PERANCANGAN SIMULATOR PLC MENGGUNAKAN PROGRAM DELPHI

Jusuf Bintoro[1]; M. Satria Pinandita[2]; Edi Santoso[3]; Komang Oka I.[4]

[1]Jurusan Teknik Elektro, Jln. Rawamangun Muka, Jakarta 13220
[2, 3, 4] Jurusan Sistem Komputer, Fakultas Ilmu Komputer, Universitas Bina Nusantara,
Jln. K.H. Syahdan No. 9, Kemanggisan, Palmerah, Jakarta Barat 11480
unj@unj.ac.id

ABSTRACT

The purpose of this research is to design a PLC Simulator, that can be used to make it more easy for beginner users to understand programming and how it works. PLC simulator programming through IDE can be conducted, to design a PLC program using Ladder Diagram and Statement List. PLC basic instructions, flag, Timer and Counter can be used for programming using Ladder Diagram. Design of STL instructions is made according to digital logic. Available inputs are normally open input and normally closed input, while available outputs are lamp, buzzer and solenoid valve. The result of PLC simulation can be seen at the program interface which represents active contact and lamp output, the sounding of buzzer and the moving solenoid valve.

Keywords: *PLC Simulator, PLC (Programmable Logic Controller)*

ABSTRAK

Penelitian ini bertujuan untuk merancang sebuah Simulator PLC, yang dapat digunakan untuk memudahkan pengguna pemula memahami pemrograman dan cara kerja PLC. Pemrograman simulator PLC melalui IDE (Integrated Development Environment) dapat dilakukan untuk membuat program PLC mengunakan diagram Ladder (Ladder diagram), dan Statement list (STL). Perintah-perintah dasar PLC, flag Timer, dan Counter dapat digunakan untuk pemrograman menggunakan Diagram Ladder. Perintah-perintah STL dibuat sesuai dengan logika digital. Masukan yang disediakan berupa input normally open dan input normally closed, sedangkan keluaran berupa lampu, buzzer, dan solenoid valve. Hasil simulasi PLC dapat dilihat pada tampilan program yaitu kontak yang aktif dan keluaran lampu yang menyala, buzzer berbunyi, dan selenoid valve yang bergerak maju atau mundur.

Kata kunci: *Simulator PLC, PLC (Programmable Logic Controller)*

PENDAHULUAN

Sejak ditemukannya PLC pada tahun 1968, daya tahan ujinya sudah terbukti pada lingkungan perangkat keras dan telah dibuat untuk menangani banyak *input* dan *output*, dimana banyak pabrik-pabrik menggunakannya sebagai dasar untuk sistem automasi. Hal ini menjadikan PLC memiliki kemungkinan akan tetap menjadi paling utama untuk masa yang akan datang. PLC sendiri dapat dikombinasikan dengan sebagian besar teknologi yang menyediakan sistem kontrol dan sistem pemantau. Banyak PLC telah digunakan dalam pabrik-pabrik *manufacturing* untuk beberapa aplikasi seperti pemantau keamanan, pengatur pemakaian energi, mesin kontrol dan alur produksi otomatis. PLC juga dapat digunakan pada aplikasi-aplikasi yang erat kaitannnya dengan sistem keamanan, diantaranya pada reaktor nuklir sebagai *Monitoring Security* yang mencegah terjadinya proses *Fail-to-Danger Fault* yaitu proses yang menyebabkan terjadinya kesalahan yang mengakibatkan sistem kontrol tidak bereaksi terhadap peringatan bahaya. Untuk lebih mengenal dan mendalami penggunaan dari fungsi PLC *(Programmable logic Controller)* maka dibuatlah simulasi PLC yang diharapkan akan memudahkan pemakaian dan memberikan gambaran mengenai PLC untuk belajar dan lebih mengenal PLC tanpa harus menggunakan perangkat keras tersebut.

Pengertian PLC (*Programmable Logic Controllers*)

PLC merupakan sebuah perangkat yang digunakan sebagai pengganti rangkaian *relay* yang digunakan untuk kontrol mesin. Dengan menggunakan program untuk berinteraksi dengan PLC, maka PLC akan bekerja sesuai dengan *input* yang dimasukkan oleh *user* dan akan menghasilkan *output* dengan kondisi hidup atau mati. Konsep PLC, sesuai dengan namanya adalah sebagai berikut : (1) ***Programmable:*** Menunjukkan kemampuan yang dapat dengan mudah diubah sesuai *input* yang dibuat oleh *user*.; (2) ***Logic***, Menunjukkan kemampuannya dalam memproses *input* secara *logic*, yakni melakukan operasi logika AND, OR, dan NOT; dan (3) ***Controller:*** Menunjukkan kemampuan dalam mengontrol dan mengatur proses sehingga menghasilkan *output* yang diinginkan.

Pemrograman PLC

Secara umum bahasa pemrograman yang banyak digunakan dalam pemrograman PLC adalah LDR *(Ladder Diagram)* dan STL (Statement List). Bahasa pemrograman yang digunakan pada PLC adalah: (1) LDR *(Ladder Diagram)*, yaitu merupakan bahasa pemrograman yang berbentuk diagram kontak dan aktuator yang menyerupai suatu rangkaian listrik. LDR dapat dipakai untuk membuat program dari beberapa tugas pengontrolan, khususnya jika tersedia suatu diagram sirkuit; dan (2) *STL (Statement List)* yang merupakan pengembangan dari bahasa BASIC yang merupakan bahasa pemrogaraman yang masih menggunakan format teks.

