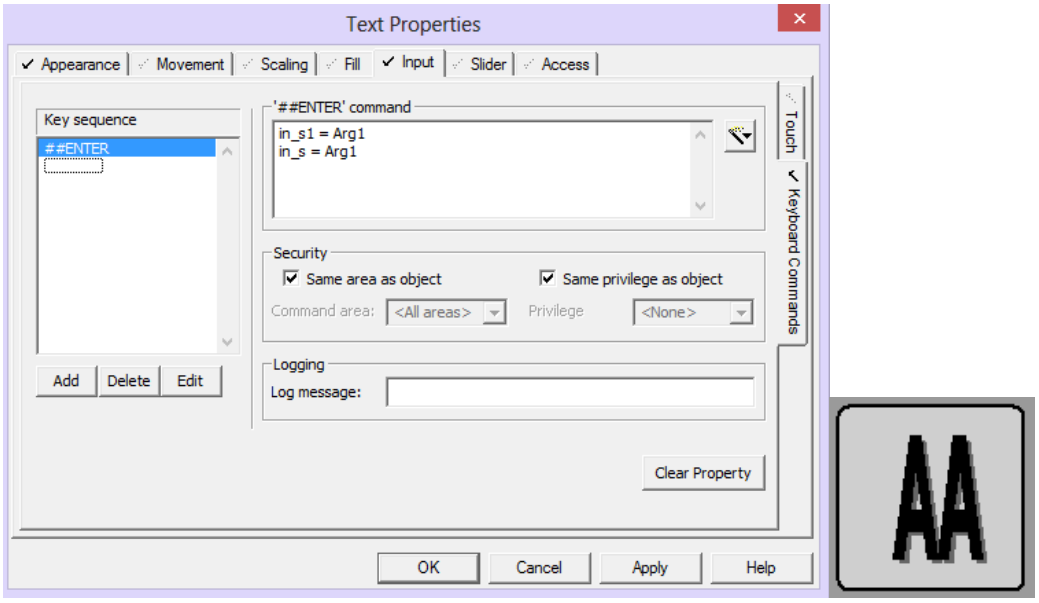
1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Setelah PC dan PLC terkoneksi, selanjutnya nomor kendaraan dimasukkan sesuai dengan kategorinya (ASCII dan INT).
3. Kemudian data nomor kendaraan yang telah masuk disimpan pada memori SCADA untuk karakter ASCII disimpan pada memori S1, S2, dan S3 (STRING). Sedangkan untuk karakter INT disimpan di memori I1 (INTEGER).
4. Data nomor kendaraan yang telah disimpan pada memori SCADA dikirim ke PLC.
5. Selanjutnya PLC menyimpan data nomor kendaraan pada memori DM (D0, D1, D2, dan D3) sesuai yang ditunjukkan pada gambar 3.53 diatas.
6. Untuk selanjutnya data pada memori DM (D0, D1, D2, dan D3) kemudian dijumlahkan menggunakan program PLC dan hasilnya di simpan pada memori D20.
7. Setelah semua data nomor kendaraan tersimpan pada memori PLC maka *in gate* parkir akan terbuka.
8. Jika *in gate* telah terbuka maka proses inputan dan penyimpanan data nomor kendaraan pada memori SCADA dan PLC berhasil.

Gambar 3.66 merupakan desain program perangkat lunak sistem parkir yang digunakan untuk memasukkan nomor kendaraan pada sistem SCADA.

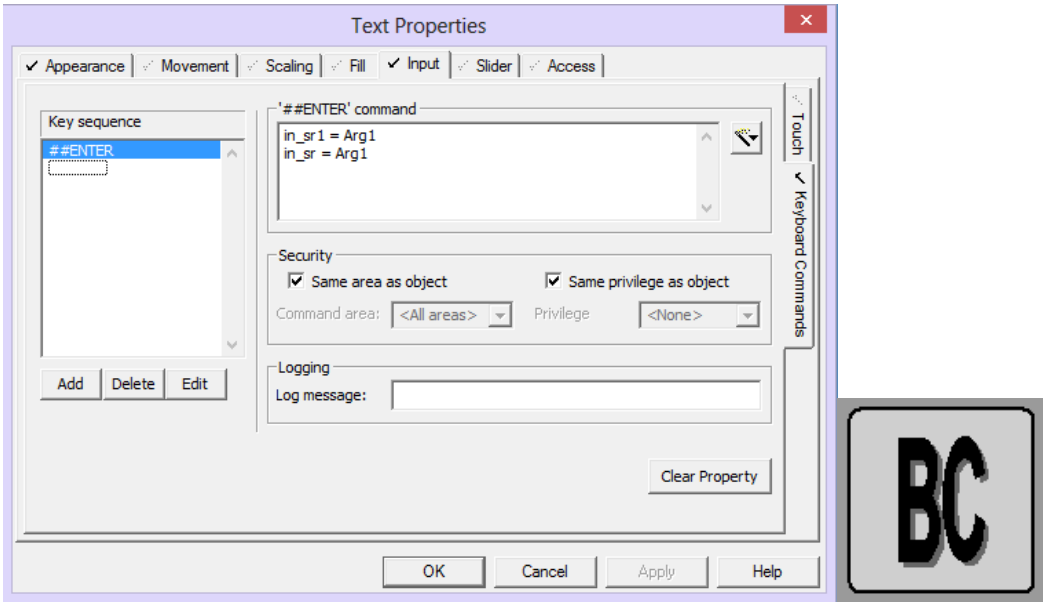


Gambar 3.66 Desain program sistem SCADA untuk masukkan nomor kendaraan

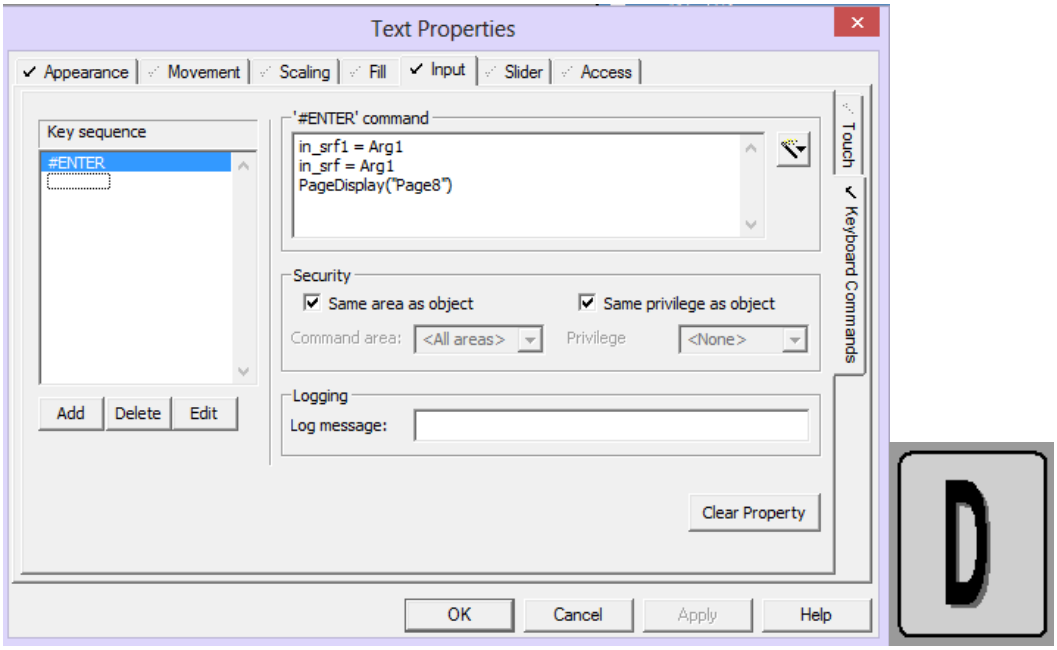
Gambar 3.67, 3.68, dan 3.69 merupakan program yang digunakan untuk membuat desain *input car number* untuk karakter ASCII.



Gambar 3.67 Program untuk memasukkan karakter ASCII huruf depan nomor kendaraan



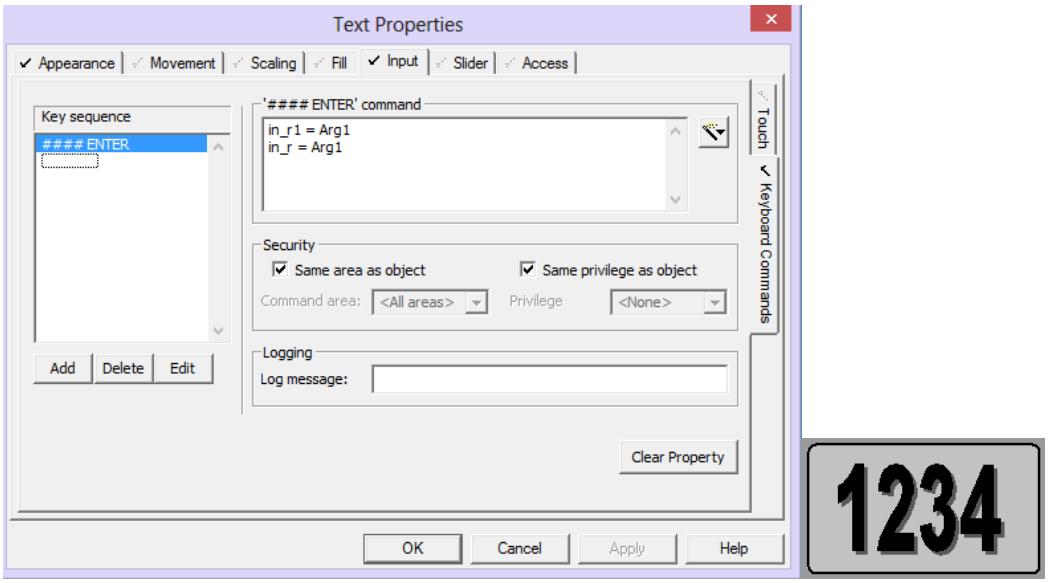
Gambar 3.68 Program untuk memasukkan karakter ASCII dua digit huruf belakang nomor kendaraan



Gambar 3.69 Program untuk memasukkan karakter ASCII satu digit terakhir huruf belakang nomor kendaraan

Langkah pertama yang dilakukan dalam membuat program untuk memasukkan karakter ASCII nomor kendaraan yaitu dengan memasukkan banyaknya digit yang akan digunakan pada menu *key sequence*. Pada Gambar 3.56 dan 3.57 akan menggunakan dua digit sehingga memakai dua kode "*#*" dan diakhiri dengan *key sequence* "*enter*". Selanjutnya untuk menu *command*, perintah fungsi yang digunakan untuk menjalankan program tersebut ialah dengan memasukkan *variable tags name* "*in_s1 = arg1* dan *in_sr1 = arg1*" untuk memerintahkan memori SCADA menyimpan data dan *variable tags name* "*in_s = arg1* dan *in_sr = arg1*" untuk memerintahkan memori PLC menerima serta menyimpan data. Sedangkan pada Gambar 3.58 di atas merupakan program untuk memasukkan satu digit terakhir huruf belakang nomor kendaraan, sehingga pada menu *key sequence* menggunakan satu kode "*#*" dan diakhiri dengan *key sequence* "*enter*". Untuk menjalankan program tersebut *variable tags name* yang digunakan ialah "*in_srf1 = arg1*" sebagai perintah memori SCADA menyimpan data dan *varable tags name* "*in_srf = arg1*" sebagai perintah memori PLC menerima serta menyimpan data. Fungsi perintah "*pagedisplay("page8")*" merupakan perintah untuk menampilkan

halaman delapan yang merupakan karcis parkir untuk menampilkan lokasi parkir. Gambar 3.70 merupakan program untuk memasukkan karakter INT untuk kelompok angka pada nomor kendaraan.



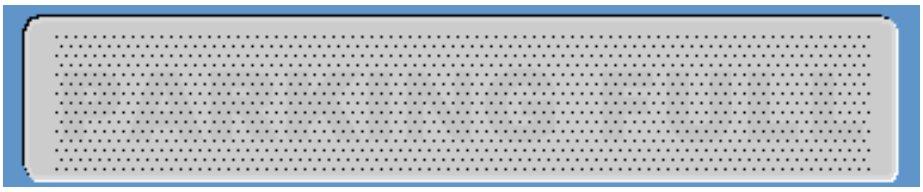
Gambar 3.70 Program untuk memasukkan karakter INT kelompok angka pada nomor kendaraan

Untuk membuat program seperti pada Gambar 3.70 langkah pertama yang dilakukan yaitu memasukkan banyaknya digit yang akan dipakai pada menu *key sequence*. Karena pada program ini jumlah digit yang akan dipakai adalah empat digit, maka menggunakan empat kode "#" kemudian diakhiri dengan *key sequence* "*enter*". Sedangkan untuk menjalankan program tersebut menggunakan perintah fungsi *variable tags name* "*in_r1 = arg1*" sebagai perintah memori SCADA menyimpan data dan *variable tags name* "*in_r = arg1*" sebagai perintah memori PLC melakukan penerimaan dan penyimpanan data.

