Prosiding Nasional Rekayasa Teknologi Industri dan Informasi XIII Tahun 2018 (ReTII)
November 2018, pp. 158~165
ISSN: 1907-5995

Pengembangan Simulator Perangkat I/O dengan Pemrograman Ekspresi Boolean dan Javascript

Pratikto[1], Raydha Zul Fitriani[2]
[1, 2]Jurusan Teknik refrigerasi dan Tata Udara, Politeknik Negeri Bandung
Korespondensi : pratikto@polban.ac.id

ABSTRAK

Ekspresi boolean merupakan hasil yang penting dari suatu algoritma penerjemahan dari suatu diagram *ladder*. Di era penggunaan PLC dan diagram *ladder* pada pemrograman perangkat digital telah marak, maka VHDL memerlukan penerjemahan kembali ke ekspresi boolean yang setara dengan diagram *ladder* untuk dijalankan dalam program. Elemen dari pemrograman berbasis diagram *ladder* tidak saja terdiri dari operasi dasar seperti *And*, *Or*, atau *Not*, namun berisi juga elemen dasar tambahan seperi *comparator*, *timer* dan *counter*. Dalam makalah ini dibahas sebuah pemrograman berbasis skrip yang mengutamakan ekspresi boolean, tiap baris dari skrip tidak lain merepresentasikan sebuah *rung* suatu diagram *ladder*. Pemrograman berbasis skrip ekspresi boolean yang dikembangkan ini diharapkan menjadi sebuah awal dari cara baru memprogram perangkat I/O menggunakan sintaks Javascript disamping metode pemrograman *Structured Text* yang telah ada. Sebuah simulator I/O berbasis pemrograman web dibangun untuk memvisualisasikan skrip ekpresi boolean yang ditulis. Skrip terdiri dari dua bagian yakni bagian utama merupakan ekspresi boolean dan skrip grafik dan visualisasi lain disesuaikan dengan sintaks penulisan Javascript. Ekspresi boolean padanan *ladder* lebih ringkas dan memperkecil luas pandang dalam menguji jalannya program dengan tambahan visualisasi hasil. Algoritma *timer* yang dirancang diterapkan dalam skala waktu detik. Nilai kesalahan untuk *on delay* sekitar 0,062 detik dan untuk *off delay* sekitar 0,046 detik.
Kata kunci: Ekspresi boolean, *ladder*, I/O, Javascript, simulator

ABSTRACT

Boolean expressions are an important result of a translation algorithm from a ladder diagram. Having a PLC and a ladder diagram on digital device programming has grown, VHDL requires translating back to boolean expressions that are equivalent to ladder diagrams to run in the program. Elements of ladder diagram-based programming not only consist of basic operations such as And, Or, or Not, but also contain additional basic elements such as comparators, timers and counters. In this paper we discuss a script-based programming that contains boolean expressions for each row of scripts, but represents a rung of a ladder diagram. Script-based programming developed using Javascript boolean expressions syntax is expected to be the beginning of a new way of programming I/O devices in addition to existing *Structured Text programming methods*. *An I/O simulator based on web programming was built to visualize the written boolean expressions. The script consists of two parts, the main part is boolean expressions beside of scripts for graphical and other visualizations adapted to the Javascript writing syntax. Programing with use boolean expressions of ladder equivalents are more concise and minimize broad view in testing the course of programs with additional visualization of results. The timer algorithm is designed to be applied on a second time scale. The error value for on delay is around 0.062 seconds and for off delay around 0.046 seconds.*
Keyword*:* Boolean expressions, ladder, I/O, Javascript, simulator

1. PENDAHULUAN

Aplikasi pemrograman dalam kontrol automatik memerlukan berbagai teknik pemrograman yang dapat mempermudah penyusunan dan implementasi suatu algoritma kontrol. Telah diketahui sejak lama pemrograman berbasis simbol *ladder* sangat memudahkan dalam menangani kasus pengontrolan variabel diskrit yang diimplementasikan dalam perangkat logik seperti *relay* dan sebagainya. Pemrograman *ladder* tidak lain merupakan aplikasi dari representasi dari ekspresi boolean. Menggunakan diagram *ladder*, logika boolean sangat mudah difahami dan ditelusuri. Dalam era VHDL penerjemahan kembali dari diagram *ladder* menjadi ekspresi boolean menjadi penting karena pada akhirnya logika tersebutlah yang dikerjakan oleh perangkat elektronik untuk menyampaikan data logika pada aktuator, sensor ataupun pada perangkat elektronik lainnya.

