PERANCANGAN DAN SIMULASI PERANGKAT LUNAK SISTEM KONTROL LIFT

Oleh:
Arie S. M. Lumenta.
Jurusan Teknik Elektro-FT UNSRAT, Manado-95115,

Abstrak

Pada perangkat sistem lift, perangkat lunak memegang peranan yang penting terhadap pengoperasian lift tersebut. Ketidaktepatan kerja lift terutama disebabkan oleh adanya kesalahan perangkat lunak yang terdapat pada perangkat kerasnya. Perangkat lunak tersebut berfungsi sebagai kontrol penentu keputusan termasuk didalamnya kontrol motor penggerak sistem lift. Untuk itu dalam pembuatan perangkat lunak kontrol lift, perancangan perangkat lunak menjadi hal yang sangat penting untuk mencegah terjadinya kesalahan pemrogrmanan. Setelah itu untuk memastikan bahwa hasil rancangan sudah benar, perlu juga disimulasikan terlebih dahulu. Kedua hal tersebut dilakukan sebelum perangkat lunak yang dibuat di-embedded-kan pada sistem lift
***Kata kunci**: Lift, finite state machine, sistem kontrol lift, visual basic*

1. Latar Belakang

Lift merupakan alat transportasi yang banyak digunakan pada gedung-gedung bertingkat. Sistem kontrol pada lift sangat kompleks pada alat yang sebenarnya. Mulai dari masalah kontrol penentuan keputusan, kontrol motor, kontrol tampilan, kontrol pintu sampai masalah keselamatan. Untuk itu perlu dilakukan pembagian, modul kerja untuk merealisasikan sistem lift tersebut. Dalam tulisan akan dilakukan perancangan software masalah penentu keputusan layanan lift yang disimulasikan dengan program yang dirancang menggunakan Visual Basic 6.0 serta diimplementasikan pada software Keil versi 2.0.

2. Permasalahan

Permasalahan yang timbul dalam perancangan software penentu keputusan pada lift adalah :

- bagaimana menerapkan *Finite state machines (FSM)* pada program penentu keputusan lift..
- bagaimana melakukan *debugging* pada software untuk permasalahan di atas.

3. Spesifikasi

Spesifikasi pengendali Lift yang dirancang adalah sebagai berikut :

Spesifikasi lift

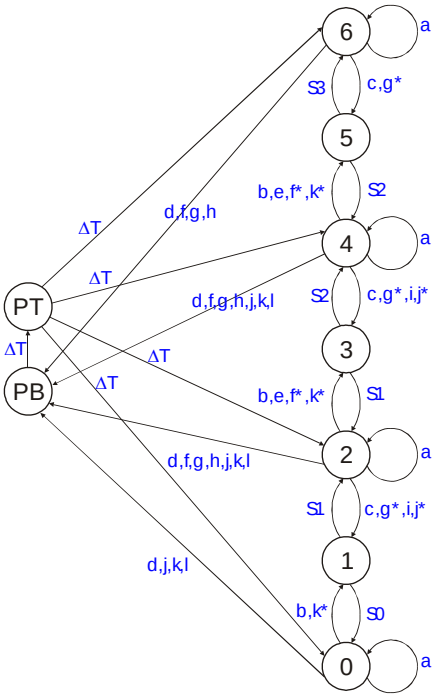
- 4 lantai (G : gorund ,1,2,3)
- Pada tiap lantai ada 1 atau 2 tombol untuk memanggil lift
- Pada lantai G, hanya ada 1 tombol pemanggil naik
- Pada lantai 3, hanya ada 1 tombol pemanggil naik
- Pada lantai 1 dan 2, ada 2 tombol pemanggil turun dan naik
- Di dalam lift ada 5 tombol:
  - G (PKG : pergi ke lantai ground)
  - 1 (PK1 : pergi ke lantai 1)
  - 2 (PK2 : pergi ke lantai 2)
  - 3 (PK3 : pergi ke lantai 3)
  - Alarm
- Ada 4 buah sensor untuk mengetahui posisi lift
  - SG : sensor ground : aktif jika lift ada persis di lantai G
  - S1 : sensor pada lantai 1
  - S2 : sensor pada lantai 2
  - S3 : sensor pada lantai 3