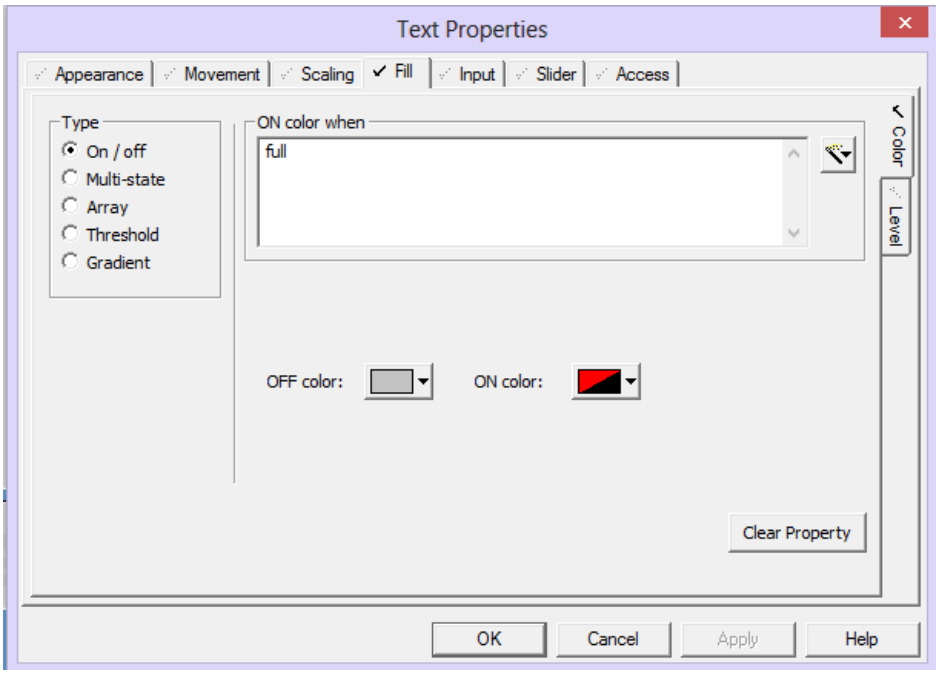
e. Program indikator jika area parkir penuh (*full parking indicator*)

Full parking indicator berfungsi untuk memudahkan operator memonitoring keadaan parkir jika semua lokasi parkir sudah terisi atau semua lokasi parkir penuh. Lampu indikator tersebut akan menyala jika indikator parkir penuh dari lantai satu

sampai lantai lima menyala. Gambar 3.71 merupakan tampilan *full parking indikator* sedangkan Gambar 3.72 adalah program *full parking indicator* yang telah dirancang pada perangkat lunak sistem SCADA.



Gambar 3.71 Tampilan *full parking indicator*



Gambar 3.72 Program *full parking indicator*

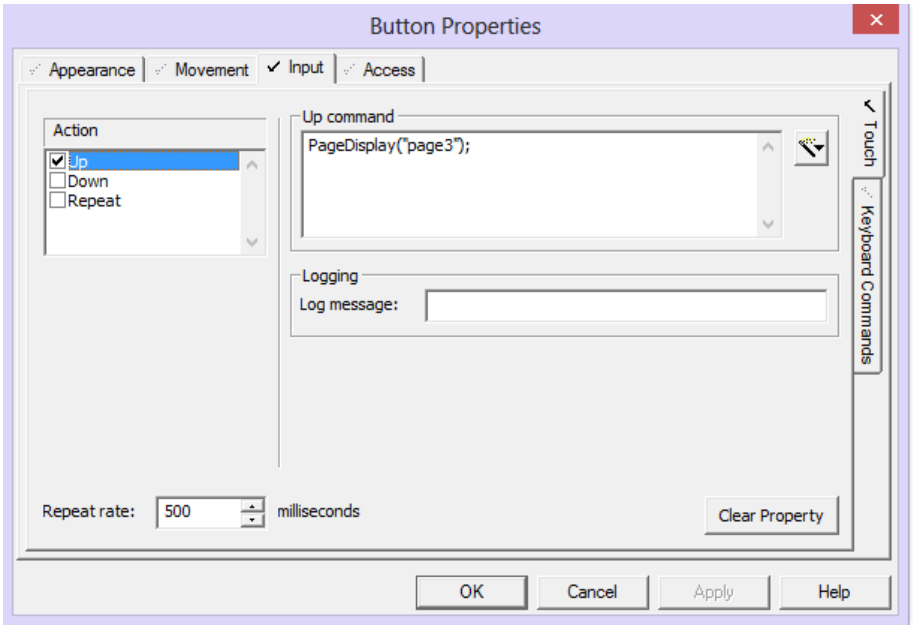
Gambar 3.72 merupakan program untuk membuat *full parking indicator*, fungsi perintah yang digunakan ialah dengan memasukkan *variable tags name* "*full*" pada menu *appearance*. Kemudian menggunakan *type on* atau *off* untuk mengoperasikan indikator tersebut. Teks "*parking full*" pada Gambar 3.71 akan berwarna merah jika dalam keadaan aktif dan berwarna silver dalam keadaan non aktif.

f. Program indikator setiap lantai (*parking location indicator*)

Parking location indicator berfungsi untuk memudahkan operator memonitoring keadaan lokasi parkir pada setiap lantai. Ada lima lokasi indikator yang telah dibuat, yaitu indikator lantai 1, lantai 2, lanta 3, lantai 4, dan lantai 5 seperti yang ditunjukkan Gambar 3.73.

Gambar 3.73 Tampilan *push button parking location indicator*

Gambar 3.74 merupakan program yang digunakan untuk membuat *push button parking location indicator* tersebut.



Gambar 3.74 Program *push button parking location indicator* lantai satu

Gambar 3.74 merupakan program *push button parking location indicator* untuk lantai satu, fungsi perintah yang digunakan untuk menjalankan program tersebut ialah "*PageDisplay("page3");*". Prinsip kerja dari program tersebut adalah jika

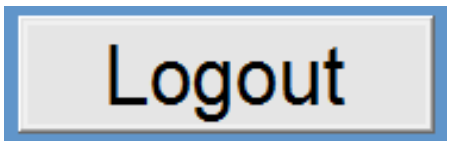
push button first floor pada Gambar 3.73 ditekan maka akan keluar tampilan halaman tiga, dimana pada halaman tersebut terdapat indikator-indikator setiap lokasi parkir. Selanjutnya untuk membuat *push button parking locatin indicator* dari lantai dua sampai lantai lima menggunakan cara yang sama seperti langkah diatas, tetapi dengan memasukkan fungsi perintah yang berbeda. Fungsi perintah yang digunakan seperti ditunjukkan pada Tabel 3.4.

Tabel 3.4 Fungsi perintah *push button parking location indicator*

No	*Push button*	Fungsi Perintah
1	*First floor*	*PageDisplay("page3");*
2	*Second floor*	*PageDisplay("page4");*
3	*Third floor*	*PageDisplay("page5");*
4	*Fourth floor*	*PageDisplay("page6");*
5	*Fiveth floor*	*PageDisplay("page7");*

g. Program *user logout*

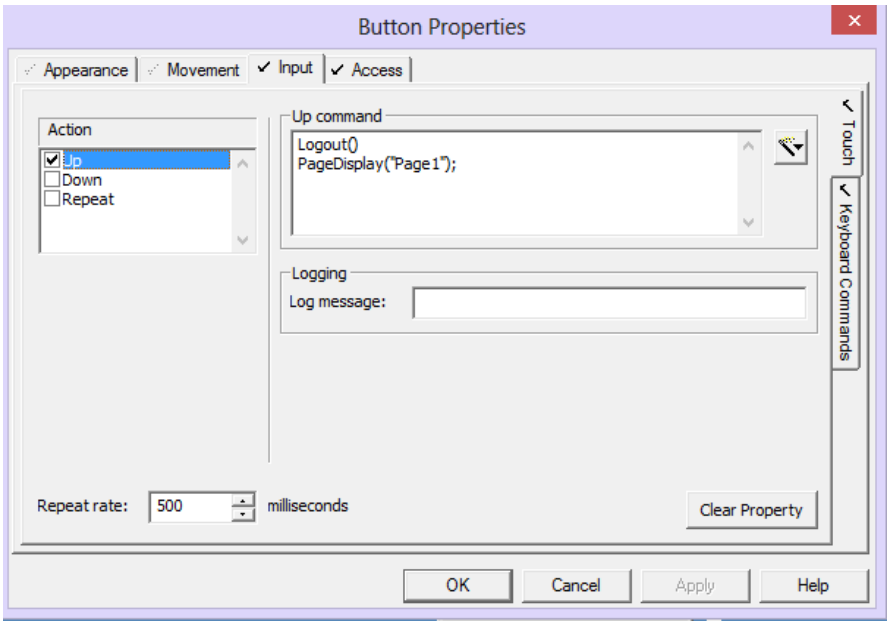
User logout berfungsi bagi operator untuk keluar dari halaman HMI dan kembali pada halaman awal yaitu halaman *user registration*. Tampilan *user logout* yang telah dibuat berupa *push button* seperti yang ditunjukkan pada Gambar 3.75.



Gambar 3.75 *Push button user logout*

Langkah – langkah untuk membuat program *user logout* yang pertama ialah mengisi fungsi perintah "*Logout()*" pada menu *up command*. Fungsi perintah tersebut digunakan sebagai perintah untuk menjalankan program agar *user* keluar dari sistem SCADA tersebut. Langkah kedua yaitu menambahkan fungsi perintah "*PageDisplay("page1");*" yang berfungsi untuk menampilkan halaman pertama yang merupakan halaman *user registration*. Prinsip kerja dari *push button user*

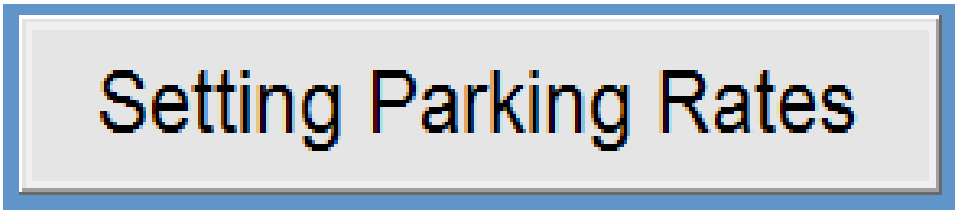
logout yaitu jika push button tersebut ditekan maka user akan keluar dari sistem HMI dan kembali pada halaman *user registration*. Gambar 3.76 merupakan program yang digunakan untuk membuat *push buttom user logout*.



Gambar 3.76 Program *user logout*

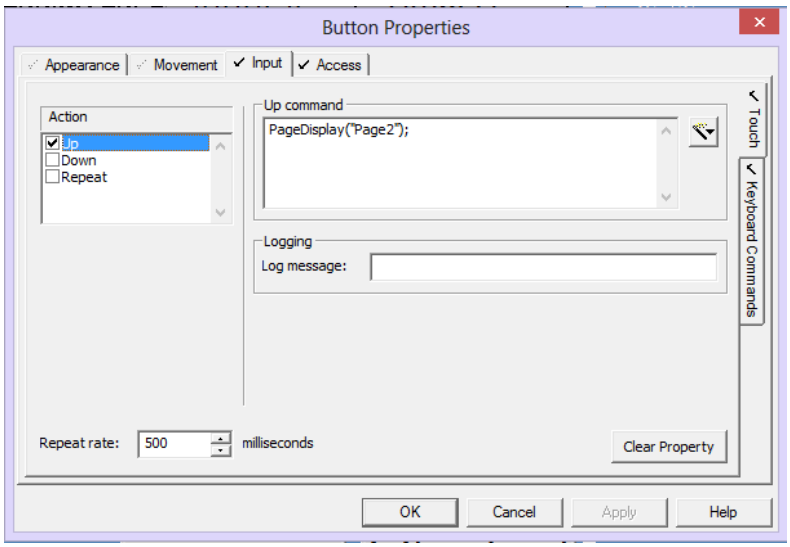
h. Program *push button parking rates*

Push button parking rates berfungsi bagi operator sebagai *shortcut* untuk membuka halaman dua yang merupakan halaman untuk memasukkan tarif parkir (*setting parking rates*). Tampilan *push button setting parking rates* yang telah dibuat dapat dilihat pada Gambar 3.77.



Gambar 3.77 *Push button setting parking rates*

Gambar 3.78 merupakan program yang digunakan untuk membuat *push button setting parking rates*.

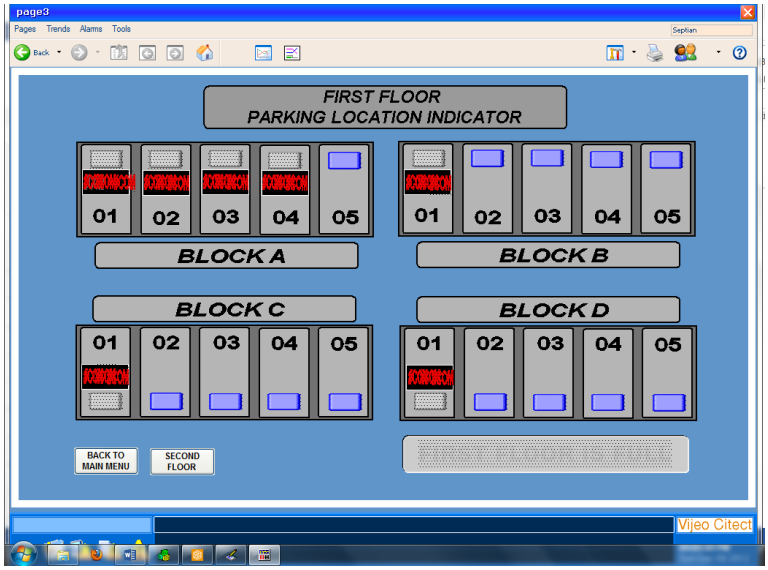


Gambar 3.78 Program *push button setting parking rates*

Gambar 3.78 merupakan program *push button setting parking rates* yang telah dibuat, fungsi perintah yang digunakan untuk menjalankan program tersebut ialah "*PageDisplay("page2");*". Prinsip kerja dari program tersebut adalah jika *push button setting parking rates* pada Gambar 3.77 ditekan maka akan keluar tampilan halaman dua yang merupakan halaman untuk memasukkan tarif biaya parkir.