Logika Operasi PLC

Proses untuk mengubah obyek kontrol menjadi sebuah program logika pada PLC membutuhkan pemikiran yang terstruktur. Dengan *Boolean Algebra* ini, dapat membantu memberikan konsep yang dibutuhkan untuk menganalisa dan mendesain program logika PLC ini. Keunggulan dari aljabar *Boolean* adalah dapat menyederhanakan sebuah sistem logika kedalam sebuah persamaan. Persamaan tersebut dapat disederhanakan dan diubah kedalam bentuk baru. Aljabar *Boolean* dapat digunakan pada pemrograman logika *ladder* dengan sangat baik. Persamaan *Boolean* terdiri dari operasi dan variabel yang serupa dengan persamaan aljabar biasa. Ketiga operator dasar adalah AND, OR dan NOT. Selain itu terdapat pula operator yang lain yang lebih kompleks yaitu *exclusive or* (XOR), *not and* (NAND) dan *not or* (NOR).

Komponen Utama PLC

1. *Normally Open*

Komponen ini merupakan salah satu komponen *input* dari PLC yang mempunyai sifat bekerja pada kondisi bit ON. *Normally Open* bernilai "0" jika berada pada kondisi *false*/salah dan bernilai "1" jika berada pada kondisi *true*/benar.

2. *Normally Close*

Normally Closed memiliki prinsip kerja kebalikan dari *Normally Open* yaitu bekerja pada kondisi bit *OFF*.

3. *Flag*

Flag merupakan sebuah *internal memory* 1 bit yang mana akan aktif jika *input* berada pada kondisi *true*/benar. *Flag* menyimpan hasil yang diinginkan *user*, yang pada nantinya dibutuhkan dalam pemrograman *sequence* (gerak sekuensial). *Output flag* penyimpan hasil akan menjadi *input flag* pemanggil yang berfungsi sebagai *trigger* (pemicu) untuk mengoperasikan logika tertentu hingga menghasilkan *output* yang sebenarnya.

4. *Timer*

Timer merupakan register yang berfungsi menghitung maju jumlah suatu nilai berdasarkan *clock rate* secara otomatis (tidak berdasarkan *input*). Pertama-tama *user* akan menentukan berapa nilai *timer* yang diberikan, setelah itu *timer* akan menghitung secara otomatis dan berdasarkan *clock rate* untuk mencapai nilai tersebut. Jika sudah sama dengan nilai *timer* yang sebelumnya telah diberi nilai *input* oleh *user* maka akan menjadi *input* pada *Timer* pemanggil pada logika berikutnya.

5. *Counter*

Counter merupakan suatu *register* yang berfungsi untuk menghitung jumlah pulsa detak dengan alur maju dan mundur berdasarkan *input* yang diberikan kepada komponen *Counter* ini. Pada PLC, *Counter* dijadikan sebuah *buffer* (penyimpan data) lalu akan dijadikan *input* pada *Counter* pemanggil. *Counter* akan mulai menghitung ketika berada pada logika *true*/benar dan menyimpannya sebagai *buffer* sampai pada nilai yang telah ditetapkan oleh *user*, setelah itu akan lanjut ke proses berikutnya.

***Basic Instruction* PLC**

Basic Instruction PLC merupakan instruksi dasar tertentu yang digunakan untuk mendesain sebuah logika *ladder*. Beberapa instruksi dasar yang sering digunakan adalah :

1. LOAD
2. LOAD NOT
3. AND
4. AND NOT
5. OR
6. OR NOT
7. AND LOAD
8. OR LOAD

PEMBAHASAN

Desain Sistem Simulator PLC

Desain sistem simulator PLC secara keseluruhan terbagi menjadi beberapa tahap:

1. ***Input Project***

Pada tahap ini ada 2 proses dalam *input project*, yaitu *new project* dan *open project*. Pada proses *new project*, *user* membuat *project* baru. Sedangkan pada proses *open project*, *user* membuka atau me-*load project* yang telah dibuat.

2. ***Programming***

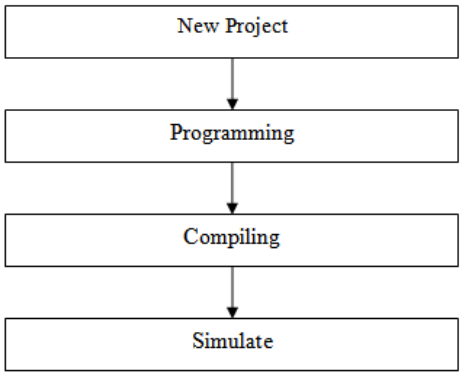
Pada PLC *programming* terdapat 2 jenis bahasa pemrograman, yaitu *ladder* dan STL (*Statement List*). Penerapan logika pada *ladder* adalah dengan interaksi antara *input*, *output*, *connector*, *timer*, *counter*, *oropen* dan *orclose* yang saling berkaitan satu sama lain sehingga menghasilkan logika operator, seperti logika AND, OR, ANDLOAD, ORLOAD. Penggunaan logika tersebut juga dapat menghasilkan *sequential movement*. Pada STL tidak jauh berbeda dengan penerapan logika *ladder*, yang membedakannya adalah visualisasi dalam penerapan logika programnya. Dimana pada STL proses perancangan logika berdasarkan penulisaan *coding*. Selain itu juga, pada bagian ini terdapat proses *Scanning* dan *logic operation*.

3. ***Compiling***

Hasil *programming* baik itu menggunakan *ladder* ataupun STL yang sudah selesai, akan diproses melalui tahap-tahap *compiling*, seperti *Error checking*.

4. ***Simulate***

Real time scanning logic operation (Pembacaan operasi logika secara langsung) yang dilakukan pada *simulate mode* akan memberikan hasil keluaran pada modul *output* sesuai dengan logika program yang dilakukan pada tahap *programming*.