Dalam penelitian ini akan dikaji penerapan ekspresi boolean setara diagram *ladder* untuk mempermudah penulisan program yang menyangkut pengontrolan pada perengkat diskrit. Secara perangkat

keras sebuah perangkat seperti aktuator diskrit misalnya *relay* digerakkan melalui pemrograman mikrokontroler yang diprogram menggunakan teknik pemrograman berbasis skrip.

Pemrograman berbasis skrip tidak secara langsung mengadopsi ekspresi boolean bila logika yang melibatkan masukan dan keluaran secara boolean diperlukan dalam program. Ekspresi dalam bentuk persamaan boolean tidak biasa digunakan dalam pemrograman berbasis skrip. Sebuah program skrip untuk memberikan nilai *true* atau *false* sebuah variabel keluaran yang bergantung pada variabel masukan boolean pula biasanya ditulis dengan instruksi skrip if-else. Bila secara skrip sebuah program ditulis dengan logika sebagai berikut:

Inisialisasi x1= false, x2 = false, y = false
Jika x1 = true maka y = true
Jika x2 = true maka y = false

Implementasi ekspresi boolean penulisan skrip menjadi

y = (y Or x1) And Not x2

Penggunaan ekspresi boolean akan menjadi lebih menyederhanakan skrip bila terdapat banyak logika boolean yang harus digunakan dalam pemrograman. Dalam pemrograman *ladder* dikenal istilah *rung*. Satu *rung* dapat dinyatakan dalam satu ekspresi boolean. Pemrograman skrip yang melibatkan *timer* dan *counter* akan menjadi lebih kompleks bila tidak mengimplementasikan ekspresi boolean.

Program untuk sistem berbasis mikroprosesor harus dimuat ke dalam kode mesin yang merupakan urutan angka kode biner untuk mewakili instruksi program. Namun, dapat digunakan bahasa Assambly yang berdasarkan penggunaan mnemonik; misalnya, LD digunakan untuk menunjukkan operasi yang diperlukan untuk memuat data yang mengikuti LD, dan program komputer yang disebut assembler digunakan untuk menerjemahkan mnemonik ke dalam kode mesin. Pemrograman dapat dibuat lebih mudah dengan menggunakan apa yang disebut bahasa tingkat tinggi, seperti C, BASIC, PASCAL, FORTRAN, COBOL. Namun, penggunaan metode penulisan program seperti ini membutuhkan keterampilan dalam pemrograman. Penulisan dengan cara lain yaitu *ladder* dikembangkan untuk PLC dimaksudkan untuk digunakan secara mudah tanpa pengetahuan pemrograman [1].

Ada dua bahasa tekstual: *Instruction List* (IL) dan *Structured Text* (ST) dan dua bahasa pemrograman grafis: *Ladder Diagram* dan *Function Blocks*. Ada juga bahasa pemrograman kelima yang disebut *Sequential Function Chart* (SFC) yang dapat diasumsikan sebagai bahasa grafis. IL mirip dengan bahasa assembly tingkat rendah. *POU* terdiri dari urutan instruksi. Setiap instruksi terletak pada garis yang terpisah. Instruksi dapat dimodifikasi oleh *modifier*, negasi, kondisionalitas, lompatan, pengembalian dan prioritas.

Ladder Diagram juga dikenal sebagai *Ladder Logic* yang merupakan bahasa pemrograman tertua untuk PLC. Hal ini didasarkan pada representasi grafis dari logika kontak *relay*, sehingga sangat cocok untuk mengekspresikan rangkaian boolean.

Ekspresi boolean berperan sangat penting dalam pemrograman. Perangkat elektronik seperti PLC tidak dapat mengeksekusi langsung diagram yang ditulis dalam pemrograman *ladder*. Yin [2] mengembangkan algoritma untuk menerjemahkan diagram *ladder* ke IL sebelum menyatakannya dalam ekspresi boolean.

Pengembangan PLC menggunakan FPGA memerlukan penerjemahan diagram *ladder* ke ekspresi boolean, Xie [3] juga mengusulkan suatu algoritma untuk menerjemahkan *ladder* ke ekspresi boolean.