3.4.5.1.3 Perancangan Program *Parking Location Indicator*

Parking location indicator terdiri dari lima halaman, dimana pada setiap halaman mewakili setiap lantai. Gambar 3.79 merupakan desain tampilan *graphic* dari program *parking location indicator* pada lantai satu yang telah dirancang. Pada halaman tersebut terdapat bagian-bagian utama program yang dibuat yaitu indikator dan *timer* pada setiap lokasi parkir serta indikator jika lokasi parkir pada lantai tersebut penuh. Indikator-indikator tersebut hanya dapat berjalan ketika sistem SCADA terkoneksi dengan PLC, dimana program yang dibuat tersebut akan mengakuisisi data yang terdapat di PLC dan kemudian menampilkanya pada sistem SCADA.

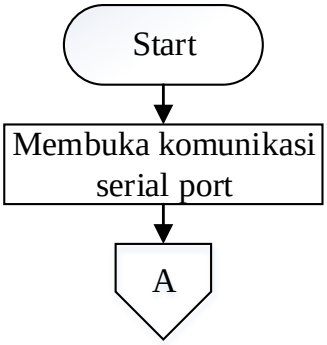


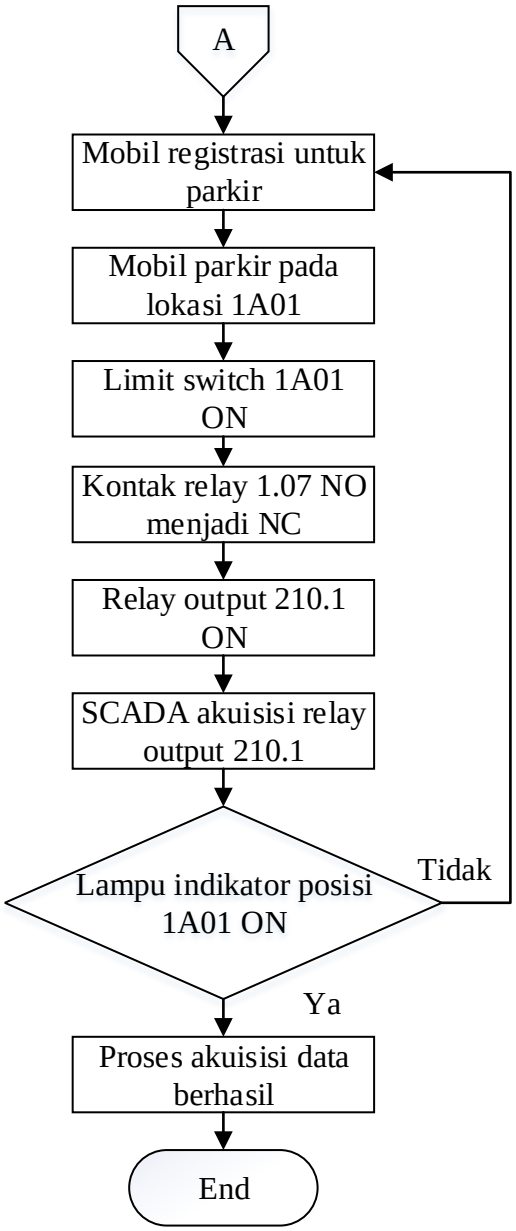
Gambar 3.79 Indikator lokasi parkir lantai satu

Berikut ini merupakan program-program yang telah dibuat pada halaman indikator lokasi parkir lantai satu yang berfungsi untuk memonitoring sistem parkir:

a. Program lampu indikator setiap lokasi parkir

Lampu indikator lokasi parkir tersebut akan bekerja berdasarkan masukan dari *limit switch* yang terdapat pada setiap lokasi parkir. *Limit switch* akan mengirimkan sinyal ke PLC pada saat mobil berada pada lokasi parkir maupun setelah mobil meniggalkan lokasi parkir tersebut. Sinyal dari *limit switch* ke PLC tersebut yang kemudian diakuisisi oleh SCADA untuk mengaktifkan atau menonaktifkan indikator pada lokasi tersebut. Gambar 3.80 merupakan diagram alir proses akuisisi data yang dilakukan oleh SCADA pada lokasi 1A01.





Gambar 5.80 Diagram alir akuisisi data untuk indikator lokasi parkir 1A01

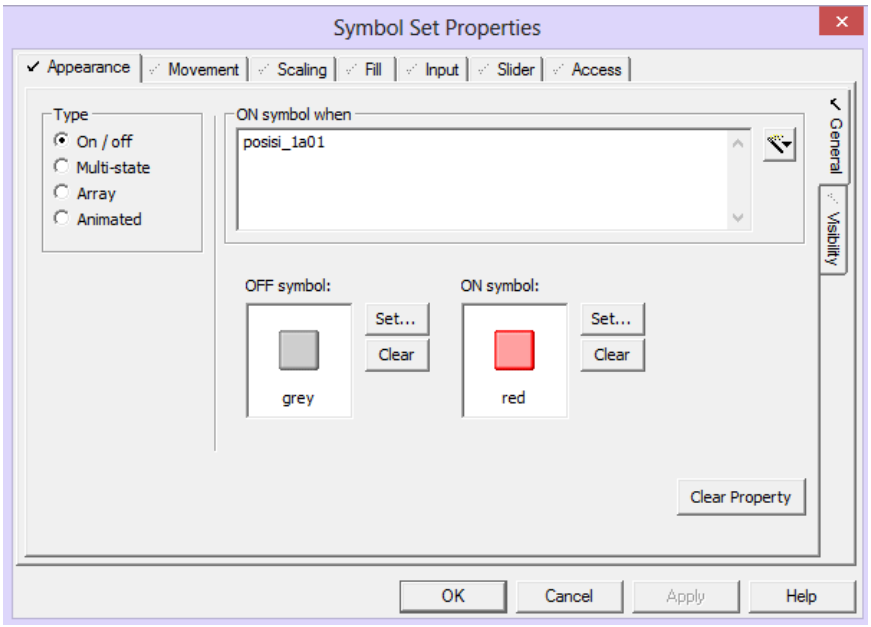
Berikut ini adalah urutan langkah kerja proses akuisisi data yang dilakukan oleh SCADA dari PLC pada lokasi parkir 1A01:

1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Mobil melakukan registrasi sebelum masuk lokasi parkir (Operator memasukkan data nomor kendaraan pada sistem SCADA). Data disimpan di PLC dan kemudian PLC memberikan informasi lokasi parkir yang masih

kosong (1A01). Informasi lokasi parkir 1A01 tersebut kemudian ditampilkan oleh SCADA.

3. Setelah proses registrasi selesai, mobil parkir lokasi parkir 1A01 sesuai informasi yang ditampilkan SCADA.
4. Selanjutnya *limit switch* pada lokasi parkir 1A01 aktif, kemudian *limit switch* mengirim sinyal tersebut pada PLC.
5. PLC menerima sinyal dari *limit switch* tersebut sehingga kontak relay 1.07 pada program PLC yang semula NO menjadi NC.
6. Kontak relay tersebut digunakan untuk mengaktifkan *relay output* 210.1, sehingga *relay output* 210.1 tersebut menyala karena relay 1.07 aktif.
7. SCADA melakukan akuisisi data pada *relay output* 210.1 tersebut.
8. *Relay output* 210.1 tersebut kemudian digunakan untuk menghidupkan lampu indikator lokasi parkir posisi 1A01.
9. Setelah lampu indikator posisi 1A01 menyala, maka proses akuisisi data berhasil.

Gambar 3.81 merupakan program yang digunakan untuk membuat lampu indikator pada lokasi parkir 1A01.



Gambar 3.81 Program lampu indikator lokasi parkir posisi 1A01

Pada Gambar 3.81 merupakan program lampu indikator lokasi parkir posisi 1A01, fungsi perintah yang digunakan untuk menjalankan indikator tersebut ialah dengan memasukkan *variable tags name* "*posisi_1a01*". Indikator tersebut menggunakan simbol lampu berwarna silver jika *limit switch* dalam keadaan non aktif sedangkan simbol lampu berwarna merah jika *limit switch* dalam keadaan aktif. Selanjutnya untuk membuat lampu indikator pada setiap lokasi parkir dari lantai satu sampai lantai lima dengan menggunakan cara yang sama seperti langkah diatas, tetapi dengan memasukkan fungsi perintah *variable tags name* yang berbeda. Fungsi perintah *variable tags name* yang digunakan seperti ditunjukkan pada Tabel 3.5.

Tabel 3.5 Fungsi perintah lampu indikator setiap lokasi parkir

No	Lampu Indikator	Fungsi Perintah (*Variable Tags Name*)
1	1A01	posisi_1a01
2	1A02	posisi_1a02
3	1A03	posisi_1a03
4	1A04	posisi_1a04
5	1B01	posisi_1b01
6	1C01	posisi_1c01
7	1D01	posisi_1d01
8	2A01	posisi_2a01
9	2A02	posisi_2a02
10	2B01	posisi_2b01
11	2C01	posisi_2c01
12	2D01	posisi_2d01
13	3A01	posisi_3a01
14	3B01	posisi_3b01
15	3C01	posisi_3c01
16	4A01	posisi_4a01

Tabel 3.5 fungsi perintah lampu indikator setiap lokasi parkir (lanjutan)

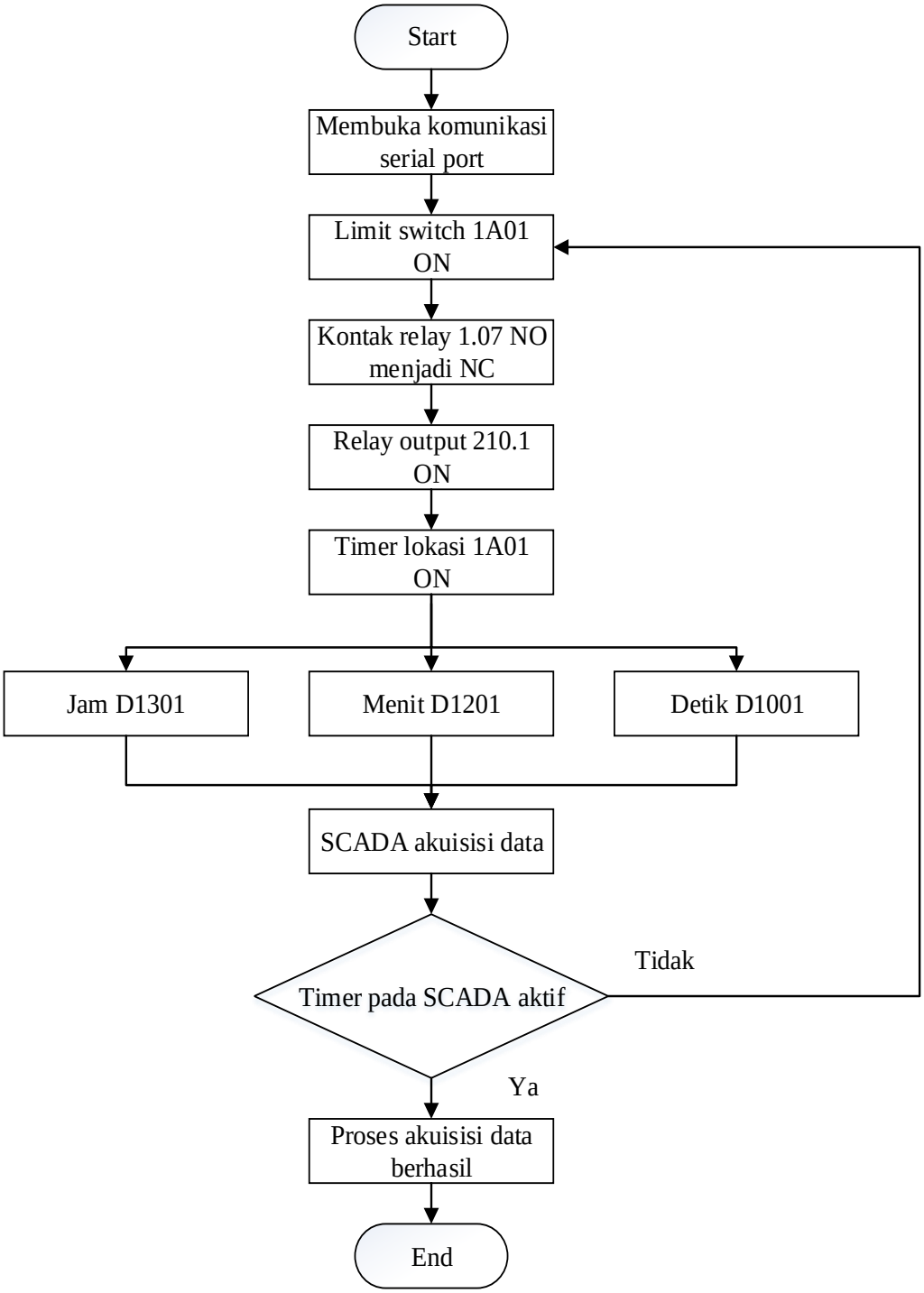
No	Lampu Indikator	Fungsi Perintah (*Variable Tags Name*)
17	4B01	posisi_4b01
18	4C01	posisi_4c01
19	5A01	posisi_5a01
20	5B01	posisi_5b01

b. Program *timer* indikator setiap lokasi parkir

Timer indikator lokasi parkir tersebut akan bekerja berdasarkan masukan dari *limit switch* yang terdapat pada setiap lokasi parkir. Jika *limit switch* pada suatu lokasi parkir aktif maka *timer* pada lokasi parkir tersebut juga aktif begitu juga jika *limit switch* non aktif maka *timer*-nya juga non aktif. *Timer* pada sistem parkir ini dibuat dengan menggunakan program *ladder diagram* pada PLC, dimana *timer* pada setiap lokasi parkir tersebut akan disimpan pada memori DM PLC. *Timer* yang tersimpan pada memori DM PLC tersebut kemudian diakuisisi oleh SCADA. Gambar 3.82 merupakan skema pengalamatan memori DM PLC yang digunakan untuk menyimpan *timer* lokasi parkir 1A01.