Gambar 1 Sistem *simulator* PLC

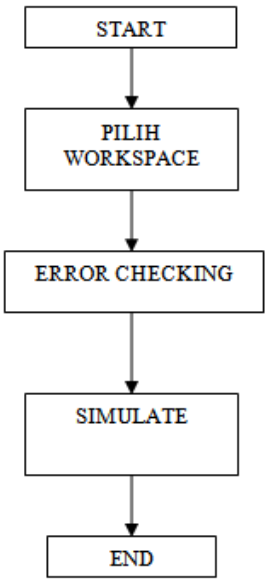
Perancangan Peraturan M+

Peraturan M+ adalah bahasa pemrograman yang dibuat untuk memudahkan *user* dalam penulisan *coding* STL. *Keyword* atau kata kunci pada Peraturan M+ dibuat merujuk pada kata kunci yang biasa digunakan pada persamaan *Boolean algebra*, seperti AND, OR, dan NOT. Kemudahan yang ditawarkan disini adalah dengan tidak terlalu banyaknya *keyword* (kata kunci) yang diberikan, serta banyaknya *Input* dan *Output* yang sudah ditentukan berdasarkan penomerannya. Secara penulisan program, Peraturan M+ tidak *case sensitive* (tidak membedakan huruf besar maupun huruf kecil dalam penulisan program) sehingga akan memperkecil persentase kesalahan yang ditulis oleh *user*.

Secara garis besar penulisan program dengan menggunakan Peraturan M+ adalah *output* harus ditulis pertama kali sebagai acuan bahwa *output* tersebut merupakah hasil keluaran dari logika pemrograman. Lalu diikuti tanda *brance* pembuka "{" yang diletakkan sesudah *output* sebagai tanda awal dari perintah-perintah logika pemrograman yang akan ditulis. Tanda *brance* penutup "}" menunjukkan akhir dari perintah-perintah logika pemrograman. Penandaan *brance* pembuka "{" dan *brance* penutup "}" menunjukan bahwa perintah-perintah logika pemrograman tersebut berlangsung dalam 1 rung.

Metode pembacaan pada Peraturan M+ menggunakan apa yang dinamakan metode prioritas. Metode prioritas itu sendiri adalah apabila pada penulisan program bertemu dengan tanda baca "(" maka prioritas akan bertambah 1 tingkat dan apabila bertemu dengan tanda baca ")" maka prioritas akan dikurangkan 1 tingkat. Proses pembacaan Peraturan M+ berdasarkan tingkat prioritas yang terbesar lalu diturunkan hingga menghasilkan tingkat prioritas terkecil yaitu prioritas 0.

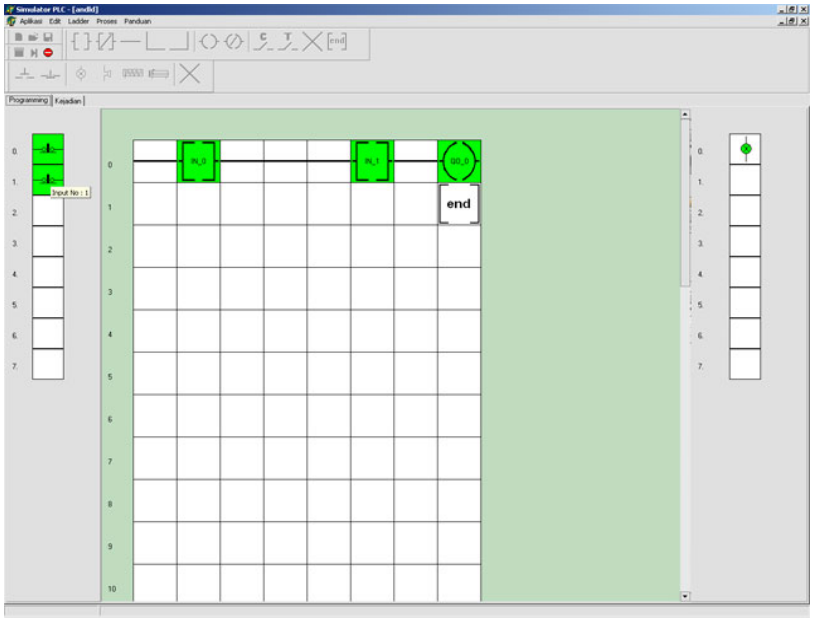
Berikut pada gambar 2 merupakan diagram blok dari keseluruhan sistem simulator PLC



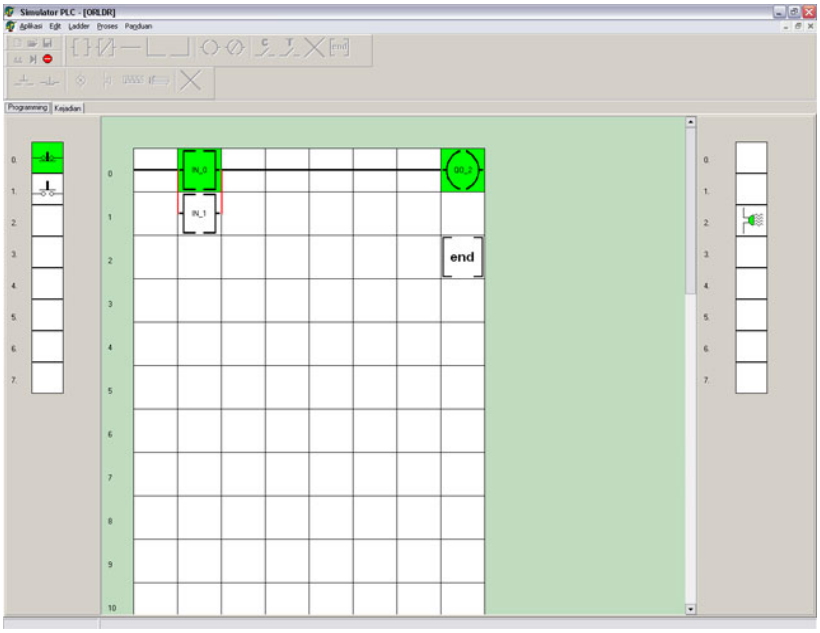
Gambar 2 Diagram Blok Sistem Simulator PLC