Disamping pemrograman langsung pada PLC sebuah simulator juga diperlukan dalam menganalisis hasil program sebelum diterapkan. E. Pruna dkk [4] mengembangkan simulator proses industri yang kompatibel dengan PLC Siemens S7-1200 yang diimplementasikan dalam perangkat lunak LabVIEW. Sementara sebelumnya L. Brito Palma dkk [5] mengembangkan simulator PLC dengan pemrograman ST untuk PLC TSX3721 untuk mensimulasikan sistem diskrit dan hibrid. Simulator PLC dengan pemrograman dalam ST berbasis WEB dikembangkan juga oleh L Brito dengan kawannya yang lain [6]. Simulator ini menyediakan saluran *input* / *output* digital, saluran *input* / *output* analog dan juga interaksi dengan model proses dinamis tipe *autoregressive* dengan *eksogen input* (ARX).

Pemrograman berbasis skrip mempunyai maksud dan tujuan yang sama dengan pemrograman grafis seperti diagram *ladder* yang di gunakan pada PLC. Kedua metode pemrograman tersebut mempunyai kekuatan masing-masing. Kedua kekuatan itu dapat terintegerasi dengan memasang teknik pemrograman logika *ladder* dalam bentuk pemrograman berbasis skrip menerapkan ekspresi boolean. Setiap ekspresi boolean dapat mewakili setiap *rung* yang digunakan untuk pemrograman pada kasus pengontrolan keluaran dan masukan diskrit.

Sekalipun ekspresi boolean dapat diimplementasikan dalam pemrograman *Structured Text* untuk PLC dan simulator untuk ini telah dibangun oleh beberapa peneliti sebelumnya[4,5,6], pada penelitian yang dilakukan ini berbeda dari penelitian yang disebutkan di atas yaitu bahwa simulator yang dibangun tidak hanya bisa dijalan dengan PC tetapi juga perangkat lain yang didukung aplikasi Web Bowser. Juga Structured text yang digunakan dipilih berbasis Javascript sebagai bahasa pemrograman yang mendukung komunikasi global.

Sehingga penelitian ini mengkombinasikan segala kekuatan skrip, termasuk skrip yang sepadan dengan diagram ladder.

Dalam penelitian ini dikembangkan pula struktur skrip dan *tool* atau algoritma program berupa *function counter* dan *timer*. *Function* yang dikembangkan diharapkan dapat diterjemahkan dalam pemrograman berbasis *script* yang lain seperti Basic, C, C#, C++ dan lain-lain serta untuk pemrograman skrip untuk berbagai mikrokontroler.

Tiga *point* hasil atau kontribusi dari penelitian ini yaitu pertama bahwa penelitian ini merupakan pendahuluan dalam usaha menerapkan logika pemrograman *ladder* yang ditulis dalam sitaks Javascript untuk selanjutnya dikembangkan pada *programable device* seperti mikrokontroler dan sebagainya.

Kedua, dari penelitian ini dihasilkan pula dasar algoritma pendeklarasian *function* atau *tool timer* dan *counter* serta struktur penulisan skrip yang fleksibel untuk dikembangkan dan ditambahkan oleh pengguna sesuai kebutuhan. Ketiga adalah sebuah simulator pemrograman berbasis skrip perangkat *input*/*output* yang disusun menggunakan kode Javascript dalam halaman HTML.

2. METODE PENELITIAN

Metode dalam penelitian ini didasarkan pada kebutuhan pemrograman skrip untuk mengendalikan keluaran dari beberapa kemungkinan masukan yang diberikan pada simulator perangkat I/O. Pada pemrograman untuk pengontrolan sistem diskrit ekspresi boolean yang melibatkan operasi *And*, *Or*, dan Negasi atau *Not* tidaklah cukup. Karena proses *input* dan *output* akan juga melibatkan proses aksi yang bergantung waktu dan hitungan. Objek *timer* dan *counter* [7], setidaknya harus juga diintegerasikan dalam skrip maka pada penelitian secara garis besar dilakukan beberapa kegiatan sebagai berikut:

1. Merancang dan menguji algoritma *timer* dan *counter* yang dapat dikombinasikan dengan ekspresi boolean standar.
2. Merancang dan menguji algoritma utama program untuk menangani variable *input* dan *output*.
3. Merancang dan menguji GUI untuk mensimulasikan pemrograman skrip yang melibatkan banyak ekspresi boolean.
4. Mengevaluasi teknik pemrograman skrip yang melibatkan ekspresi boolean dalam hubungannya dengan pemrograman yang ditulis dalam bentuk diagram *ladder*.