Spesifikasi rangkaian penggerak lift
Input rangkaian penggerak lift :
- MU (motor naik) : motor menaikkan lift
- MT (motor turun) : motor menaikkan lift
- PT (pintu tutup) : menutupkan pintu lift
- PB (pintu buka) : membuka pintu lift

Output rangkian driver lift :
- L3T : Lantai 3 turun
- L2N : Lantai 2 naik
- L2T : Lantai 2 turun
- L1N : Lantai 1 naik
- L1T : Lantai 1 turun
- LGN : lantai G naik
- PKG : tombol dalam lift untuk pergi ke lantai G
- PK1 : tombol dalam lift untuk pergi ke lantai 1
- PK2 : tombol dalam lift untuk pergi ke lantai 2
- PK3 : tombol dalam lift untuk pergi ke lantai 3
- Alarm : tombol dalam lift (alarm)
- SG : sensor lantai ground
- S1 : sensor lantai 1
- S2 : sensor lantai 2
- S3 : sensor lantai 3
- GND : Signal Ground

4. Tujuan
Tujuan yang ingin dicapai adalah merancang finite state machines pada program penentu keputusan sistem lift dan mensimulasikannnya.

5. FSM (Finite State Machine) dan Tabel Logika
Pengontrol lift ini dapat dipresentasikan dalam bentuk FSM dengan 9 state seperti pada gambar 1.



Gambar 1. FSM Pengendali Lift

Kesembilan state tersebut yaitu:
1. State 0 : Tabung lift berada di lantai ground
2. State 1 : Tabung lift berada di antara lantai ground dan laintai 1
3. State 2 : Tabung lift berada di lantai 1
4. State 3 : Tabung lift berada di antara lantai 1 dan laintai 2
5. State 4 : Tabung lift berada di lantai 2
6. State 5 : Tabung lift berada di antara lantai 2 dan laintai 3
7. State 6 : Tabung lift berada di lantai 3
8. State PB : Pintu lift membuka
9. State PT : Pintu lift menutup

Selanjutnya kesembilan state di atas dapat dikelompokan menjadi 3 kelompok yaitu :

1. Kelompok state yang menunjukan tabung berada di lantai tertentu (State 0, 2, 4, dan 6).
2. Kelompok state yang menunjukan tabung berada di antara 2 lantai (State 1, 3, dan 5).
3. Kelompok state yang menunjukan keadaan pintu (State PB dan PT).

Setiap kelompok mempunyai fungsi pengontrolan yang berbeda sebagai berikut :
Setiap state pada kelompok 2 mempunyai 2 input dan 2 output. Hanya ada dua kemungkinan perpindahan state yaitu dari bawah ke atas atau dari atas ke bawah. Output diaktivasi oleh sensor keberadaan tabung yaitu SG bila tabung berada di lantai ground, S1 bila tabung berada di lantai 1 dan seterusnya. Pada keompok state ini bisa juga ditambahkan alarm, yaitu bila tabung lift terlalu lama berada pada kelompok ini maka alarm akan berbunyi.
Kelompok state yang menunjukan keadaan pintu mengalir hanya satu arah yaitu dari state yang lain pindah ke PB (membuka pintu) lalu setelah waktu tertentu akan pindah ke PT (menutup pintu) dan kemudian kembali ke asal dari mana state itu datang.
Kelompok state yang menunjukan posisi tabung berada di lantai tertentu ialah yang paling rumit. Pertama tinjau masalah input, ia bisa berasal dari state diatas/dibawahnya, bisa dari state itu sendiri (looping) atau berasal dari state PT. Kemudian masalah output, ia bisa ke state di atas atau di bawahnya, ke state itu sendiri atau ke PB.
Dari input dan output kelompok state nomor 2 di atas dapat dibuat logika beberapa kemungkinan yang bisa terjadi pada tabung dalam tabel 1 sebagai berikut :