Evaluasi Hasil Percobaan

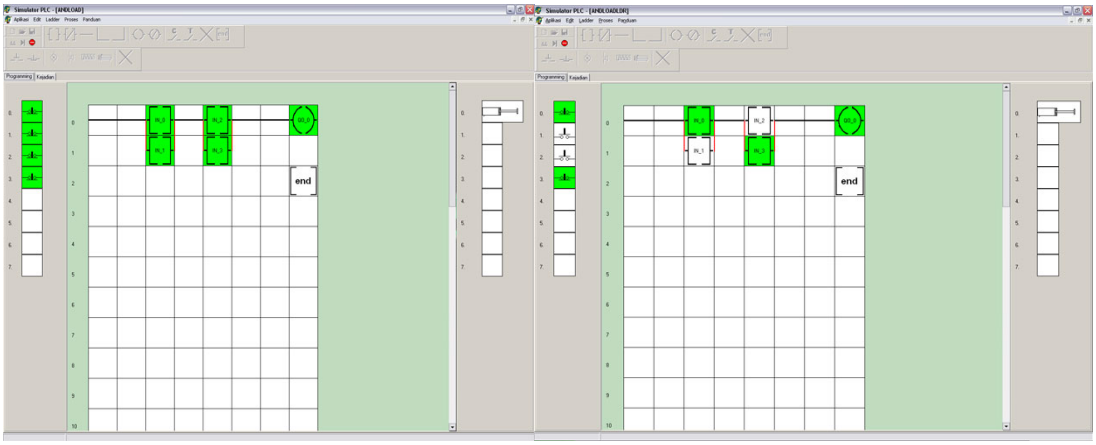
Evaluasi hasil percobaan dilakukan dengan mendesign *input-input* pada LDR sesuai dengan bentuk persamaan yang ada seperti AND, OR, AND LOAD dan OR LOAD. Seperti pada gambar berikut ini :