Pelaksanaan dari seluruh tahap-tahap di atas akan dikerjakan dengan menggunakan pemrograman Javascript sebagai bahasa pemrograman berorientasi objek fungsional dinamis yang tidak hanya dapat digunakan untuk memperkaya halaman web, tetapi juga untuk mengimplementasikan berbagai jenis aplikasi web, termasuk simulasi berbasis web. Program dapat dijalankan pada perangkat *front-end* : ponsel, tablet dan komputer desktop, serta pada *back-end* komputer yang handal, mungkin juga di beberapa infrastruktur *cloud*. Kinerja Javascript cukup untuk banyak jenis simulasi dan mengungguli pesaingnya dalam hal kemudahan penggunaan dan produktivitas pengembang, apalagi untuk simulasi berbasis web [8]. Simulator dirancang dapat dijalankan dalam Web Browser Chrome. Perancangan dan pengujian akan melibatkan perangkat keras (Tablet/PC) dan perangkat lunak *Text Editor*.

Sebagaimana tujuan penelitian ini yakni menghasilkan sebuah simulator I/O yang dapat diprogram dengan skrip berbasis ekspresi boolean dasar (logika *And*, *Or*, dan *Not*) maka juga logika yang melibatkan *timer* dan *counter* akan dibentuk.

Sifat dari *timer* dirancang berdasarkan sifat dasarnya yaitu satu *input* dan satu *output*. Status *input* dari *false* ke *true* akan mengaktifkan *timer* untuk memberikan status *output true* setelah waktu *preset*. Objek *counter* disusun dengan sifat dasar dengan tiga masukan dan satu keluaran. Jumlah sinyal *true* dari masukan pertama dikurangi dengan jumlah sinyal *true* dari masukan kedua, bila jumlah sesuai dengan nilai *preset* maka keluaran *counter* bernilai *true*. Masukan ke tiga adalah untuk mendeteksi perintah *reset* ketika berstatus *true*.

Simulator ini juga menerima perintah Javascript standar untuk menampilkan animasi atau simulasi plant yang dikontrol. Dua perintah dasar yang telah disediakan adalah menggambar garis dan busur. Kombinasi dari keduanya dapat digunakan untuk menampilkan sebuah animasi *level tank* yang dijadikan sebuah contoh untuk mendemonstrasikan program yang ditulis dengan skrip ekspresi boolean.

Struktur file text yang dapat dibuka dengan aplikasi ini tersusun seperti berikut:

```
/* Komentar tentang penjelasan ekspresi Boolean yang dituliskan*/
//komentar per baris skrip
Q0=I0; // ini contoh sebuah baris ekspresi boolean
//Var, penulisan komentar Var
/* Komentar Var menandai inisialisai variabel yang digunakan dalam skrip simulasi dan perintah
dasar tampilan dari suatu simulasi */
```

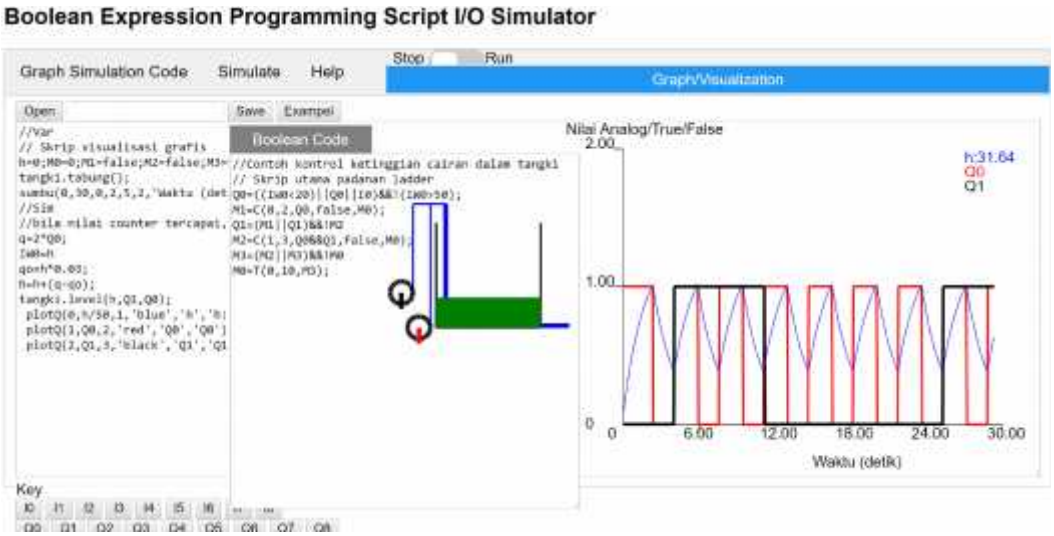
M0=false; h=0;// contoh penulisan inisialisasi variabel yang digunakan
// Sim, komentar Sim mengawali skrip kode dan dinamika simulasi misalnya:
h=h+1;

