BAB IV

ANALISIS SISTEM

4.1. ANALISIS PERANGKAT KERAS

4.1.1. Analisis Rangkaian Antarmuka

PPI 8255 yang digunakan sebagai antarmuka pada perancangan ini, terdiri dari 3 *port* 8 *bit I/O* sehingga total ada 24 bit I/O, sehingga 1 obyek memakai 1 bit I/O. Pemrograman pada *port A, Port B* dan *port C* sama, hanya perbedaan pada variabel *port*-nya saja.

Pertama kali dilakukan inisialisasi *base address* PPI 8255 dengan perintah :

```
BaseAddress = 640
PortA = BaseAddress
PortB = BaseAddress + 1
PortC = BaseAddress + 2
Cntrl = BaseAddress + 3
```

Jadi *base address* untuk *Port A* adalah 640d (280h), *Port B* adalah 641d (281h), *Port C* adalah 642d (282h) dan untuk *Control Word* adalah 643d (283h)

Sedang untuk menginisialiasi semua *port* menjadi *output* digunakan perintah :

```
Dummy = Out8255(Cntrl, 128)
```

Dan untuk membuat kondisi awal semua *port* 0 digunakan perintah :

```
Dummy = Out8255(PortA, 0)
Dummy = Out8255(PortB, 0)
Dummy = Out8255(PortC, 0)
```

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

Untuk menghidupkan atau mematikan obyek dengan memakai antarmuka PPI 8255 sangat sederhana. Ketika dikirim bit 1 (*high*) maka obyek akan *on*, sedangkan ketika dikirim bit 0 (*low*) maka obyek akan *off*. Pemprograman pada PPI dilakukan secara per-*port* artinya metode pengiriman dilakukan sekaligus 1 *byte* (8 *bit*), sehingga apabila ingin mengubah nilai suatu *bit* diperlukan teknik manipulasi bit dengan memakai operator OR dan AND.

Untuk menghidupkan obyek dapat digunakan operator OR dengan melakukan operasi : Q_N OR Y. Sedangkan untuk mematikan obyek dapat digunakan operator AND dengan melakukan operasi : Q_N AND Y. Q_N adalah kondisi *port* saat ini sedangkan Y adalah nilai pengubah bit. Adapun penjelasan tiap prosedur adalah sebagai berikut :

Prosedur menghidupkan A0

```
Dummy_A = Out8255(PortA, Dummy_A Or 1)
```

Artinya adalah : kondisi saat ini di-OR dengan1, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 0 yang mengalami perubahan.

X X X X X X X X

0 0 0 0 0 0 0 1 OR

X X X X X X X 1

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

Prosedur menghidupkan A1

```
Dummy_A = Out8255(PortA, Dummy_A Or 2)
```

Artinya adalah : kondisi saat ini di-OR dengan 2, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 1 yang mengalami perubahan.

X X X X X X X X

0 0 0 0 0 0 1 0 OR

X X X X X X 1 X

Prosedur menghidupkan A2

```
Dummy_A = Out8255(PortA, Dummy_A Or 4)
```

Artinya adalah : kondisi saat ini di-OR dengan 4, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 2 yang mengalami perubahan.

X X X X X X X X

0 0 0 0 0 1 0 0 OR

X X X X X 1 X X

Prosedur menghidupkan A3

```
Dummy_A = Out8255(PortA, Dummy_A Or 8)
```

Artinya adalah : kondisi saat ini di-OR dengan 8, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 3 yang mengalami perubahan.

X X X X X X X X

0 0 0 0 1 0 0 0 OR

X X X X 1 X X X

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

Prosedur menghidupkan A4

```
Dummy_A = Out8255(PortA, Dummy_A Or 16)
```

Artinya adalah : kondisi saat ini di-OR dengan 16, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 4 yang mengalami perubahan.

X X X X X X X X

0 0 0 1 0 0 0 0 OR

X X X 1 X 0 X X

Prosedur menghidupkan A5

```
Dummy_A = Out8255(PortA, Dummy_A Or 32)
```

Artinya adalah : kondisi saat ini di-OR dengan 32, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 5 yang mengalami perubahan.