Gambar 3.82 Skema penyimpanan *timer* pada lokasi parkir 1A01

Gambar 3.83 merupakan diagram alir proses akuisisi data yang dilakukan oleh SCADA untuk menampilkan *timer* pada lokasi 1A01.

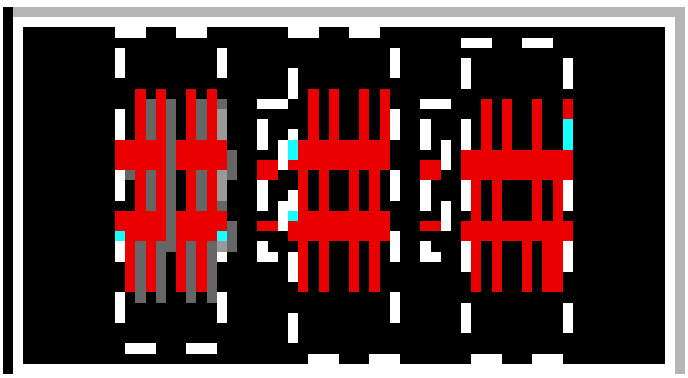


Gambar 3.83 Diagram alir akuisisi data *timer* lokasi parkir posisi 1A01

Berikut ini adalah urutan langkah kerja proses akuisisi data yang dilakukan oleh SCADA dari PLC pada lokasi parkir 1A01:

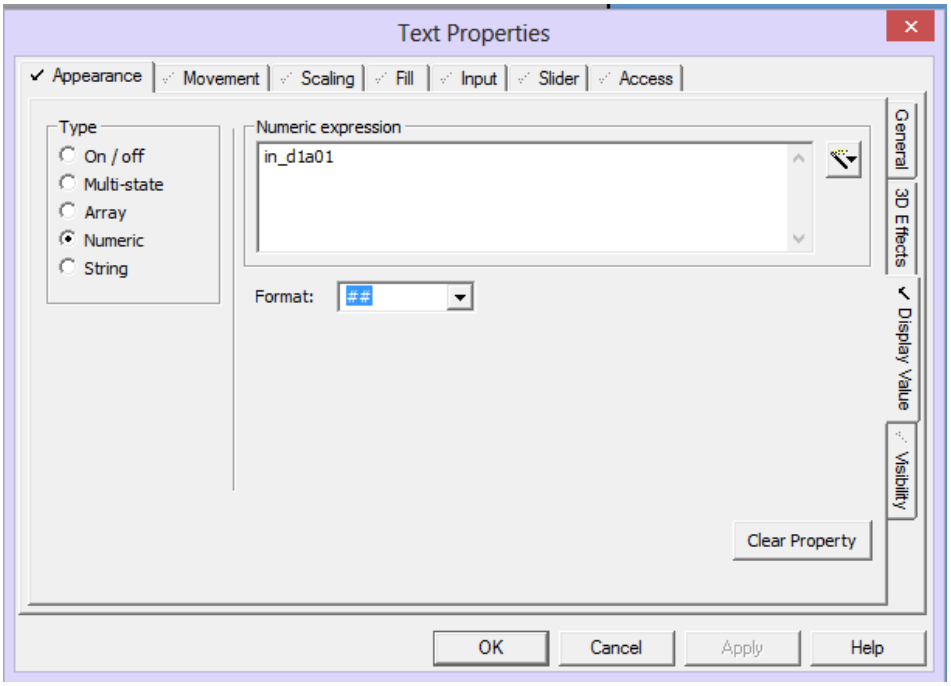
1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. *Limit switch* pada lokasi parkir 1A01 aktif, kemudian *limit switch* mengirim sinyal tersebut pada PLC.
3. PLC menerima sinyal dari *limit switch* tersebut sehingga kontak relay 1.07 pada program PLC yang semula NO menjadi NC.
4. Kontak relay tersebut digunakan untuk mengaktifkan *relay output* 210.1, sehingga *relay output* 210.1 tersebut.
5. *Relay output* 210.1 tersebut kemudian digunakan untuk mengaktifkan *timer* lokasi 1A01.
6. *Timer* tersebut disimpan pada memori DM PLC berdasarkan jenis urutan waktu seperti yang ditunjukkan Gambar 3.61 diatas. Untuk jam disimpan pada memori D1301, menit disimpan pada D1201, dan detik disimpan pada memori D1001.
7. SCADA melakukan akuisisi data pada setiap memori DM tersebut.
8. Selanjutnya SCADA menampilkan *timer* tersebut pada *timer* indikator posisi 1A01.
9. Setelah *timer* pada posisi 1A01 menyala, maka proses akuisisi data berhasil.

Gambar 3.84 merupakan desain tampilan *timer* pada lokasi 1A01 hasil dari akuisisi data dari PLC yang dilakukan oleh SCADA.

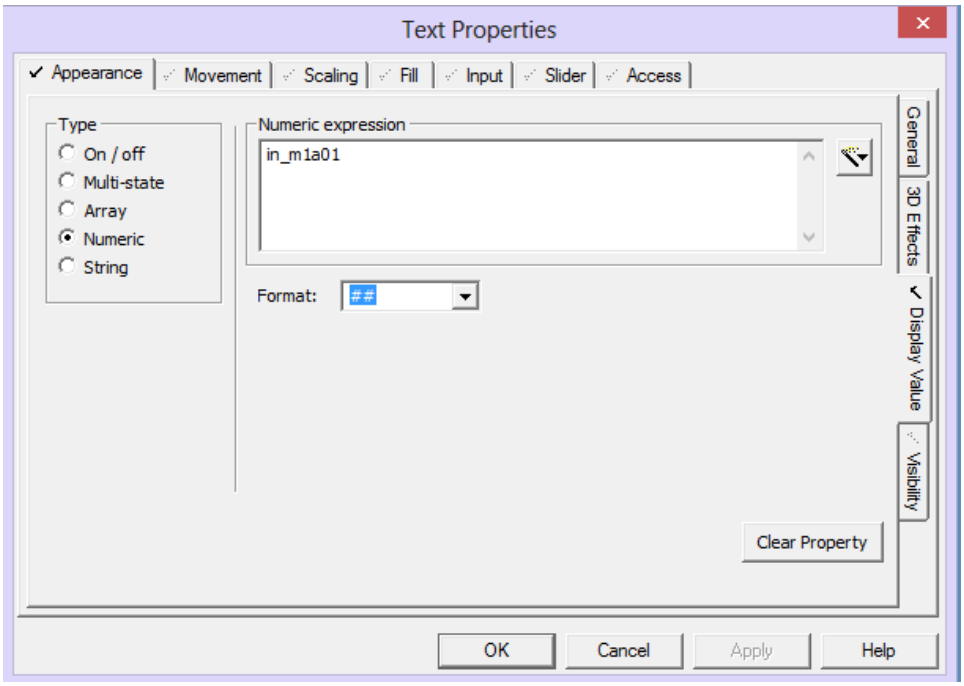


Gambar 3.84 Desain tampilan *timer* pada lokasi 1A01

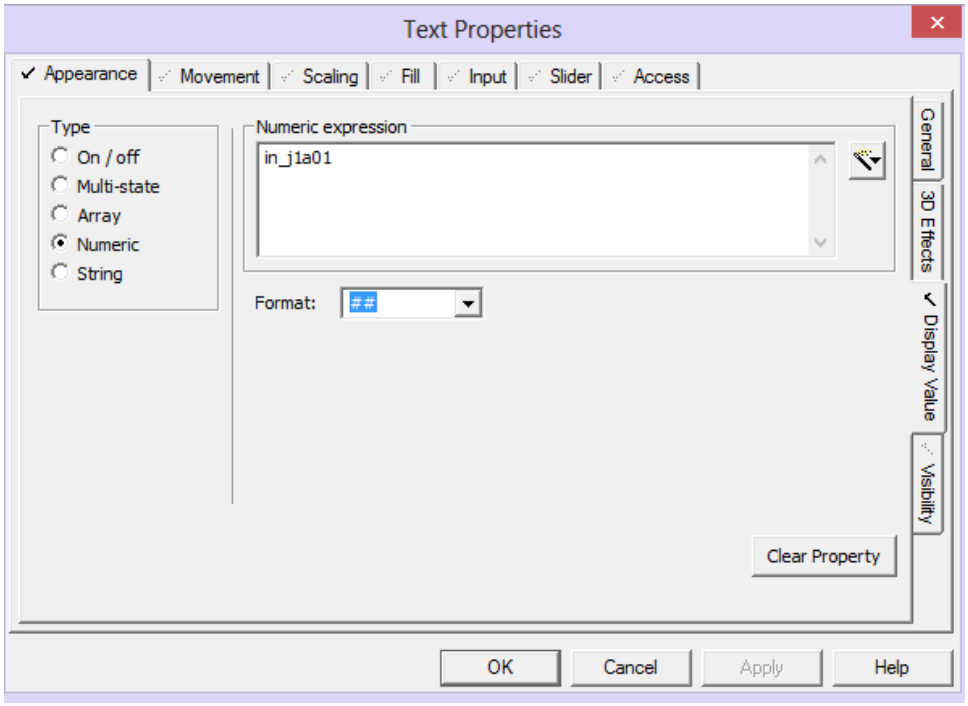
Gambar 3.85, 3.86, dan 3.87 merupakan program yang digunakan untuk menampilkan *timer* pada posisi 1A01.



Gambar 3.85 Program untuk menampilkan *timer* "detik" pada lokasi parkir 1A01



Gambar 3.86 Program untuk menampilkan *timer* "menit" pada lokasi parkir 1A01



Gambar 3.87 Program untuk menampilkan *timer* "jam" pada lokasi parkir 1A01

Langkah pertama yang dilakukan dalam membuat program tersebut ialah memilih tipe data yang akan ditampilkan, karena tipe data yang akan ditampilkan adalah data dalam bentuk angka maka memilih tipe data "*Numeric*". Langkah kedua yaitu memilih banyaknya format digit yang akan dipakai, pada gambar di atas menggunakan dua digit yang ditandai memakai dua kode "*#*". Selanjutnya untuk menampilkan data *timer* tersebut dengan cara memasukan perintah fungsi yang akan digunakan pada menu *display value*. Perintah fungsi yang digunakan untuk menjalankan program tersebut ialah dengan memasukkan *variable tags name* "*in_d1a01*" untuk menampilkan data *timer* dalam bentuk detik, *variable tags name* "*in_m1a01*" untuk menampilkan data *timer* dalam bentuk menit, dan *variable tags name* "*in_j1a01*" untuk menampilkan data *timer* dalam bentuk jam. Langkah selanjutnya untuk menampilkan data *timer* pada setiap lokasi parkir dari lantai satu sampai lantai lima dengan menggunakan cara yang sama seperti langkah diatas, tetapi dengan memasukkan fungsi perintah yang berbeda-beda seperti yang ditunjukkan pada Tabel 3.6.

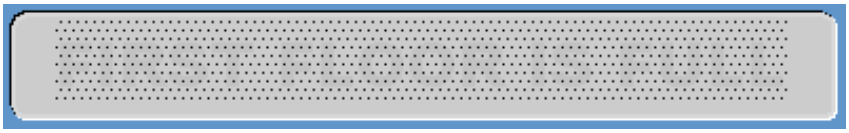
Tabel 3.6 Fungsi perintah untuk menampilkan *timer* setiap lokasi parkir

No	*Timer*	Perintah Fungsi (*Variable Tags Name*)		
		Jam	Menit	Detik
1	1A01	in_j1a01	in_m1a01	in_d1a01
2	1A02	in_j1a02	in_m1a02	in_d1a02
3	1A03	in_j1a03	in_m1a03	in_d1a03
4	1A04	in_j1a04	in_m1a04	in_d1a04
5	1B01	in_j1b01	in_m1b01	in_d1b01
6	1C01	in_j1c01	in_m1c01	in_d1c01
7	1D01	in_j1d01	in_m1d01	in_d1c01
8	2A01	in_j2a01	in_m2a01	in_d2a01
9	2A02	in_j2a02	in_m2a02	in_d2a02
10	2B01	in_j2b01	in_m2b01	in_d2b01
11	2C01	in_j2c01	in_m2c01	in_d2c01
12	2D01	in_j2d01	in_m2d01	in_d2d01
13	3A01	in_j3a01	in_m3a01	in_d3a01
14	3B01	in_j3b01	in_m3b01	in_d3b01
15	3C01	in_j3c01	in_m3c01	in_d3c01
16	4A01	in_j4a01	in_m4a01	in_d4a01
17	4B01	in_j4b01	in_m4b01	in_d4b01
18	4C01	in_j4c01	in_m4c01	in_d4c01
19	5A01	in_j5a01	in_m5a01	in_d5a01
20	5B01	in_j5b01	in_m5b01	in_d5b01

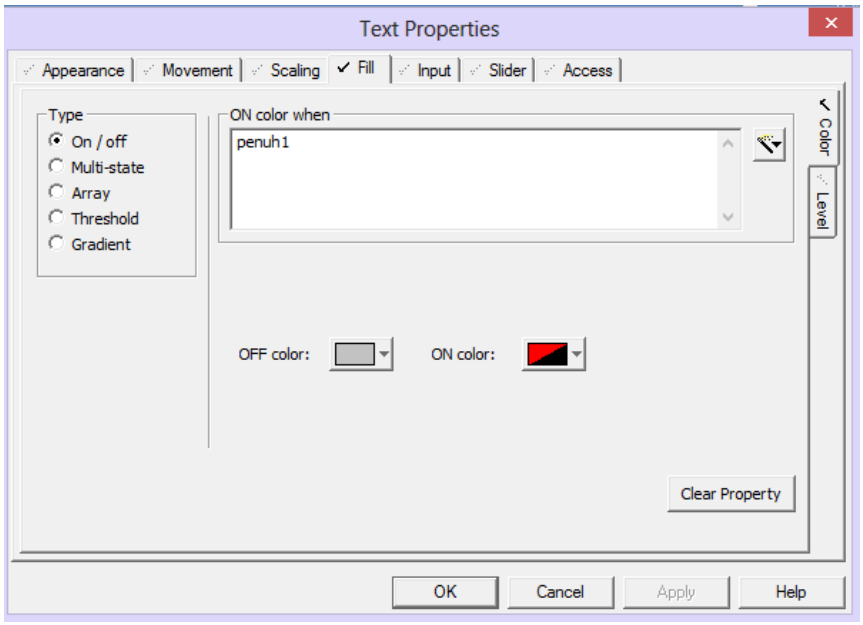
i. Program indikator jika area parkir setiap lantai penuh (*floor is full indicator*)

Floor is full indicator berfungsi untuk memudahkan operator memonitoring keadaan parkir jika lokasi parkir pada lantai tersebut telah penuh. Teks indikator tersebut akan menyala jika indikator semua lokasi parkir pada lantai tersebut sudah menyala. Gambar 3.88 merupakan tampilan *floor is full indikator* untuk lantai satu

sedangkan Gambar 3.89 adalah program *floor is full indicator* lantai satu yang telah dirancang pada perangkat lunak sistem SCADA.