Menggunakan struktur di atas maka skrip file kosong berisi:
//Var
//Sim

Variabel *input* diskrit yang telah ditentukan dalam aplikasi ini adalah:
I0, I1, I2, I3, I4, I5, I6, I7, dan I8. Sementara variabel output adalah Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7 dan Q8. Variabel *input* dan *output* analog secara internal hanya ditentukan berjumlah dua masing masing adalah: IW0, IW1 dan QW0, QW1. Varibel lain dapat ditentukan dengan menuliskan setelah komemtar Var dalam *textarea* kode simulasi plant.

3. HASIL DAN ANALISIS

Program simulator I/O ini ditulis dalam kode Javascript dan HTML, maka program dijalankan dalam aplikasi Browser. Tampilan aplikasi diperlihatkan pada Gambar 3.1. Sebuah *textarea* disediakan untuk menuliskan program skrip ekspresi boolean, dan sebuah *textarea* lain digunakan untuk menuliskan kode Javascript untuk inisialisasi dan atau deklarasi variabel dan penggambaran grafik serta *plant* yang disimulasikan.



Gambar 3.1 Tampilan simulator.

Tombol simulasi *input* ditampilkan bersifat sebagai *momentary button*. Kode ekspresi boolean ditampilkan pada area dengan *header* Boolean Code yang dapat digeser. Penulisan kode untuk mengedit atau menambahkan variabel baru, menggambarkan grafik dan visualisasi *plant* yang disimulasikan ditampilkan pada tab Graph Simulation Code. *Input* analog dapat disimulasikan dengan mengaktifkan slider.

3.1. Algoritma Utama

Prinsip kerja secara garis besar perangkat lunak/aplikasi simulator yang dibangun ini diuraikan dalam algoritma sebagai berikut:

Algoritma Utama:

I. *Mulai*
II. *Tampilan utama*
III. *Pilih Tab atau switch* **Run** atau Close*(X)*
IV. *Bila dipilih Run lanjut ke tahap V.*
Bila dipilih Tab **Graph Simulation Code** *lanjut ke tahap VI.*
Bila dipilih Tab **Simulate** *lanjut ke tahap III.*
Bila dipilih Tab **Help**, *lanjut ke tahap VII.*

Bila dipilih Close *lanjut ke tahap VIII*

V. *Kode di jalankan, kembali ke tahap III.*

VI. *Menampilkan textarea proses pengeditan dapat dikerjakan, lanjut ke tahap III.*

VII. *Menampilkan penjelasan program dan contoh-contoh. Pilih* **Try it** *dan lanjut ke tahap III.*

VIII. *Selesai.*

3.2. Skrip *Timer* dan *Counter*

Penggunaan *timer* dan *counter* melengkapi proses jalannya program sehingga tidak hanya berjalan dari atas ke bawah, melainkan dapat juga berjalan berdasarkan waktu. Program juga harus dapat digunakan untuk menghitung perubahan sinyal yang diterima. Ketersediaan *timer* dan *counter* menjadi sebuah dasar yang harus dipenuhi dalam aplikasi pemrograman. Jumlah *timer* dan *counter* yang digunakan seharusnya tidak dibatasi sehingga perlu dirancang dan diimplementasikan algoritma untuk mendapatkan jumlah *timer* dan *counter* sesuai keperluan, namun tidak berarti membutuhkan logika dan baris program yang panjang dalam pembuatan *timer* dan *counter*. *Timer on delay* dan *timer on delay* dalam simulator yang di bangun diberi nama masing-masing T dan TOF dengan tiga argumen. Argumen T dan TOF masing masing menyatakan indeks, preset dan sinyal untuk mengaktifkan *timer*. Berikut ini adalah skrip untuk T dan TOF.