X X X X X X X X

0 0 1 0 0 0 0 0 OR

X X 1 X X X X X

Prosedur menghidupkan A6

```
Dummy_A = Out8255(PortA, Dummy_A Or 64)
```

Artinya adalah : kondisi saat ini di-OR dengan 64, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 6 yang mengalami perubahan.

X X X X X X X X

0 0 1 0 0 0 0 0 OR

X X 1 X X X X X

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

Prosedur menghidupkan A7

```
Dummy_A = Out8255(PortA, Dummy_A Or 128)
```

Artinya adalah : kondisi saat ini di-OR dengan 128, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 7 yang mengalami perubahan.

X X X X X X X X

0 0 1 0 0 0 0 0 OR

X X 1 X X X X X

Prosedur mematikan A0

```
Dummy_A = Out8255(PortA, Dummy_A And 254)
```

Artinya adalah : kondisi saat ini di-AND dengan 254, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 0 yang mengalami perubahan.

X X X X X X X X

1 1 1 1 1 1 1 0 AND

X X X X X X X 0

Prosedur mematikan A1

```
Dummy_A = Out8255(PortA, Dummy_A And 253)
```

Artinya adalah : kondisi saat ini di-AND dengan 253, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 1 yang mengalami perubahan.

X X X X X X X X

1 1 1 1 1 1 0 1 AND

X X X X X X 0 X

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

Prosedur mematikan A2

```
Dummy_A = Out8255(PortA, Dummy_A And 251)
```

Artinya adalah : kondisi saat ini di-AND dengan 251, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 2 yang mengalami perubahan.

X X X X X X X X

1 1 1 1 1 0 1 1 AND

X X X X X 0 X X

Prosedur mematikan A3

```
Dummy_A = Out8255(PortA, Dummy_A And 247)
```

Artinya adalah : kondisi saat ini di-AND dengan 247, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 3 yang mengalami perubahan.

X X X X X X X X

1 1 1 1 0 1 1 1 AND

X X X X 0 X X X

Prosedur mematikan A4

```
Dummy_A = Out8255(PortA, Dummy_A And 239)
```

Artinya adalah : kondisi saat ini di-AND dengan 239, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 4 yang mengalami perubahan.

X X X X X X X X

1 1 1 0 1 1 1 1 AND

X X X 0 X 1 X X

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

Prosedur mematikan A5

```
Dummy_A = Out8255(PortA, Dummy_A And 223)
```

Artinya adalah : kondisi saat ini di-AND dengan 223, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 5 yang mengalami perubahan.

X X X X X X X X

1 1 0 1 1 1 1 1 AND

X X 0 X X X X X

Prosedur mematikan A6

```
Dummy_A = Out8255(PortA, Dummy_A And 191)
```

Artinya adalah : kondisi saat ini di-AND dengan 191, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 6 yang mengalami perubahan.

X X X X X X X X

1 0 1 1 1 1 1 1 AND

X 0 X X X 1 X X

Prosedur mematikan A7

```
Dummy_A = Out8255(PortA, Dummy_A And 127)
```

Artinya adalah : kondisi saat ini di-AND dengan 127, sehingga kondisi ke 7 *bit* lainnya tidak mengalami perubahan, hanya bit 7 yang mengalami perubahan.

X X X X X X X X

0 1 1 1 1 1 1 1 AND

0 X X X X X X X

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

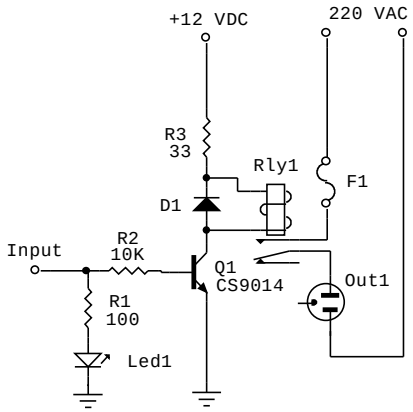
4.1.2. Analisis Rangkaian Buffer

Disini akan diperbandingkan antara hasil perhitungan dengan pengukuran kondisi *cutoff* dan *saturation* pada transistor yang dipergunakan pada sistem :