Gambar 3.88 Tampilan *floor is full indicator* lantai satu



Gambar 3.89 Program *floor is full indicator* lantai satu

Gambar 3.89 merupakan program untuk membuat *floor is full indicator* lantai satu, fungsi perintah yang digunakan ialah dengan memasukkan *variable tags name* "*penuh1*" pada menu *appearance*. Kemudian menggunakan *type on* atau *off* untuk mengoperasikan indikator tersebut. Teks "*first floor is full*" pada Gambar 3.88 akan berwarna merah jika dalam keadaan aktif dan berwarna silver dalam keadaan non aktif. Untuk membuat *floor is full indicator* lantai dua sampai lantai lima yaitu dengan cara yang sama, tetapi menggunakan *variable tags* yang berbeda. Tabel 3.6 merupakan *variable tags name* yang digunakan sebagai fungsi perintah untuk membuat indikator tersebut.

Tabel 3.7 Fungsi perintah *floor is full indicator*

No	Lampu Indikator	Fungsi Perintah (*Variable Tags Name*)
1	*Second floor is full*	Penuh2
2	*Third floor is full*	Penuh3
3	*Fourth floor is full*	Penuh4
4	*Fiveth floor is full*	Penuh5

3.4.5.1.4 Perancangan Program Karcis Parkir

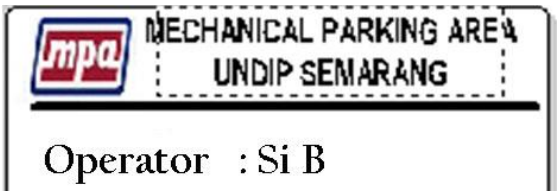
Karcis parkir pada *input section* ini berfungsi untuk memberikan informasi lokasi parkir yang masih kosong bagi pengguna parkir. Dalam penyusunan program karcis tersebut, ada tiga bagian utama yang terdapat pada halaman tersebut yaitu *operator name*, *car number*, dan *parking location*. Gambar 3.90 merupakan desain tampilan karcis parkir yang telah dirancang.

Gambar 3.90 Tampilan karcis parkir pada *input section*

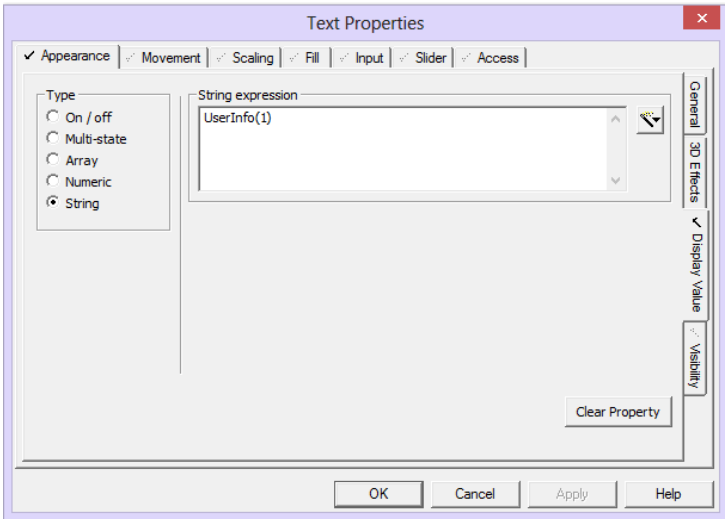
Berikut ini merupakan program-program yang telah dibuat pada halaman karcis parkir:

a. Nama operator (*operator name*)

   Gambar 3.91 merupakan desain tampilan dari *operator name* yang telah dirancang, sedangkan Gambar 3.92 merupakan program yang digunakan untuk membuat dan menjalankan *operator name* tersebut.



Gambar 3.91 Desain tampilan *operator name*

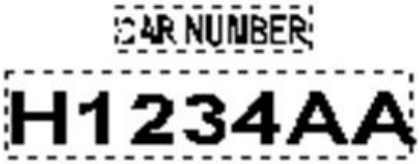


Gambar 3.92 Program *operator name*

Langkah pertama yang dilakukan dalam membuat program tersebut ialah memilih tipe data yang akan ditampilkan, karena tipe data yang akan ditampilkan adalah data dalam bentuk teks maka memilih tipe data "*String*". Selanjutnya untuk menampilkan data tersebut dengan cara memasukan perintah fungsi yang akan digunakan pada menu *display value*. Perintah fungsi yang digunakan untuk menjalankan program tersebut ialah dengan memasukkan fungsi "*userinfo1*" untuk menampilkan data nama operator tersebut. Fungsi "*userinfo1*" akan menampilkan nama pengguna yang telah melakukan *login* pada menu *user* registrasi.

b. Nomor kendaraan (*car number*)

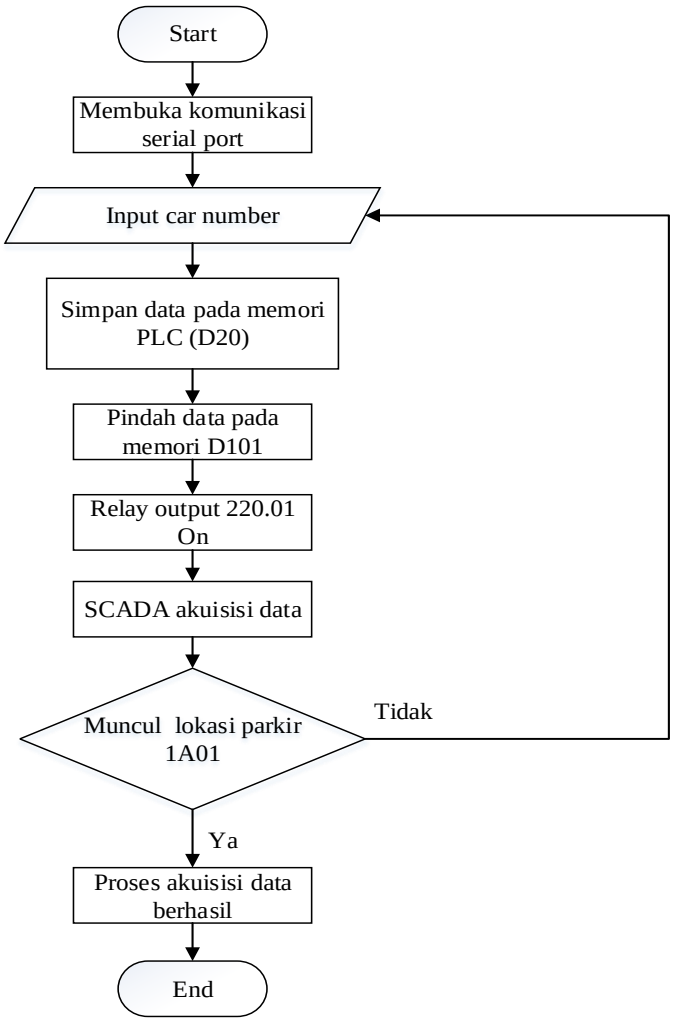
Program *car number* yang terdapat pada halaman karcis parkir ini mengambil data dari halaman HMI pada program *input number*. Gambar 3.93 merupakan tampilan desain untuk menampilkan nomor kendaraan pada halaman karcis.



Gambar 3.93 Tampilan nomor kendaraan pada karcis parkir

c. Program informasi lokasi parkir (*parking location information*)

Parking location information berfungsi untuk menampilkan lokasi parkir yang masih kosong dari program PLC yang telah dirancang. Gambar 3.94 berikut ini merupakan diagram alir porses kerja PLC dan SCADA untuk menampilkan lokasi parkir 1A01.

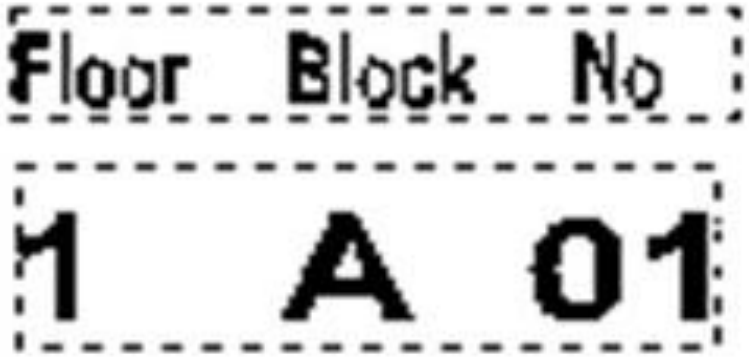


Gambar 3.94 Diagram alir proses akuisisi data untuk menampilkan lokasi 1A01

Berikut ini adalah urutan langkah kerja proses akuisisi data yang dilakukan oleh SCADA dari PLC pada lokasi parkir 1A01:

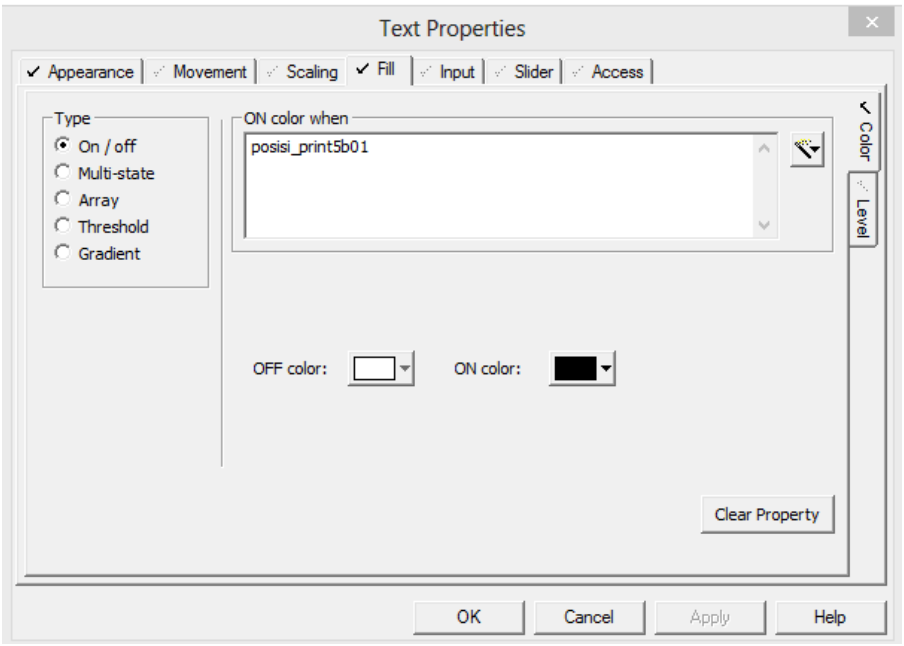
1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Operator memasukkan data nomor kendaraan pada sistem SCADA.
3. Data nomor kendaraan tersebut disimpan di memori PLC D20.
4. Selanjutnya data tersebut dipindah ke memori PLC D101 yang merupakan memori untuk menyimpan nomor kendaraan posisi 1A01.
5. Setelah memori dipindah ke D101, maka *relay output* 220.01 menyala.
6. SCADA melakukan akuisisi data pada *relay output* 220.01 tersebut.
7. *Relay output* tersebut yang digunakan untuk memampilkan lokasi parkir 1A01.
8. Setelah informasi lokasi parkir posisi 1A01 menyala, maka proses akuisisi data berhasil.

Gambar 3.95 di bawah ini merupakan desain tampilan untuk menampilkan lokasi parkir 1A01 hasil dari akuisisi data dari PLC yang dilakukan oleh SCADA.



Gambar 3.95 Desain tampilan informasi lokasi parkir 1A01

Gambar 3.96 merupakan program yang digunakan untuk menampilkan informasi lokasi parkir posisi 1A01. Fungsi perintah yang digunakan untuk menjalankan program tersebut dengan memasukkan *variable tags name* "*posisi_print1a01*". Teks "*1A01*" pada Gambar 3.95 akan berwarna hitam jika dalam keadaan aktif dan berwarna putih dalam keadaan non aktif.



Gambar 3.96 Program informasi lokasi parkir 1A01

Untuk menampilkan informasi lokasi parkir dari lantai satu sampai lantai lima dengan menggunakan cara yang sama, tetapi menggunakan *variable tags* yang berbeda. Tabel 3.7 merupakan *variable tags name* yang digunakan sebagai fungsi perintah untuk menampilkan lokasi parkir tersebut.