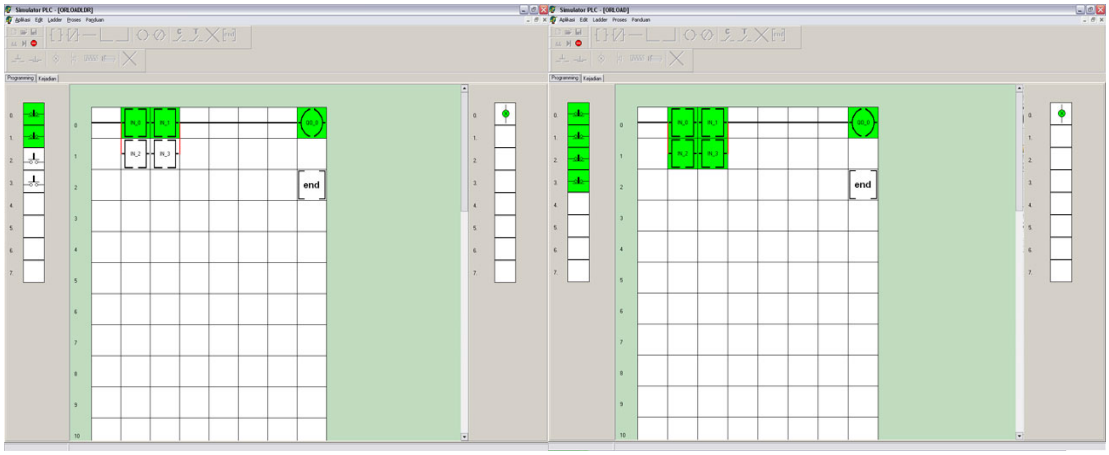
Gambar 3 Simulasi Operasi AND pada LDR



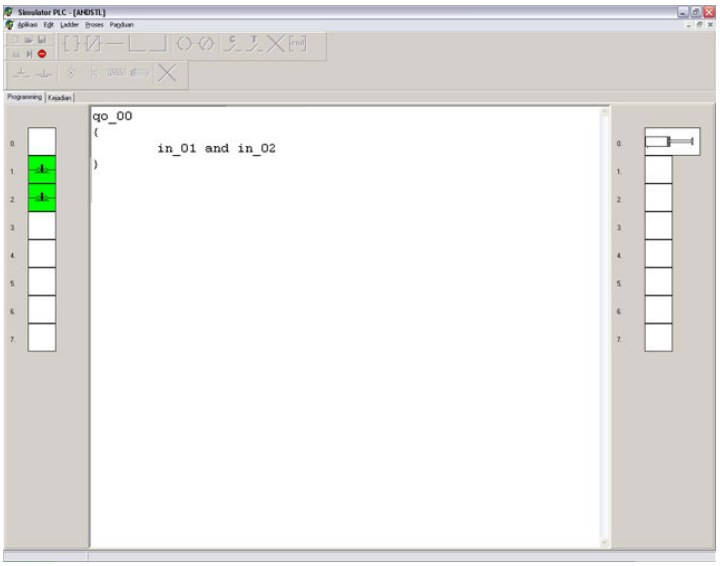
Gambar 4 Simulasi Operasi OR pada LDR



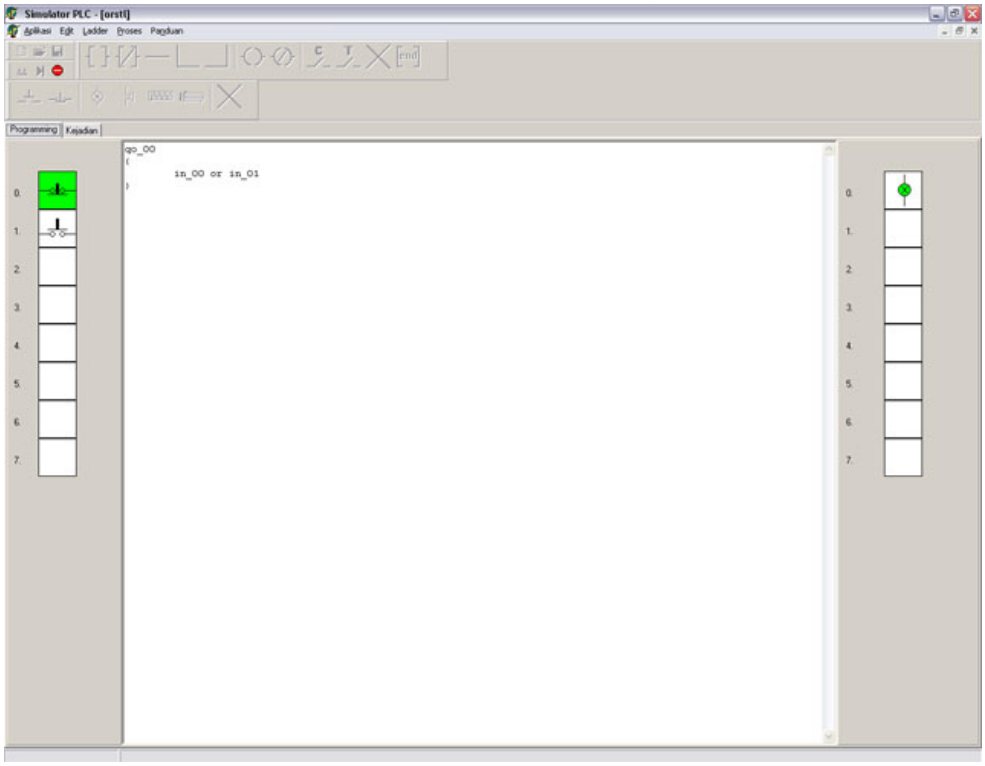
Gambar 5 *Output* simulasi logika AND LOAD pada LDR



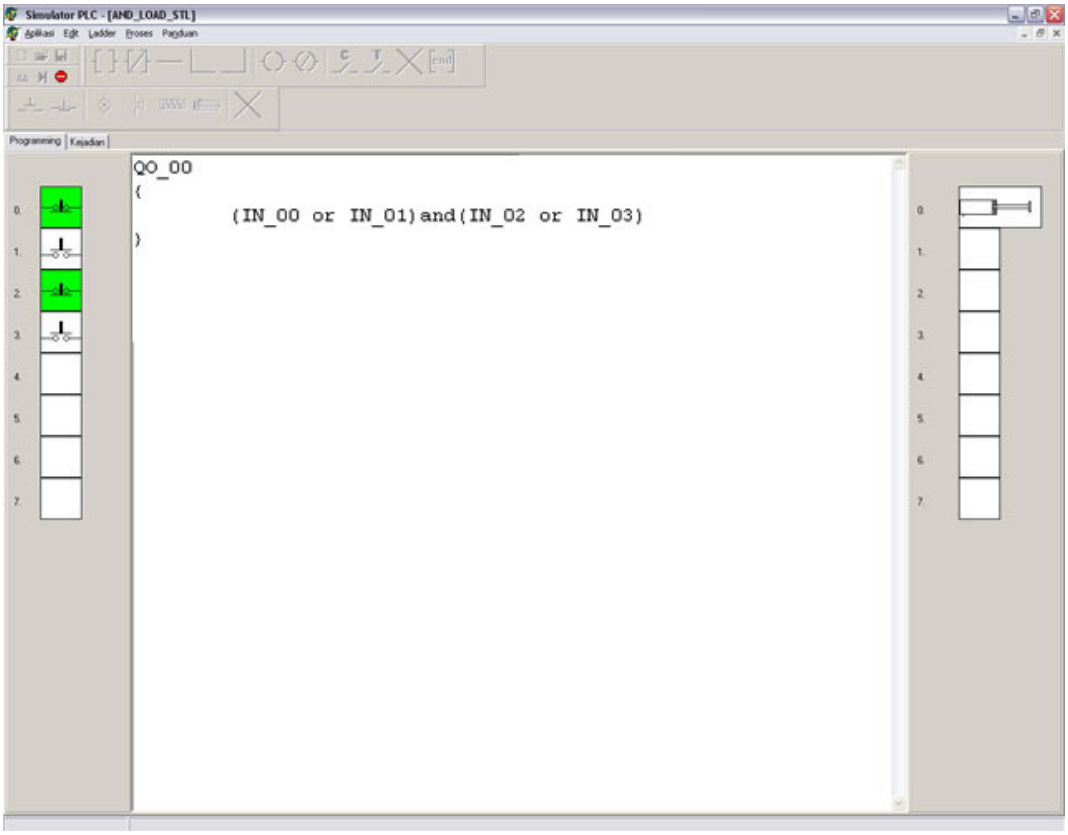
Gambar 6 *Output* simulasi logika OR LOAD pada LDR



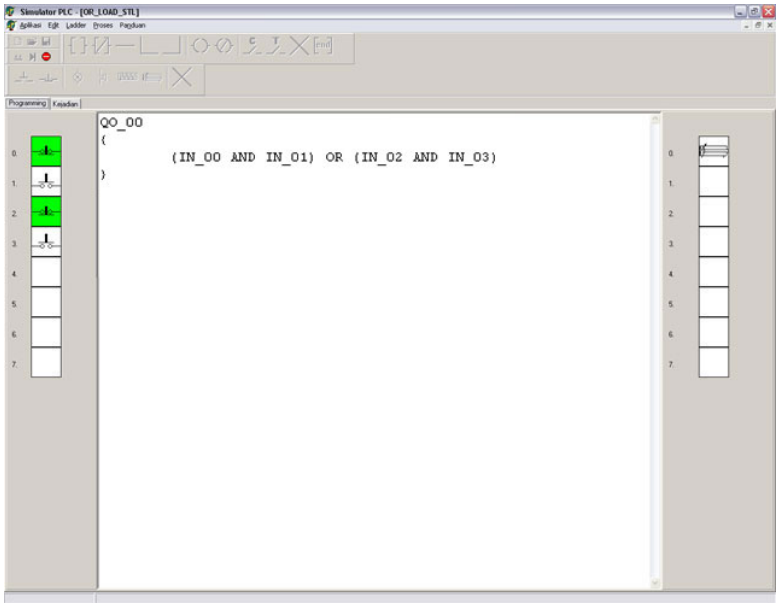
Gambar 7 *output* simulasi logika AND STL