Tabel 4.1. Perbandingan Perhitungan dan Pengukuran Transistor

Perhitungan	Pengukuran
$V_{CE(cutoff)}$ = 12 V $I_{C(sat)}$ = 51,50 mA $I_{B(sat)}$ = 858 µA $V_{CE\ (sat)} \cong 0$ V_{BE} = 0,7 V	$V_{CE(cutoff)}$ = 12 V $I_{C(sat)}$ = 45 mA $I_{B(sat)}$ = 800 µA $V_{CE\ (sat)}$ = 0,07 V V_{BE} = 0,70 V

Dari hasil di atas terlihat, antara perhitungan dengan pengukuran tidak terlihat adanya perbedaan yang menonjol, sehingga bisa dikatakan transistor sebagai satu-satunya komponen aktif pada rangkaian *dP-Buffer* bekerja dengan kondisi normal. Rangkaian bias basis yang dirancang adalah sebagai berikut :



Gambar 4.1. Rangkaian bias basis *dP-Buffer*

Ketika tansistor dalam kondisi *cut off*, nilai V_{CE} sebesar V_{CC} sehingga relay mendapat tegangan 12 V pada kedua pin catu dayanya dan akibatnya relay tidak bekerja. Ketika transistor dalam kondisi *saturation*, V_{CE} hampir sama dengan 0 sehingga relay mendapat tegangan 0 V dan 12 V pada kedua pin catu dayanya dan akibatnya relay bekerja. Hasil dari kerja relay inilah yang meneruskan tegangan

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

220 V pada obyek yang dikendalikan. Seperti yang bisa dilihat pada rangkaian, obyek membutuhkan tegangan 220 V AC. *Netral*-nya di *by-Pass* sedangkan *phase*-nya menunggu dari relay aktif. Dengan kata lain, begitu transistor dalam kondisi *saturation*, relay akan aktif dan obyek bisa bekerja. Dan ketika transistor dalam kondisi *cutoff*, relay tidak aktif dan obyek tidak bekerja. Penentuan *saturation* dan *cutoff* transistor tergantung dari keluaran PPI 8255 yang dirancang oleh dikendalikan oleh komputer yang terhubung ke jaringan. Led1 berfungsi sebagai indikator input dan diseri dengan R1 sebagai tahanan muka Led. Fungsi D1 sebagai pembatas arus pada transistor, sehingga arus dari V_{CC} maksimal hanya sebesar arus yang melalui D1. Sedang fungsi F1 sebagai pengaman untuk obyek yang dikendalikan apabila terjadi hubungan singkat.

4.2. ANALISIS PERANGKAT LUNAK

Analisis perangkat lunak disini adalah pembahasan tentang unjuk kerja sistem yang telah dibuat difokuskan pada perangkat lunak. Digunakan pendekatan kasus sehingga bisa lebih memperjelas dalam pemahaman alur kerja dan unjuk kerja sistem serta *bug* pada sistem karena walaupun perencanaan sistem sudah direncanakan secara matang dan mendalam, masih ditemui kendala-kendala yang mengakibatkan pengendalian sistem tidak berjalan sebagaimana yang diharapkan. Adapun kondisi-kondisi yang terjadi di sini adalah semua hal yang ditemui selama melakukan pengujian sistem.

- Kondisi awal sistem

  Saat pertama kali *User* ingin mengakses obyek, *User* harus menjalankan program *dP-Client* pada komputer *client.* Setelah program aktif, *User* harus

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

mengisi LOGIN-ID dan PASSWORD-ID sehingga admin sistem bisa mengidentifikasi *User*. Penggunaan LOGIN dan PASSWORD jelas akan sangat berguna untuk keamanan sistem. Sehingga siapapun yang ingin mengakses obyek harus mempunyai LOGIN-ID dan PASSWORD-ID.

- Kondisi saat koneksi sedang berlangsung

  Saat koneksi sedang berlangsung, baik operator sisi lokal atau *User* maupun operator sisi jauh atau *admin*, bisa mengetahui kondisi obyek yang sedang diakses dengan cara melihat *frame* pada program. *Frame* hijau berarti obyek ON dan *frame* merah berarti obyek OFF. Dan kedua operator bisa berkomunikasi dengan cara menulis pesan / *chatting* pada layar pesan.