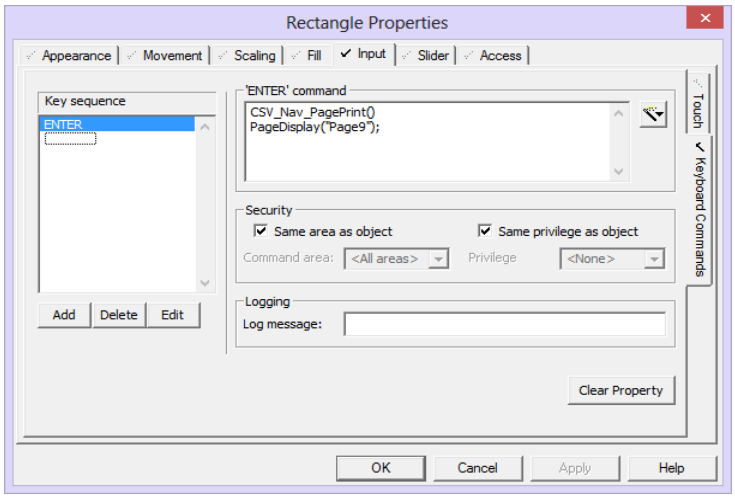
Tabel 3.8 Fungsi perintah untuk menampilkan informasi setiap lokasi parkir

No	Informasi Lokasi Parkir	Fungsi Perintah (*Variable Tags Name*)
1	1A01	posisi_print1a01
2	1A02	posisi_ print1a02
3	1A03	posisi_ print1a03
4	1A04	posisi_ print1a04
5	1B01	posisi_ print1b01
6	1C01	posisi_ print1c01
7	1D01	posisi_ print1d01
8	2A01	posisi_ print2a01

Tabel 3.8 Fungsi perintah untuk menampilkan informasi setiap lokasi parkir (lanjutan)

No	Informasi Lokasi Parkir	Fungsi Perintah (*Variable Tags Name*)
9	2A02	posisi_ print2a02
10	2B01	posisi_ print2b01
11	2C01	posisi_ print2c01
12	2D01	posisi_ print2d01
13	3A01	posisi_ print3a01
14	3B01	posisi_ print3b01
15	3C01	posisi_ print3c01
16	4A01	posisi_ print4a01
17	4B01	posisi_ print4b01
18	4C01	posisi_ print4c01
19	5A01	posisi_ print5a01
20	5B01	posisi_ print5b01

Agar halaman karcis parkir tersebut dapat di-*print* maka halaman kita masukkan fungsi perintah untuk menjalankan perintah *print* tersebut. Gambar 3.97 merupakan program yang digunakan untuk memberikan perintah *print*.



Gambar 3.97 Program perintah *print* halaman karcis parkir

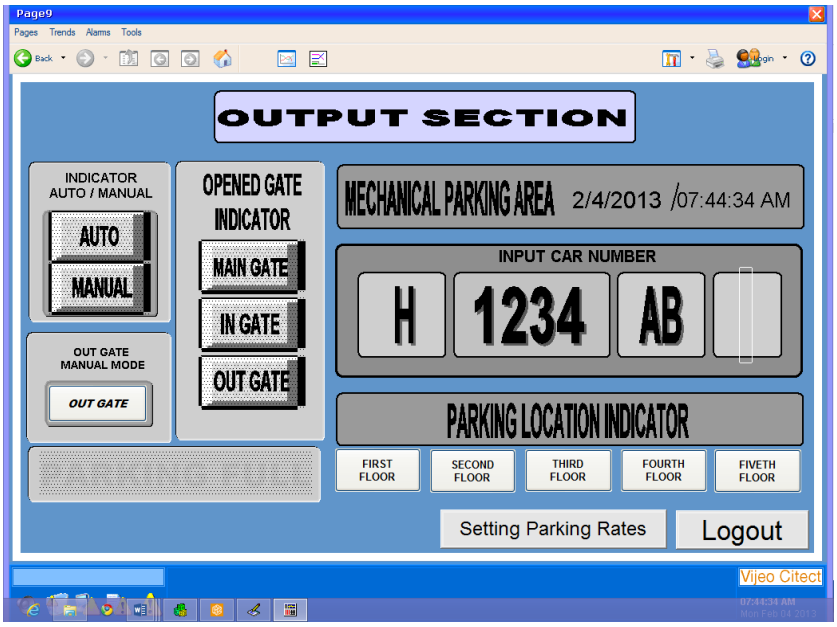
Pada Gambar 3.97 merupakan program untuk memerintahkan SCADA mencetak halaman karcis parkir. Langkah pertama yaitu mengisisi *key sequence* "*enter*" yang befungsi untuk menjalankan program tersebut jika tombol "*enter*" pada *keyboard* ditekan . Untuk menjalankan program tersebut fungsi perintah yang digunakan ialah "*CSV_Nav_PagePrint()*" sebagai perintah SCADA melakukan proses *print* dan fungsi perintah "*PageDisplay("Page9");*" sebagai perintah untuk menampilkan halaman sembilan yang merupakan halaman HMI.

3.4.5.2 Program Sistem SCADA Pada *Output Section*

Pada sistem SCADA bagian *output section* sebagian besar program yang dibuat sama dengan program pada perangkat lunak sistem SCADA bagian *input section*. Jadi pada penjelasan ini hanya dibahas program-program yang tidak sama dengan bagian *input section*. Berikut ini merupakan bagian-bagian program yang tidak sama tersebut.

3.4.5.2.1 Perancangan Program *Human Machine Interface* (HMI)

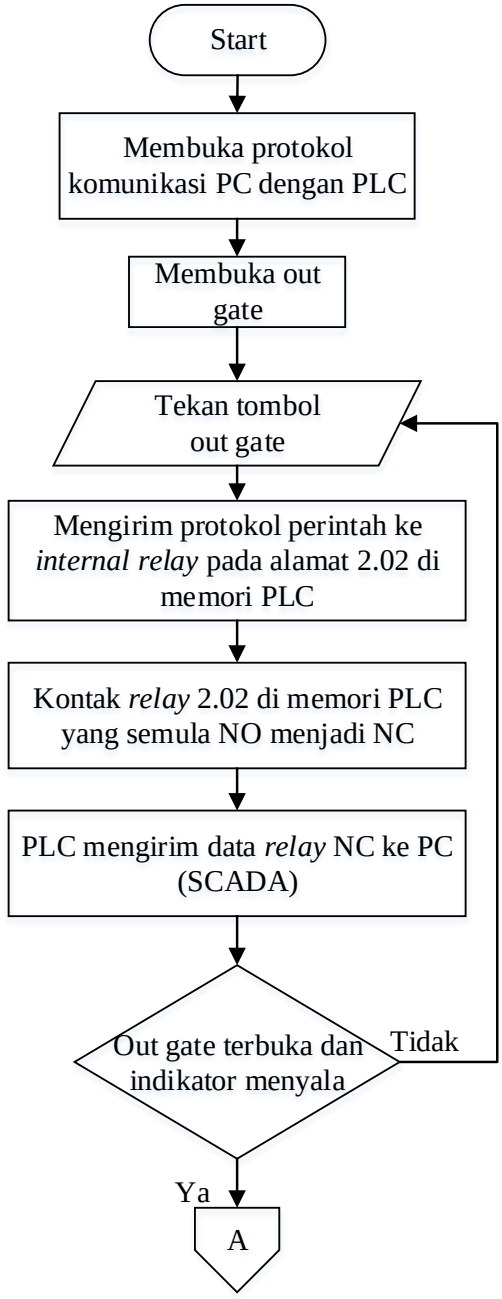
Gambar 3.98 merupakan desain tampilan *graphic* dari program HMI bagian *output section* yang dirancang. Hampir semua bagian sama pada tampilan HMI *input section* kecuali kendali *out gate* dan *input car number*.

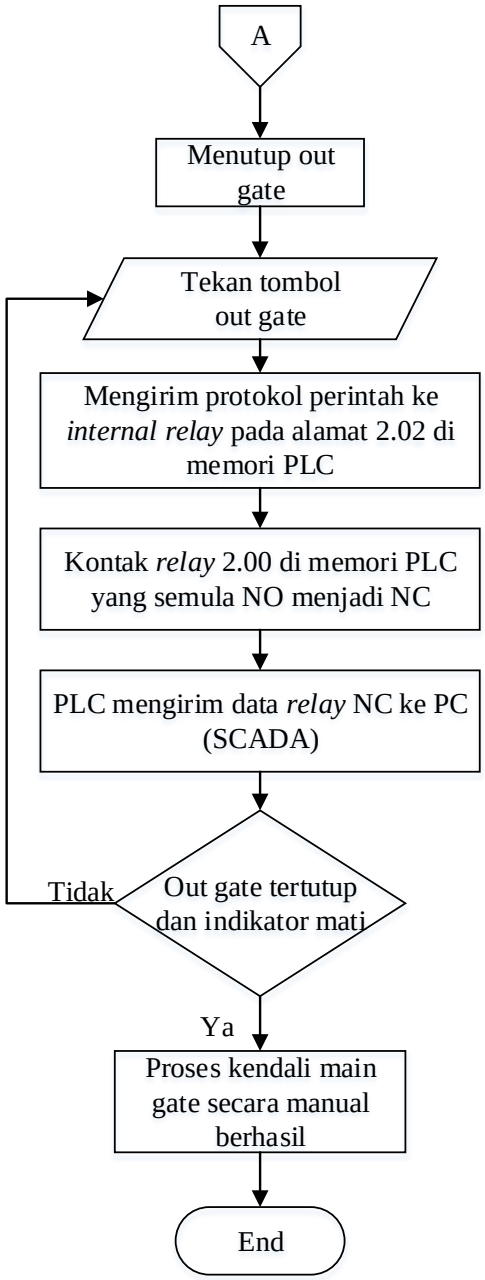


Gambar 3.98 Tampilan HMI *output section*

a. Program kendali *gate* secara manual (*push button out gate*)

Untuk memerintahkan PLC melakukan proses kendali *gate* parkir secara manual dengan menggunakan push button *out gate*, maka PC akan mengirimkan protokol perintah ke PLC dan selanjutnya PLC mengirimkan datanya tersebut ke PC. Gambar 3.99 adalah diagram alir dari program proses kendali *out gate* secara manual.



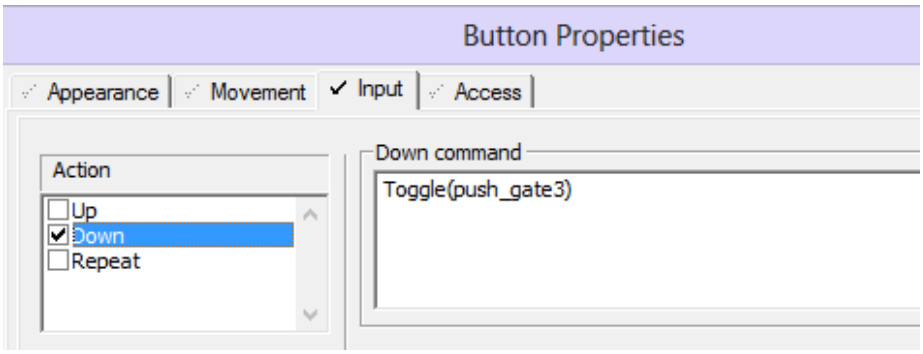


Gambar 3.99 Diagram alir program proses kendali *gate* secara manual

Berikut ini adalah urutan langkah kerja proses kendali *out gate* parkir secara manual:

1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Setelah PC dan PLC terkoneksi, selanjutnya untuk membuka *out gate* secara manual dengan cara menekan *push button out gate*.

3. Kemudian PC (SCADA) akan mengirim protokol perintah ke memori PLC, dimana alamat yang dituju adalah kontak *relay* 2.02 pada *ladder diagram* yang telah dibuat pada memori PLC.
4. PLC akan menjalankan perintah tersebut sehingga kontak *relay* 2.02 pada *ladder diagram* yang semula NO berubah menjadi NC.
5. Perubahan data kontak *relay* 2.02 tersebut selanjutnya dikirim PLC ke PC (SCADA).
6. Jika data kontak *relay* 2.02 tersebut sampai ke SCADA dan sesuai dengan alamat yang dituliskan pada indikator, maka indikator *out gate* akan menyala dan *out gate* akan terbuka.
7. Untuk selanjutnya menutup *out gate* secara manual dengan cara menekan kembali *push button out gate*.
8. PC (SCADA) akan mengirim protokol perintah ke memori PLC, dimana alamat yang dituju adalah kontak *relay* 2.02 pada *ladder diagram* yang telah dibuat pada memori PLC.
9. PLC akan menjalankan perintah tersebut sehingga kontak *relay* 2.02 pada *ladder diagram* yang semula NO berubah menjadi NC.
10. Perubahan data kontak *relay* 2.02 tersebut selanjutnya dikirim PLC ke PC (SCADA).
11. Jika data kontak *relay* 2.02 tersebut sampai ke SCADA dan sesuai dengan alamat yang dituliskan pada indikator, maka indikator *out gate* akan mati dan *out gate* tertutup.