Gambar 8 *output* simulasi logika OR STL (satu input ditekan)



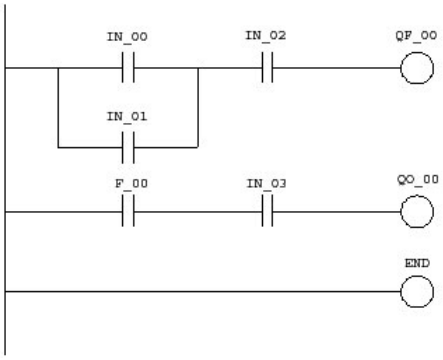
Gambar 9 *output* AND LOAD aktif



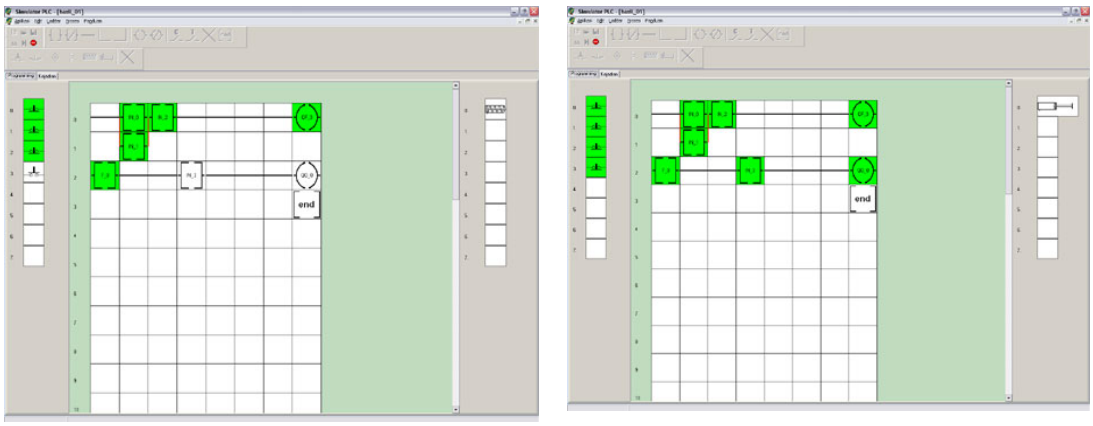
Gambar 10 *output* OR LOAD tidak aktif

Berikut adalah gambar-gambar dari hasil percobaan menggunakan *flag*, *timer* dan *counter* :