- Kondisi ketika ada dua *User* yang ingin mengakses obyek

  Ketika obyek sedang diakses oleh *User A*, kemudian *User B* menggunakan komputer *Client B* mencoba berkoneksi dengan komputer server, maka ketika ia menjalankan program *dP-Client* pada komputer *Client B*, program akan menampilkan *error message* karena komputer *server* sedang terkoneksi dengan komputer *Client A.* ketika *User A* memutuskan koneksi dengan *server*, baru *User B* berkoneksi dengan *server*, sehingga tidak akan terjadi bentrok akses antara *User A* dengan *User B.*

- Kondisi saat *Admin* memutuskan koneksi *User*

  Operator sisi pengendali jauh atau *Admin* mempunyai *authority* untuk memutuskan koneksi dari operator sisi lokal atau *User*. *Admin* cukup menekan tombol *Close* pada program *dP-Server* pada komputer *server,* sehingga koneksi dari *User* terputus. *Admin* bisa mangakses obyek setelah terlebih dahulu mengisi ADMIN-PASSWORD pada *dP-Server*. Selama tombol *Close*

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

aktif, semua koneksi dari komputer *client* manapun akan gagal, sampai tombol *Open* aktif. Ketika tombol *Open* aktif, maka semua tombol kendali obyek pada *dP-Server* non aktif sehingga *Admin* tidak bisa mengakses obyek dan di saat itu *user* manapun bisa mengakses obyek, sehingga tidak akan terjadi bentrokan akses antara *Admin* dengan *User*.

- Kondisi saat obyek diakses

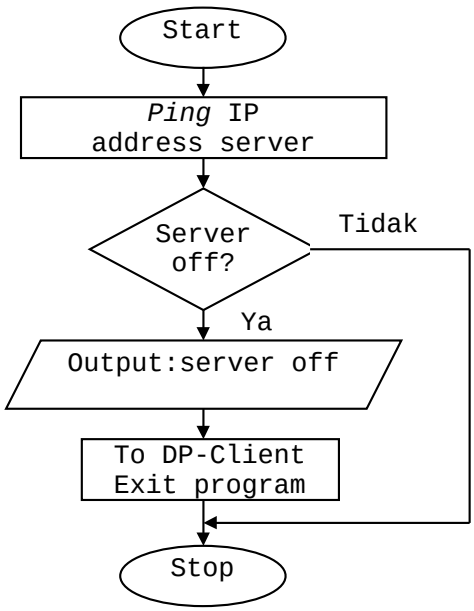
  Saat obyek diakses, baik oleh *User* maupun *Admin*, maka *frame* obyek berwarna hijau yang menandakan obyek ON atau merah yang menandakan obyek OFF. Pada layar *chatting* juga tertulis obyek yang diakses disertai dengan waktu pengaksesan obyek, sehingga baik *User* maupun *Admin* bisa melihat kondisi obyek saat ini dan waktu kapan obyek itu diakses.

- Kondisi saat komputer *server* sedang dalam keadaan mati (*OFF*).

  Saat komputer *server* dalam posisi mati, prosedur *connection error* baru akan tampil setelah 45 detik sejak program *dP-Client* dijalankan. Artinya operator *client* harus menunggu selama 45 detik untuk menunggu hasil koneksi yang pasti gagal sebab kala itu komputer *server* dalam posisi mati. 45 detik adalah hasil pengujian dengan memakai 6 komputer yang terhubung ke jaringan sehingga apabila jumlah komputer melebihi itu maka akan dibutuhkan waktu lebih dari 45 detik. Sebab pada kasus ini, protokol ARP (*Address Resolution Protokol*) yang berfungsi menerjemahkan *IP Address* ke alamat ethernet harus mencari nomor *IP Address* komputer *server* ke seluruh jaringan, sehingga makin luas jaringannya, maka waktu yang dibutuhkannya juga makin lama. Hal ini bisa diatasi dengan cara membuat *ping shortcut* pada program *dP-Client* yang berfungsi untuk memeriksa status komputer *server* yang secara

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

otomatis berjalan sesaat *IP address* komputer *server* dimasukkan pada layar IP Address tujuan. Dengan adanya *ping shortcut* ini, maka apabila saat itu komputer *server* sedang dalam kondisi mati, maka bisa ditampilkan display server off. Adapun `prosedur server_off` dapat dibuat dengan algoritma sebagai berikut :