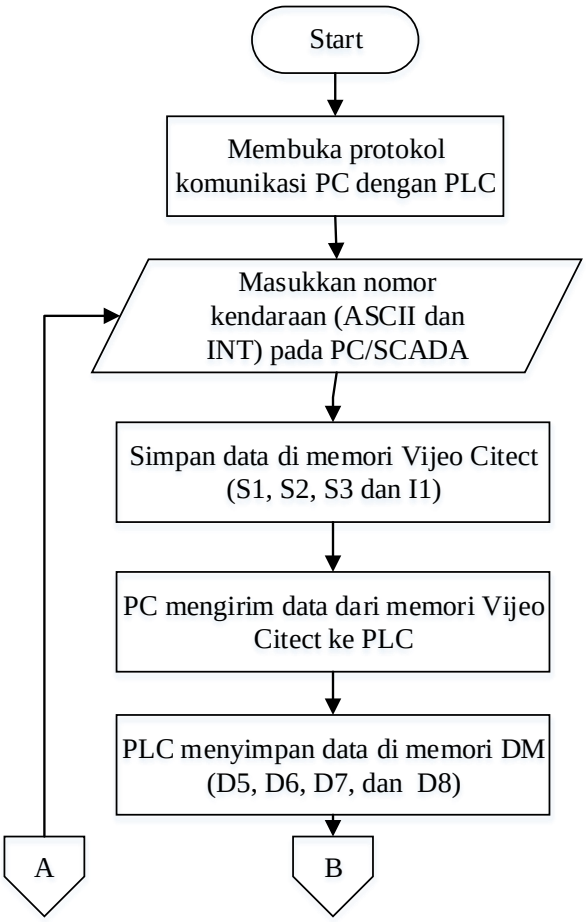


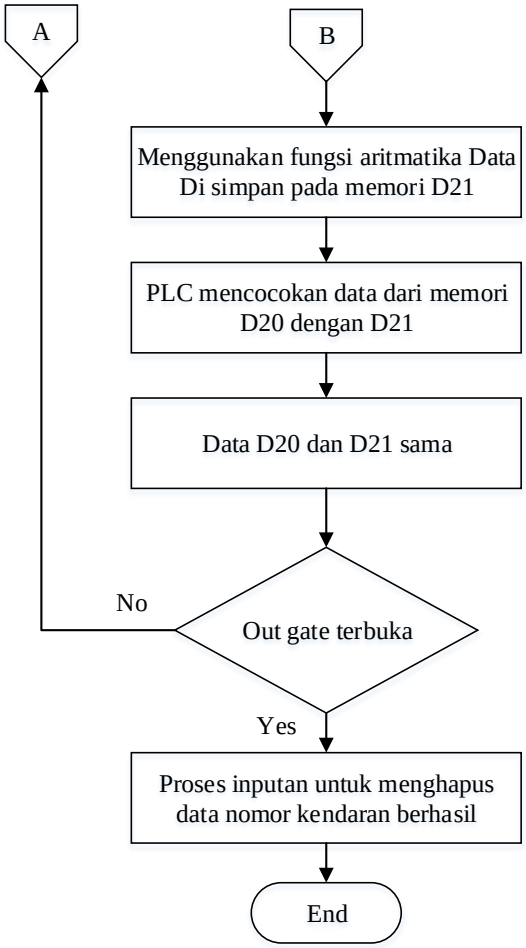
Gambar 3.100 Program *push button out gate*

Gambar 3.100 merupakan program yang digunakan untuk membuat *push button out gate*, fungsi perintah yang digunakan untuk menjalankan *push button* tersebut ialah dengan memasukkan *variable tags* "*Toggle(push_gate3)*".

b. Program untuk memasukkan nomor kendaraan (*input car number output section*)

Pada prinsipnya program *input car number output section* sama dengan program *input car number pada input section*, namun ada perbedaan dalam pengalamatan memori pada PLC. Jika pada *input section* nomor kendaraan disimpan pada memori D0, D1, D2, dan D3 maka untuk *output section* nomor kendaraan disimpan pada memori D5, D6, D7, dan D8. Untuk menjalankan program tersebut fungsi perintah yang dipakai baik pada input maupun *output section* menggunakan *variable tags name* yang sama. Gambar 3.101 di bawah ini merupakan diagram alir proses kerja program *input car number pada output section*.





Gambar 3.101 Diagram alir proses kerja *input car number* pada *output section*

Berikut ini adalah urutan langkah kerja proses inputan nomor kendaraan pada *output section*:

1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Setelah PC dan PLC terkoneksi, selanjutnya nomor kendaraan dimasukkan sesuai dengan kategorinya (ASCII dan INT).
3. Kemudian data nomor kendaraan yang telah masuk disimpan pada memori SCADA untuk karakter ASCII disimpan pada memori S1, S2, dan S3 (STRING). Sedangkan untuk karakter INT disimpan di memori I1 (INTEGER).
4. Data nomor kendaraan yang telah disimpan pada memori SCADA dikirim ke PLC.

5. Selanjutnya PLC menyimpan data nomor kendaraan pada memori DM (D5, D6, D7, dan D8).
6. Untuk selanjutnya data pada memori DM (D5, D6, D7, dan D8) kemudian dijumlahkan menggunakan program PLC dan hasilnya di simpan pada memori D21.
7. Kemudian PLC mencocokan data yang tersimpan pada memori D20 dengan D21, jika data pada kedua memori tersebut sama maka data pada kedua memori tersebut akan terhapus atau bernilai nol.
8. Setelah proses pencocokan data berhasil maka *out gate* parkir akan terbuka.
9. Jika *out gate* telah terbuka maka proses inputan untuk menghapus data nomor kendaraan pada memori SCADA dan PLC berhasil.

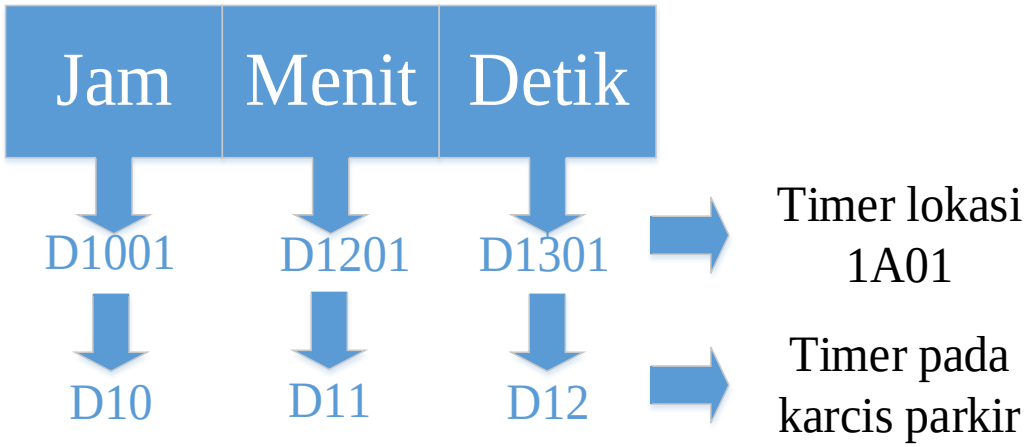
3.4.5.2.2 Peracangan Program Karcis Parkir Keluar

Karcis parkir pada *output section* ini berfungsi untuk memberikan informasi durasi parkir dan biaya parkir. Dalam penyusunan program karcis tersebut, ada tiga bagian utama yang terdapat pada halaman tersebut yaitu *operator name*, durasi parkir, dan biaya parkir. Gambar 3.102 merupakan desain tampilan karcis parkir yang telah dirancang. Karena program untuk *operator name* sama dengan program pada karcis *input section* maka yang akan dijelaskan adalah proses akuisisi data untuk waktu lama parkir dan biaya parkir.

Gambar 3.102 Tampilan karcis parkir pada *input section*

a. Program durasi parkir (*parking duration*)

Program durasi parkir berfungsi untuk menampilkan durasi parkir setiap mobil yang telah keluar dari sistem parkir. Sistem kerja program durasi parkir ini adalah setiap mobil yang telah selesai parkir maka *timer* pada lokasi parkir mobil tersebut dipindah pada memori D10, D11, dan D12. Gambar 3.103 merupakan skema pengalamatan memori PLC untuk menampilkan durasi parkir pada lokasi 1A01.

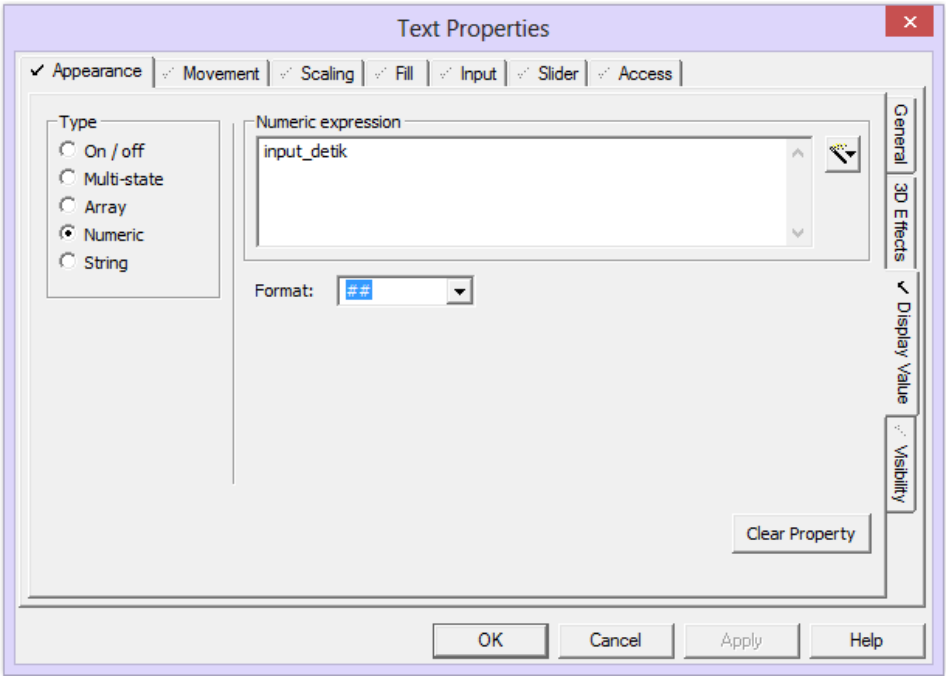


Gambar 3.103 Skema pengalamatan memori PLC untuk durasi parkir lokasi 1A01

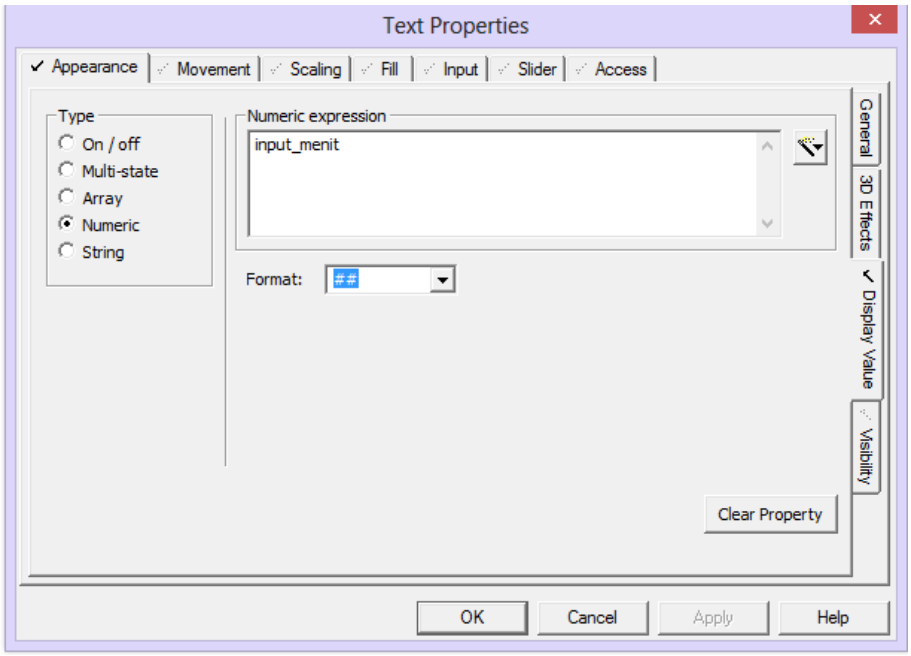
Gambar 3.104 di bawah ini merupakan desain tampilan durasi parkir pada karcis parkir hasil dari akuisisi data dari PLC yang dilakukan oleh SCADA.

Gambar 3.104 Desain tampilan durasi parkir pada karcis parkir

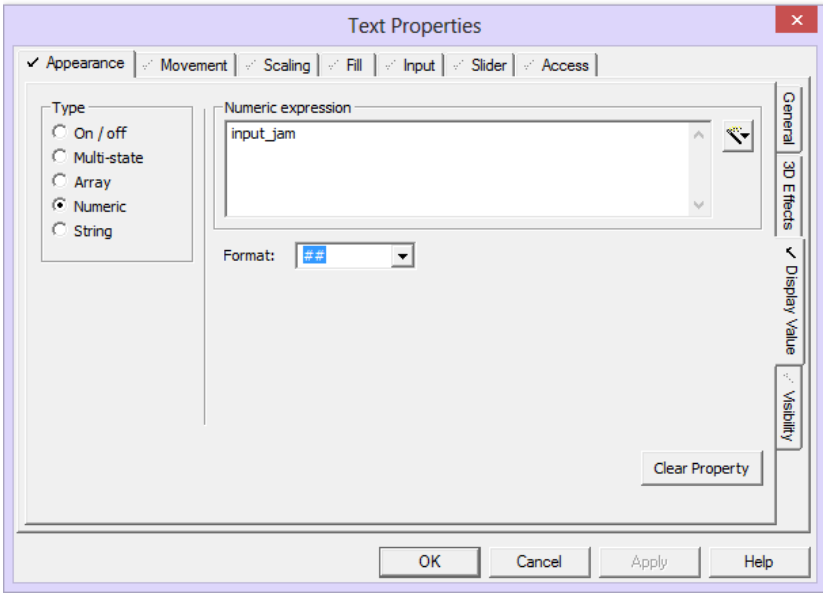
Gambar 3.105, 3.106, dan 3.107 merupakan program yang digunakan untuk menampilkan durasi parkir.