Skrip *timer on delay*

```
function T(n,d,v){
if (v==true){
  if (TimerInit[n]==0) {
    d1[n]=ttick;
   TimerInit[n]=1; }
}else{d1[n]=ttick;TimerInit[n]=0;}
var t=ttick-d1[n];
if (t<=d*1000) valTimer[n]=(ttick-d1[n])/1000;
if (t>=d*1000) {outTimer[n]=true;valTimer[n]=d;
}else {outTimer[n]=false;}
return outTimer[n];}
```

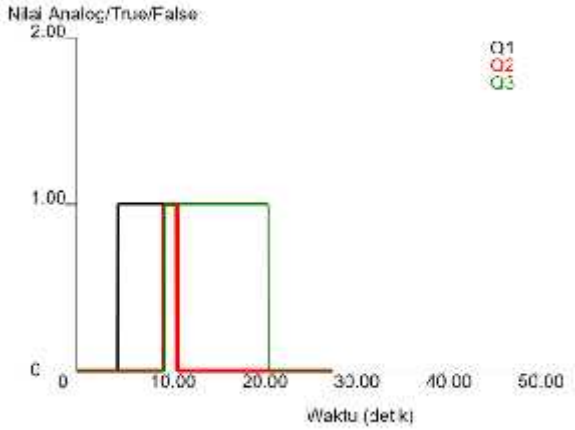
Skrip *timer off delay*

```
function TOF(n,d,v){
if (v){TimerTOFInit[n]=1;outTimerTOF[n]=true;}
if (!v&&!TimerTOFInit[n]==0) {
    d1TOF[n]=ttick;
    TimerTOFInit[n]=0;
    valTimerTOF[n]=0;}
if (TimerTOFInit[n]==0&&outTimerTOF[n]){
   var t=ttick-d1TOF[n];
   valTimerTOF[n]=t/1000;
   if                                    (t>=d*1000)
{outTimerTOF[n]=false;valTimerTOF[n]=d;}}
if(outTimerTOF[n]==undefined)
outTimerTOF[n]=false;
return outTimerTOF[n];}
```

Timer on delay dan *off delay* diuji dengan mengandaikan ada proses berikut. Sebuah *Output* Q1 dapat di-*on*-kan dengan tombol I1 dan dimatikan dengan tombol I0. *Output* lain Q2 dapat diaktifkan oleh Q1 dengan *on delay timer* berindeks 0 dengan *preset* 5 detik. *Output* Q3 diaktifkan oleh Q2 dengan *off delay timer* berindeks 3, dengan *preset* 10. Q4 diaktifkan dengan *off delay timer* 4 dengan Q3. Berikut adalah skrip ekspresi boolean untuk kasus yang dijelaskan.

```
//Skrip utama
Q1=(Q1||I1)&&!I0;
Q2=T(0,5,Q1);
Q3=TOF(3,10,Q2);
Q4=TOF(4,1,Q3);
```

Gambar 3.2 memperlihatkan grafik yang diperoleh dengan skrip yang dijelaskan di atas.



Gambar 3.2 Grafik perilaku *timer on* dan *off delay*.

Menurut data simulasi di atas, Q2 dalam kondisi *on* pada detik ke 4,581 setelah Q1 *on*. Q2 *on* pada detik ke 9,555 setelah Q1 *on*. Q2 *off* pada detik ke 10,971 dan Q3 *off* pada detik ke 20,925. Dari simulasi waktu *on delay* untuk Q2 dalah 4,974 sementara *preset* yang diberikan adalah 5. Waktu Q3 *off* dengan dengan *off delay* adalah 9,954 dengan *preset* dalam skrip sebesar 10. Maka nilai kesalahan waktu terhadap nilai *preset* yang diberikan untuk *on delay timer* sekitar 0,062 detik dan untuk *off delay timer* sekitar 0,046 detik.

Counter diberi nama C dengan 5 argumen. Argumen dari C masing- masing menyatakan indeks, *preset*, sinyal masukan yang dihitung bertambah, sinyal masukan yang dihitung berkurang, dan sinyal *reset*. Berikut adalah skrip program untuk mendefiniskan *timer on delay*, *off delay*, dan *counter*.