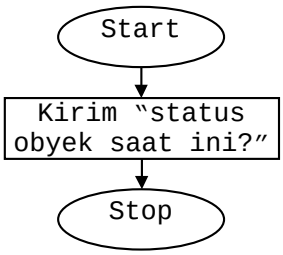


Gambar 4.2. Algoritma Prosedur Server Off

- Kondisi status obyek saat ini

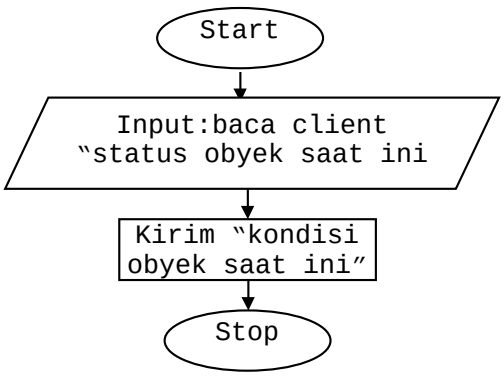
  Ketika program *dP-Client* dijalankan pertama kali, warna *frame* tidak menunjukkan status obyek (warna hijau untuk obyek pada kondisi *ON* warna merah untuk obyek pada kondisi *OFF*). Warna *frame* sama dengan warna *background* sehingga operator *client* tidak bisa mengidentifikasi kondisi obyek kala itu. Hal ini sebenarnya bisa diatasi dengan cara penambahan `prosedur kirim_kalibrasi` pada program *dP-Client* dengan algoritma sebagai berikut :

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com



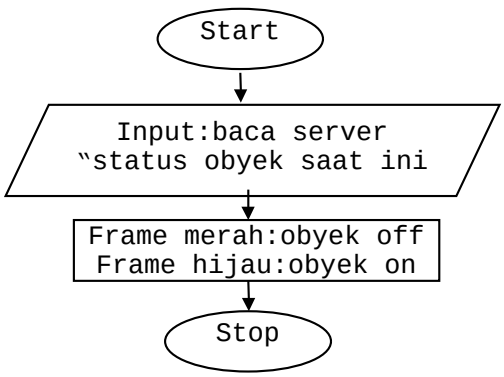
Gambar 4.3. Algoritma Prosedur Kirim Kalibrasi

Kemudian pada program *dP-Client* juga ditambahkan `prosedur kalibrasi` dengan algortima sebagai berikut :



Gambar 4.4. Algoritma Prosedur Kalibrasi

Terakhir, dibuat lagi `prosedur terima_kalibrasi` pada program *dP-Client* yang algoritmanya dapat dibuat sebagai berikut :



Gambar 4.5. Algoritma Prosedur Terima Kalibrasi

Ketika algoritma tadi disesuaikan dengan sistem yang telah dibuat, maka diperlukan *case* dari 00000000 (semua obyek mati) hingga 11111111 (semua

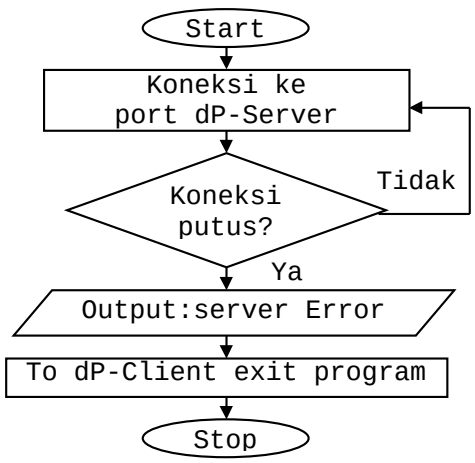
PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

obyek hidup), artinya akan ada 255 *case* hanya untuk satu *port*, sedangkan PPI 8255 mempunyai 3 *port* sehingga total dibutuhkan 765 *case*!

Masih ada cara lain yang lebih sederhana dibandingkan dengan memakai solusi di atas yaitu dengan memakai konsep *database*. Semua kondisi didefinisikan pada program *dP-Client* serta *dP-server*, sehingga *programmer* tidak lagi dipusingkan dengan 765 *case* kalau memakai solusi sebelumnya.