Gambar 3.105 Program untuk menampilkan durasi parkir “detik”



Gambar 3.106 Program untuk menampilkan durasi parkir “menit”

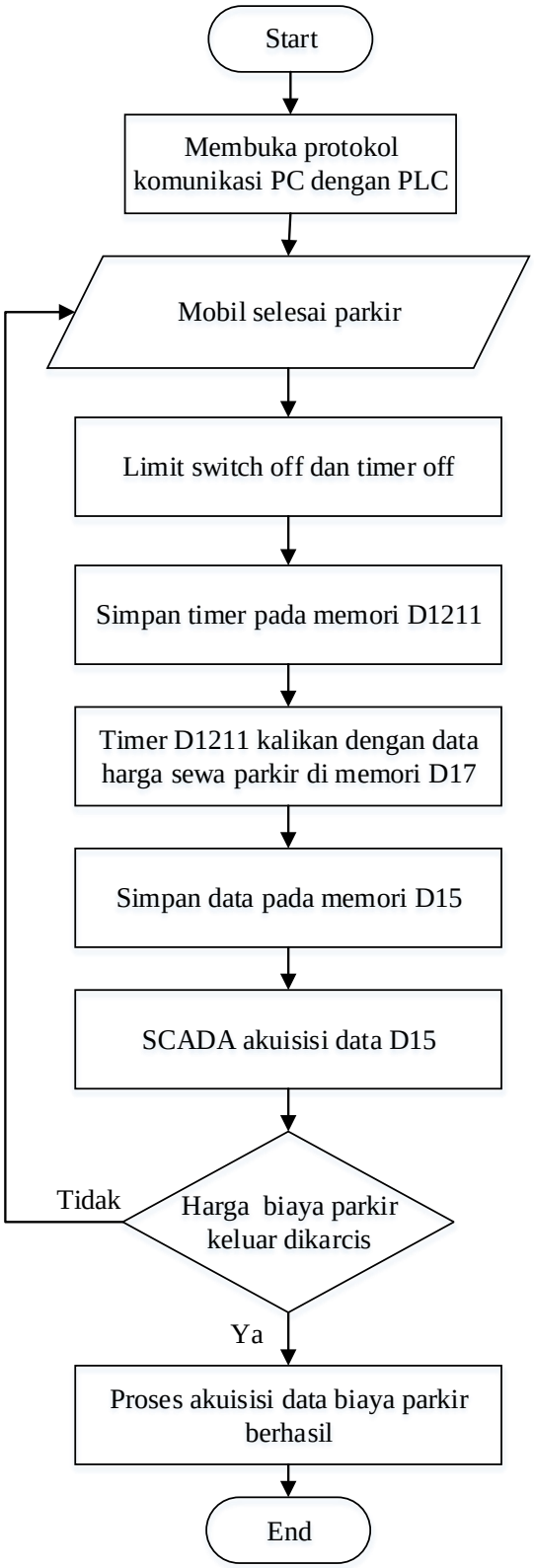


Gambar 3.107 Program untuk menampilkan durasi parkir "jam"

Langkah pertama yang dilakukan dalam membuat program tersebut ialah memilih tipe data yang akan ditampilkan, karena tipe data yang akan ditampilkan adalah data dalam bentuk angka maka memilih tipe data "*Numeric*". Langkah kedua yaitu memilih banyaknya format digit yang akan dipakai, pada Gambar 3.107 menggunakan dua digit yang ditandai memakai dua kode "*#*". Selanjutnya untuk menampilkan data *timer* tersebut dengan cara memasukan perintah fungsi yang akan digunakan pada menu *display value*. Perintah fungsi yang digunakan untuk menjalankan program tersebut ialah dengan memasukkan *variable tags name* "*input_detik*" untuk menampilkan data durasi parkir dalam bentuk detik, *variable tags name* "*input_menit*" untuk menampilkan data durasi parkir dalam bentuk menit, dan *variable tags name* "*input_jam*" untuk menampilkan data durasi parkir dalam bentuk jam.

b. Program biaya parkir (*parking rates*)

Program *parking rates* berfungsi untuk menampilkan biaya yang dikenakan pada pengguna parkir setelah selesai parkir. Gambar 3.108 merupakan diagram alir proses akuisisi data yang dilakukan oleh SCADA untuk menampilkan biaya pada karcis parkir.

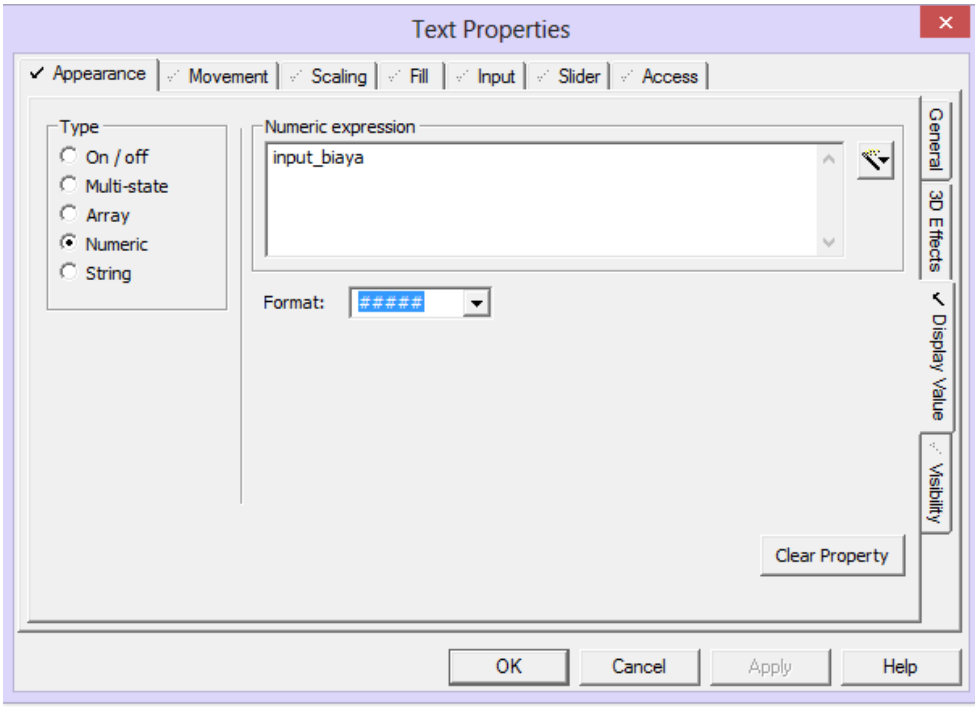


Gambar 3.108 Diagram alir proses akuisisi data untuk menampilkan biaya tarif parkir

Berikut ini adalah urutan langkah kerja proses akuisisi data yang dilakukan oleh SCADA dari PLC pada lokasi parkir 1A01:

1. Mula-mula PC membuka komunikasi serial dengan PLC terlebih dahulu.
2. Mobil selesai parkir dan meninggalkan lokasi parkir.
3. *Limit switch* dan timer pada lokasi parkir tersebut *off*.
4. *Timer* pada lokasi parkir tersebut disimpan pada memori PLC D1211.
5. *Timer* yang disimpan pada memori D1211 dikalikan dengan harga sewa parkir pada memori D17.
6. Hasil perkalian data tersebut kemudian disimpan pada memori PLC D15.
7. Data biaya parkir pada memori D15 diakuisisi data oleh SCADA yang kemudian ditampilkan pada halaman karcis parkir.
9. Setelah informasi biaya parkir keluar pada karcis, maka proses akuisisi data berhasil.

Gambar 3.109 merupakan program yang digunakan untuk menampilkan biaya sewa parkir.



Gambar 3.109 Program untuk menampilkan biaya sewa parkir

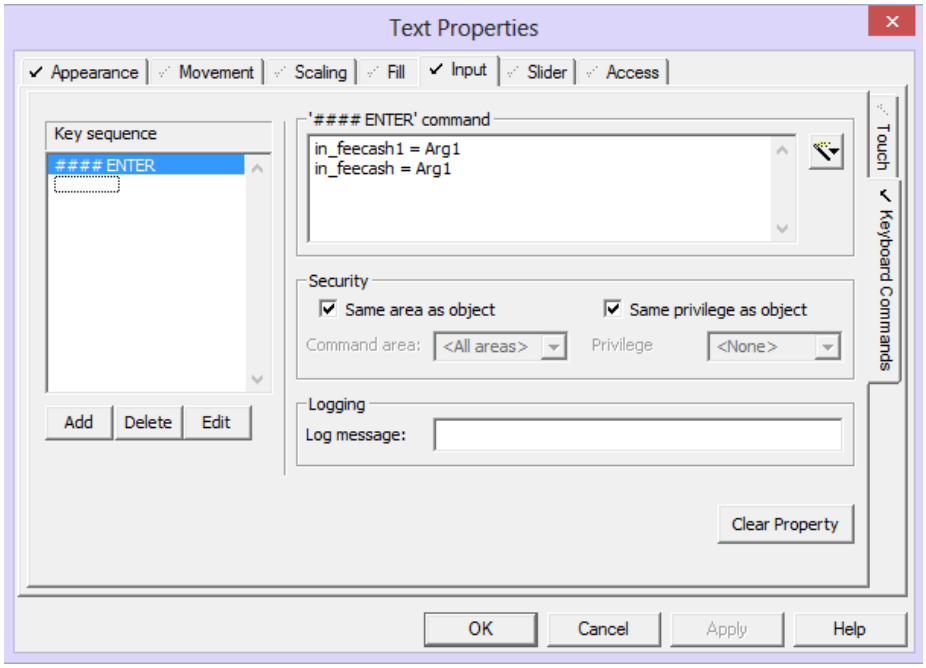
Langkah pertama yang dilakukan dalam membuat program tersebut ialah memilih tipe data yang akan ditampilkan, karena tipe data yang akan ditampilkan adalah data dalam bentuk angka maka memilih tipe data "*Numeric*". Langkah kedua yaitu memilih banyaknya format digit yang akan dipakai, untuk menampilkan data biaya parkir akan menggunakan lima digit yang ditandai memakai lima kode "*#*". Selanjutnya untuk menampilkan data biaya tersebut dengan cara memasukan perintah fungsi pada menu *display value*. Perintah fungsi yang digunakan untuk menjalankan dan menampilkan program tersebut ialah dengan memasukkan *variable tags name* "*input_biaya*".

3.4.5.2.3 Perancangan Program Pengaturan Tarif Dasar Parkir (*Setting Parking Rates*)

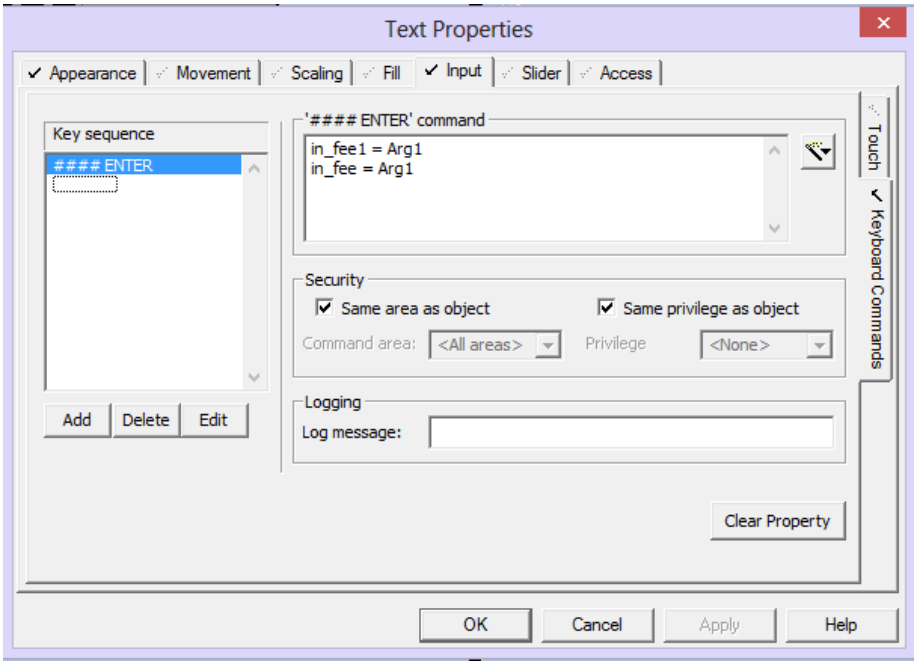
Program *setting parking rates* digunakan bagi operator untuk memasukkan tarif dasar biaya sewa parkir. Gambar 3.110 merupakan desain tampilan *setting parking rates* yang telah dibuat.

Gambar 3.110 Tampilan *input parking rates*

Gambar 3.111 dan 3.112 merupakan program yang digunakan untuk memasukkan tarif biaya sewa parkir.



Gambar 3.111 Program *base parking rates*



Gambar 3.112 Program *parking rates after 1 hour*

Untuk membuat program seperti pada Gambar 3.111 dan 3.112 langkah pertama yang dilakukan yaitu memasukkan banyaknya digit yang akan dipakai pada menu *key sequence*. Karena pada program ini jumlah digit yang akan dipakai adalah empat digit, maka menggunakan empat kode "*#*" kemudian diakhiri dengan *key sequence* "*enter*". Sedangkan untuk menjalankan program tersebut menggunakan perintah fungsi *variable tags name* "*in_fee1* dan *in_feecash1*" sebagai perintah memori SCADA menyimpan data pada memori I6 dan I8 sedangkan *variable tags name* "*in_fee* dan *in_feecash*" sebagai perintah memori PLC melakukan penerimaan dan penyimpanan data pada memori D17 dan D19.