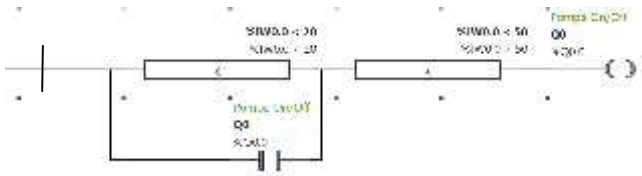
Skrip counter

```
function C(n, count, v,vd,Reset){
  var ret=false;
  if (Reset) {vlast[n] = false;
        vlastD[n] = false;
          Cs[n]=0;
          CS[n]=false;}
  else {
  if ( (v!=vlast[n]) && v) {
          Cs[n] = Cs[n] + 1
          Cv[n] = Cs[n]}
  if ( (vd!=vlastD[n]) && vd) {
          Cs[n] = Cs[n]-1
          Cv[n] = Cs[n]}
ret = false;
  if (Cs[n] == count)  { ret = true;}
  if (Cs[n] >= count)  { CS[n]=true;} else
{CS[n]=false;}
  vlast[n] = v
  vlastD[n] = vd
  if (Cs[n]>50*count) Cs[n]=count+1; }
return ret; }
```

3.3. Struktur Skrip Pemrograman

Skrip ekspresi boolean dan visualisasi atau animasi *plant* yang ditulis menggunakan program simulator ini tetap mengikuti sintaks Javascript. Struktur dari kode program diperlihatkan dan dijelaskan dengan contoh kontrol pengisian cairan dalam tangki sebagai berikut.

Permukaan ketinggian cairan dalam tangki sering kali dikontrol dengan prinsip ketika permukaan cairan pada tingkat ketinggian terendah pompa akan hidup untuk mengisi sampai pada ketinggian maksimum tertentu, kemudian pompa dimatikan kembali. Program dalam diagram *ladder* untuk kasus ketinggian terendah 20, pompa hidup, dan tinggi maksimum 50, pompa mati digambarkan sebagai berikut:



Gambar 3.3 Diagram *ladder* kontrol ketinggian permukaan cairan dalam tangki.

Skrip padanan *ladder* di atas dalam skrip ekspresi boolean dalam simulator ditulis dengan pernyataan Q0=((IW0<20)||Q0|)&&!(IW0>50). *Ladder* kontrol ketinggian permukaan cairan dalam tangki akan semakin kompleks jika melibatkan dua pompa yang bekerja bergantian dan sebagainya. Berikut ini adalah suatu skrip ekspresi boolean untuk kontrol ketinggian permukaan cairan dalam tangki dengan dua pompa bekerja

bergantian. Pompa kiri bekerja bila Q0 *on* dan Q1 *off*. Pompa sebelah kanan bekerja dibatasi dua kali bila Q0 *on* dan Q1 *on*. Selang waktu bekerja pompa kiri diset berdasarkan *preset timer* 0.

Skrip ekspresi boolean yang digunakan untuk mengatur I/O simulator ditulis sebelum komentar //Var. Skrip ini adalah skrip utama, dijalankan oleh program simulator untuk mengatur nilai *input* dan *output*. Skrip tambahan ditulis setelah //Var dan setelah //Sim. Skrip antara //Var dan //Sim merupakan inisialisasi variabel dan lain-lain seperti sumbu grafik.

```
// Skrip utama padanan ladder
Q0=((IW0<20)||Q0||I0)&&!(IW0>50);
M1=C(0,2,Q0,false,M0);
Q1=(M1||Q1)&&!M2
M2=C(1,3,Q0&&Q1,false,M0);
M3=(M2||M3)&&!M0
M0=T(0,10,M3);
//Sim
//bila nilai counter tercapai, pompa berganti
q=2*Q0;
IW0=h
//Simulasi plant
qo=h*0.03;
h=h+(q-qo);
```

```
//Var
// Skrip visualisasi grafis
h=0;M0=0;M1=false;M2=false;M3=false
tangki.tabung();
sumbu(0,30,0,2,5,2,'Waktu (detik)','Nilai Analog/True/False');

//Visualisasi Plant
tangki.level(h,Q1,Q0);
//Menampilkan grafik
plotQ(0,h/50,1,'blue','h','h:'+h.toFixed(2));
plotQ(1,Q0,2,'red','Q0','Q0');
plotQ(2,Q1,3,'black','Q1','Q1');
```

Fungsi tangki telah ditambahkan menjadi *internal library* dalam program yang dapat dipanggil dengan perintah tangki.tabung() dan tangki.level(h,Q1,Q0) dengan argumen terdiri dari nilai ketinggian (h), sinyal pemilihan pompa (Q1), dan kendali *on/off* pompa (Q0). Hasil dari skrip di atas diperlihatkan pada Gambar 3.1. Dalam simulasi nilai ketinggian permukaan cairan h pada grafik di normalisasi terhadap ketinggian maksimum 50.