- Koneksi terputus sesudah prosedur *connection success* tampil

  Kondisi ini bisa terjadi disebabkan karena beberapa hal contohnya kabel jaringan terputus, *consentrator* tidak berfungsi sebagaimana mestinya, *rutin* jaringan pada sistem operasi *error* dan lain sebagainya. Hal ini bisa diatasi dengan cara menambah `prosedur lost_connection` pada program *dP-Client* yang mana prosedur ini akan terus hidup ketika port *dP-Client* dan port *dP-server* terkoneksi dan prosedur ini akan tampil ketika port *dP-Client* dan port *dP-server* tidak terkoneksi atau koneksi terputus. Adapun algoritmanya bisa dibuat sebagai berikut :

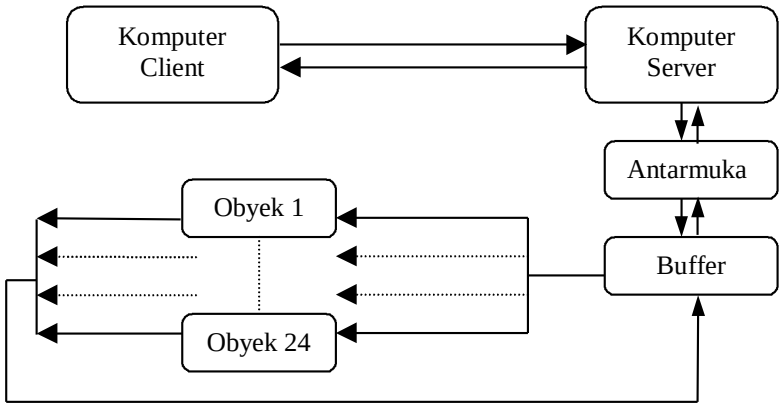


Gambar 4.6. Algoritma Prosedur lost connection

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com

- Kondisi obyek tidak berjalan sebagaimana yang diinginkan

  Karena sistem hanya dirancang secara satu arah, maka apabila obyek tidak berjalan sebagaimana yang diinginkan, tidak ada prosedur error yang tampil. Hal ini bisa diatasi dengan membuat sistem secara dua arah (*close loop*) hanya karena batas masalah pada Tugas Akhir ini hanya pada sistem satu arah maka perancangan tidak sampai pada tahap dua arah. Untuk blok sistem dua arah bisa dibuat sebagai berikut :

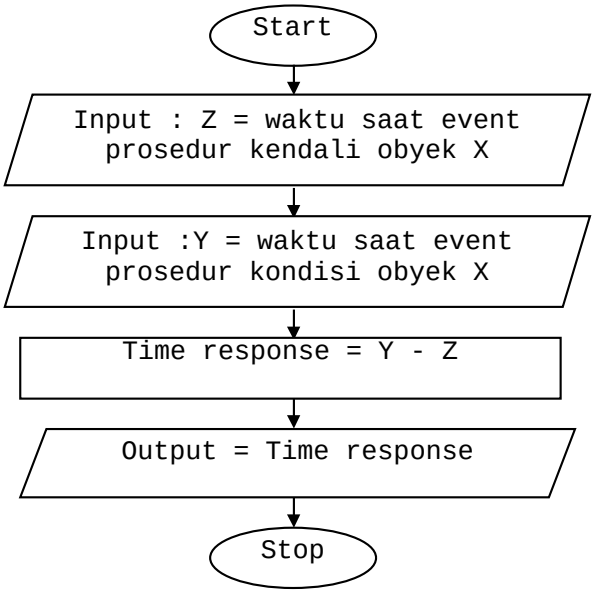


Gambar 4.7. Blok sistem dua arah

- Kondisi *time response system*

  Pada perancangan sistem ini, *time response system* atau waktu tanggap sistem tidak bisa terdeteksi karena sistem hanya bekerja secara satu arah. Apabila sistem bekerja secara dua arah maka bisa diketahui *time respons sistem.* Untuk algoritma `prosedur time_response` bisa dibuat sebagai berikut pada program *dP-Client* :

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com



Gambar 4.8. Algoritma Prosedur time response

PDF created with FinePrint pdfFactory trial version http://www.softwarelabs.com