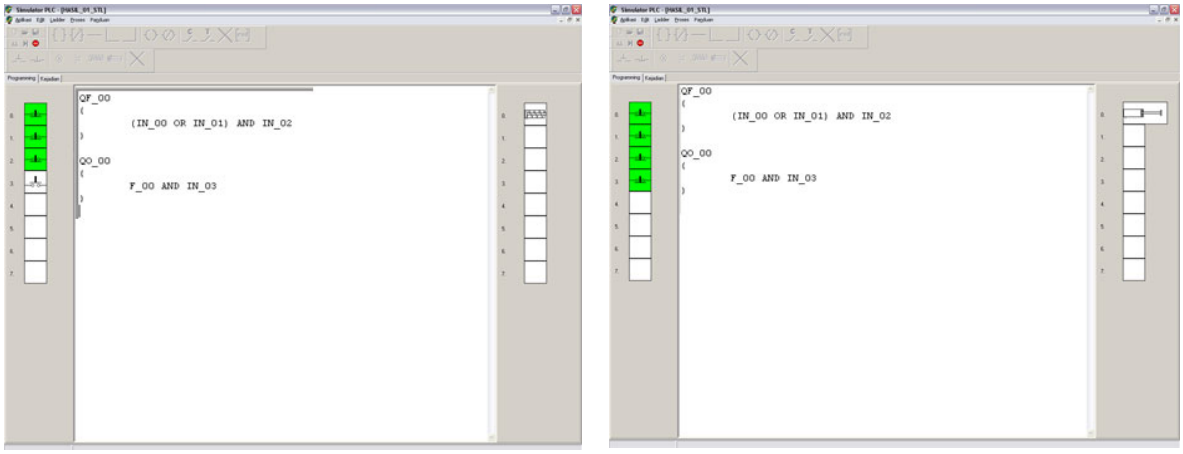
1. Percobaan 1



Gambar 11 Perancangan Logika PLC Menggunakan *Flag*

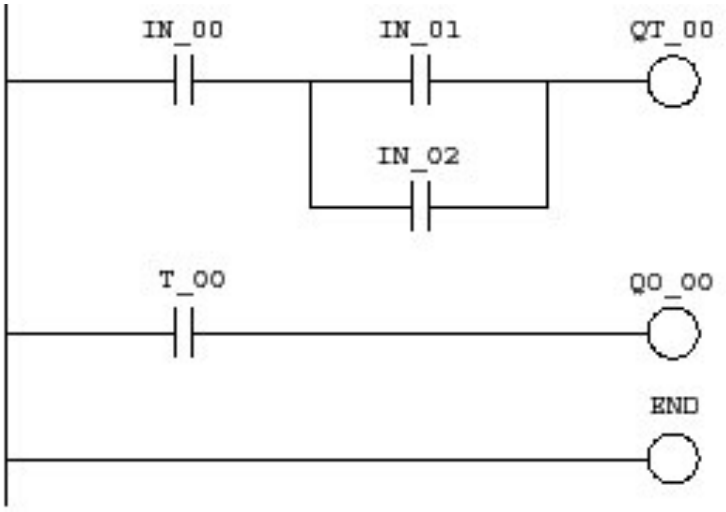


Gambar 12 Hasil Simulasi *flag* Menggunakan *Ladder*

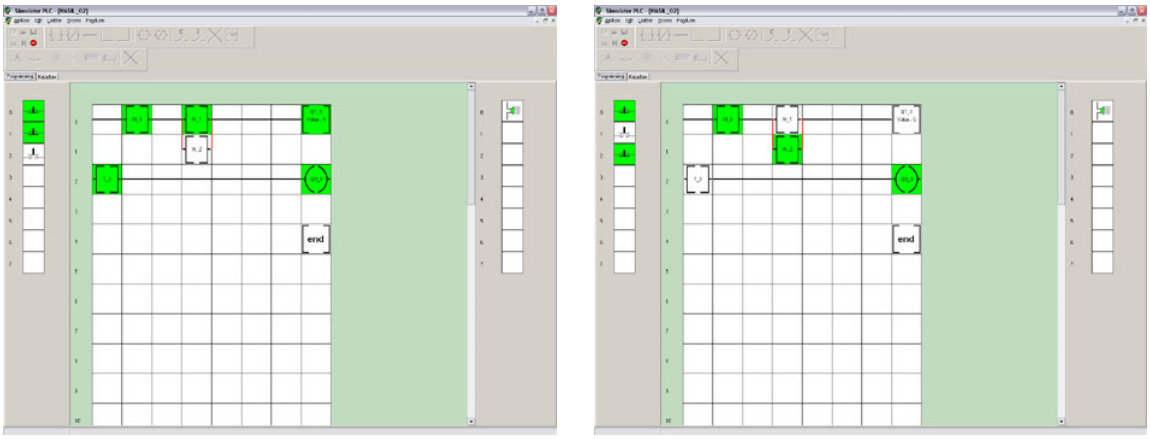


Gambar 13 Hasil Simulasi *flag* Menggunakan STL

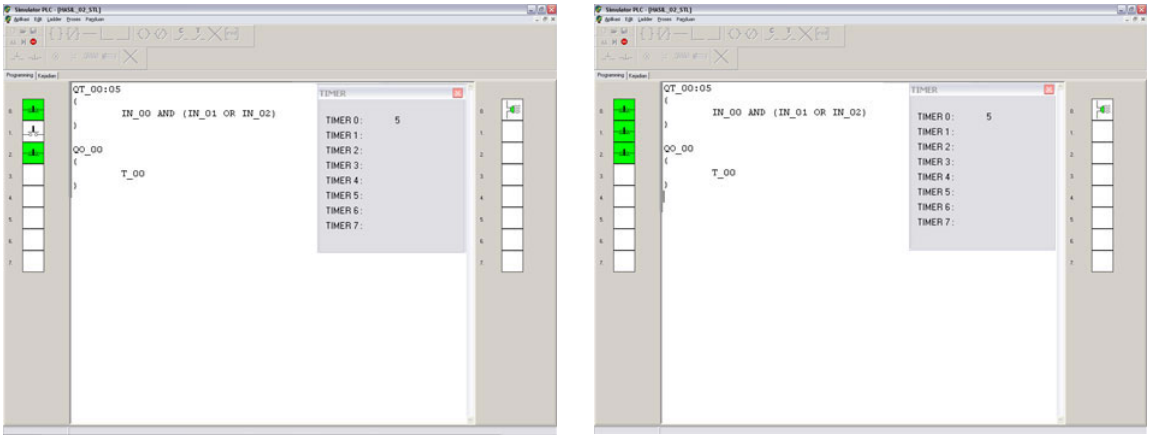
2. Percobaan 2



Gambar 14 Perancangan Logika PLC Menggunakan *Timer* (T_00 : 5detik)