Penggunaan simulator yang dibuat ini diharapkan dapat mempermudah dan menyederhanakan untuk mensimulasikan pemrograman perangkat I/O (seperti PLC dan mikrokontroler). Sebagaimana penulisan skrip padanan diagram *ladder* dengan ekspresi boolean pada simulator dan visualisasi hasil dapat lebih ringkas dan memperkecil ruang pandang dalam menguji jalannya program sehingga dapat diperoleh program lebih tepat.

4. KESIMPULAN

Penelitian ini menghasilkan sebuah simulator I/O dengan pemrograman skrip dalam bentuk ekspresi boolean padanan diagram *ladder* ditambah dengan tampilan grafik dan visualisasi plant yang dikontrol. Struktur skrip terdiri dari dua bagian yakni bagian utama yang merupakan skrip ekspresi boolean ditambah dengan *timer* dan *counter*. Bagian kedua adalah skrip grafik animasi atau visualisasi. Seluruh skrip ditulis dalam sintak pemrograman Javascript. Hasil penelitian ini diharapkan dapat dikembangkan dan diterapkan untuk memprogram perangkat keras I/O seperti mikrokontroler, mini PC dan PLC. Simulator ini dapat digunakan mensimulasikan diagram *ladder* dengan menuliskan ekspresi boolean padanannya sehingga lebih ringkas dan memperkecil ruang pandang dalam menguji jalannya program dengan tambahan visualisasi hasil. Algoritma *timer* dirancang diterapkan dalam skala waktu detik. Nilai kesalahan untuk *on delay* sekitar 0,062 detik dan untuk *off delay* sekitar 0,046 detik.

UCAPAN TERIMAKASIH

Penelitian ini dibiayai oleh Politeknik Negeri Bandung dengan Skema Dana Penelitian Mandiri. Pelaksanaan penelitian ini telah dilaksanakan dengan fasilitasi yang ada di Laboratorium Listrik dan Instrumen Jurusan Teknik Refrigerasi dan Tata Udara. Penulis mengucapkan terimakasih sebesar besarnya kepada Politeknik Negeri Bandung dan Jurusan Teknik Refrigerasi dan Tata Udara yang telah mendukung sehingga penelitian ini dapat diselesaikan dengan baik.

DAFTAR PUSTAKA

[1] W. Bolton. Programmable Logic Controllers. 4th Edition. Newnes. 2006: 455-456.

[2] Yan Yi, Zhang Hang Ping, *A New Translation Algorithm from Ladder Diagrams to Instruction Lists*. 17th World Congress The International Federation of Automatic Control. Seoul. 2008: 14804- 14809.

[3] Hongxia Xie and Zheng-Yun Zhuang. An Algorithm for Generating Boolean Expressions in VHDL Based on Ladder Diagrams, *Mathematical Problems in Engineering*. 2015. Volume 2015:1-10.
[4] E. Pruna, F. Bayas, H. Cocha, I. Escobar, A. Gordon, P. Constante. *Implementation of a simulator of industrial processes*, IEEE International Conference on Automatica (ICA-ACCA). Curico. 2016:pp. 607-6012.
[5] L. B. Palma, J. A. Rosas, J. Pecorelli, P. S. Gil. *Simulation of structured text language for PLC programming*. Experiment Conference. Ponta Delgada. 2015: 296-301.
[6] L. Brito Palma, V. Brito , J. Rosas , P. Gil . *WEB PLC simulator for ST programming*. 2017 4th Experiment@International Conference (exp.at'17). Faro. 2017: 303-308.
[7] Shobha S 1, S. Ramachandran. Realization of Timers, Counters and Shift Registers for Programmable Controller Using Ladder Diagram. *International Journal of Scientific & Engineering Research.* 2016; 7(4):137-148.
[8] Gerd Wagner. *Introduction to Simulation Using Javascript.* Winter Simulation Conference. Arlington. 2016:148-162.