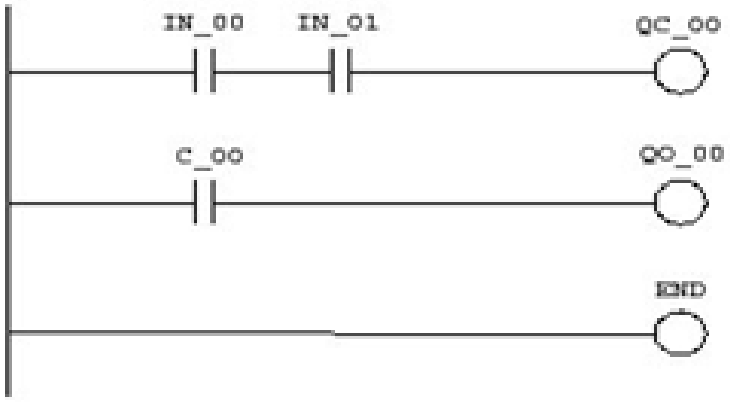


Gambar 15 Hasil Simulasi *Timer* Menggunakan *Ladder*

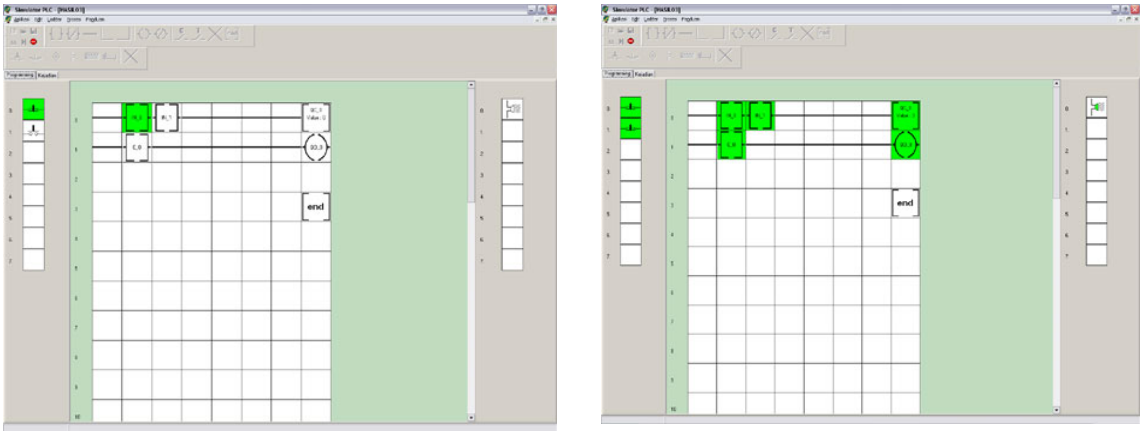


Gambar 16 Hasil Simulasi *Timer* Menggunakan STL

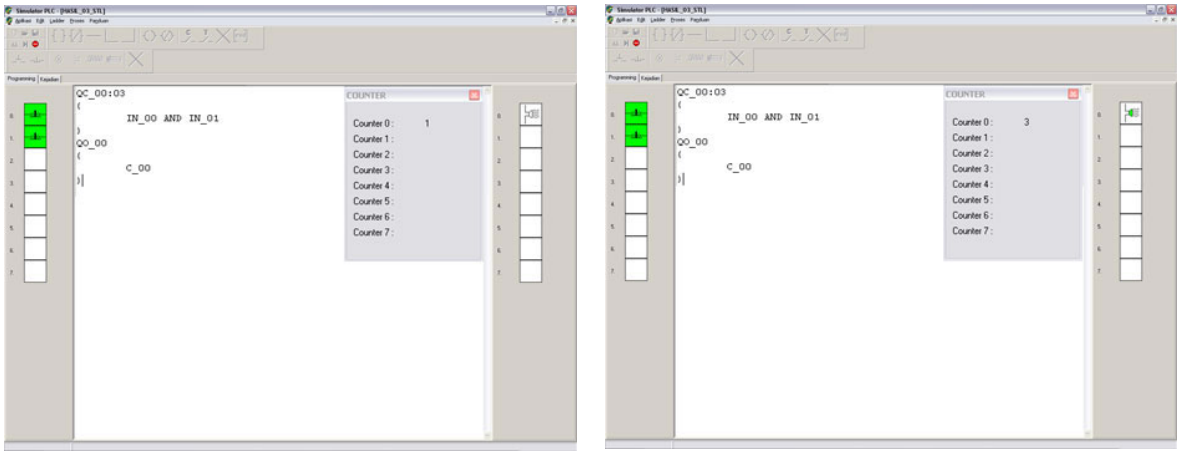
3. Percobaan 3



Gambar 17 Perancangan Logika PLC Menggunakan *Counter* (C_00 : 3)



Gambar 18 Hasil Simulasi *Counter* Menggunakan *Ladder*



Gambar 19 Hasil Simulasi *Counter* Menggunakan STL

SIMPULAN

Dari penelitian ini dapat disimpulkan beberapa hal, yaitu: (1) Aplikasi ini menggunakan standar GUI *(Graphic User Interface)* yang bertujuan untuk memudahkan *user* dalam penggunaan aplikasi ini; (2) Logika operasi simulasi pada program ini mengacu pada teori *Boolean Algebra*; (3) Simulator PLC berbasiskan Delphi ini bersifat *offline* (tidak dapat di-*upload* ke PLC sesungguhnya); (4) Kendala dari penelitian ini adalah banyaknya kemungkinan yang dapat terjadi pada tiap operasi logika. Untuk membatasinya dibatasi pula jumlah *input* dan *outputnya*; (5) Hanya bisa mensimulasikan sebuah *output* pada tiap operasi logika, selain itu *output* tidak dapat diparalel. Apabila ingin mensimulasikan lebih dari 1 *output* operasi logika dapat menggunakan *flag*; (6) Dalam program ini, operasi logika pada PLC dapat dijalankan baik pada *ladder diagram* maupun STL dan juga dapat disimulasikan sehingga dapat langsung diketahui hasil dari perancangan logika tersebut; (7) Pada STL *programming* menggunakan peraturan M+, dengan menggunakan parameter-parameter yang juga mengacu pada teori *Boolean Algebra* seperti AND, OR dan NOT; (8) Pada peraturan M+ logika pembacaan STL dimulai pada prioritas yang terbesar; dan (9) Logika pembacaan *ladder diagram* dimulai dari *input* sebelah kiri lalu *input* sebelah kanan atau dari *input* pada bagian atas lalu ke *input* yang berada dibawahnya.

DAFTAR PUSTAKA

Ackermann, R.; Franz, J.; Hopf, A.; Kantel, M.; Plagemann, B. (1994) *Programmable Logic Controllers Basic Level TP 301 Textbook*. Germany: Festo Didactic

Hsieh, S. J., Hsieh, P. Y. (2002) Web-Based Programmable Logic Controller Learning System. *32nd ASEE / IEEE Frontiers in Education Conference*, p6.

Kadir, A. (2001). *Dasar Pemrograman Delphi 5 Jilid 1*. Andi, Yogyakarta

Malvino, A. P. (1991). *Elektronika Komputer Digital Pengantar Mikrokomputer Edisi Kedua*. Erlangga, Jakarta

Pressman, R. S. (2002) *Rekayasa Perangkat Lunak Pendekatan Praktisi (Buku 1)*. Terjemahan Harnaningrum, L.N. Andi, Yogyakarta