Tabel 1. Tabel Logika state

	Sebelum		Keadaan Sekarang	Keadaan Nanti
	Posisi	Keadaan		
a	Di sini	Diam	Diam	Diam
b	Di sini	Diam	Diam	Naik
c	Di sini	Diam	Diam	Turun
d	Di sini	Diam	Stop	Diam
e	Di bawah	Naik	Lewat	Naik
f	Di bawah	Naik	Stop	Naik
g	Di bawah	Naik	Stop	Turun
h	Di bawah	Naik	Stop	Diam
i	Di atas	Turun	Lewat	Turun
j	Di atas	Turun	Stop	Turun
k	Di atas	Turun	Stop	Naik
l	Di atas	Turun	Stop	Diam

Disini dibedakan antara Diam dan Stop. Yang dimaksud Diam ialah tidak melakukan kegiatan (dalam FSM = looping), sedangkan Stop ialah pindah ke state PB kemudian PT dan kembali lagi atau membuka dan menutup pintu.
Untuk memudahkan pencarian logika dan pemindahan ke software tabel diatas dapat dibuat dalam logika 0 dan 1 seperti pada tabel 2 berikut :

Tabel 2. Tabel logika penjabaran state

Kom-binasi	Sebelum				Skr	Nanti	
	A	B	N	T	S	N	T
a	0	0	0	0	0	0	0
b	0	0	0	0	0	1	0
c	0	0	0	0	0	0	1
d	0	0	0	0	1	0	0
e	0	1	1	0	0	1	0
f	0	1	1	0	1	1	0
g	0	1	1	0	1	0	1
h	0	1	1	0	1	0	0
i	1	0	0	1	0	0	1
j	1	0	0	1	1	0	1
k	1	0	0	1	1	1	0
l	1	0	0	1	1	0	0

Keterangan :
Pada posisi Sebelum: A dan B menunjukan :
A : tabung berada di atas
B : tabung berada di bawah
Sedangkan N dan T menunjukan :
N : tabung dalam keadaan naik
T : tabung dalam keadaan turun
Pada posisi sekarang S menunjukan
1 : Stop (pintu dibuka kemudian ditutup)
0 : Diam
Pada posisi nanti N dan T menunjukan naik dan turun (seperti pada posisi sebelum).
Cara lain dalam pencarian logika ialah menuliskan semua logika kombinasi dari ke-tujuh input (2^7 = 128 kombinasi) kemudian menghilangkan semua logika yang tidak mungkin. Hasil yang didapat akan sama seperti di atas. Pencarian logika seperti ini dapat dilihat di lampiran A.
Tahap berikutnya dari perancangan ialah mencari kemungkinan keadaan tombol dari setiap kombinasi yang telah dihasilkan di atas. Karena kelompok state nomor 1 dan nomor 3 sudah cukup mudah untuk langsung diterjemahkan ke software, maka disini akan dibuat logika untuk kelompok state nomor 2.
Pada keadaan tabung tepat berada di lantai tertentu yang perlu diperhatikan ialah tombol PK(X), LN(X), dan LT(X) → X=current stage. Untuk tombol yang lainya bila diperiksa satu per satu akan terlalu banyak sehingga perlu dibuat suatu variabel Br(X) yang menunjukan permintaan untuk berhenti pada lantai X. Dari variabel ini bisa dibuat variabel lain yaitu A(X) yang menunjukan adanya permintaan berhenti dari lantai di atas X dan B(X) yang menunjukan adanya permintaan behenti dari lantai di bawah X.
Pada lift sebenarnya sebelum berhenti diperlukan waktu tertentu untuk mengerem secara perlahan dan dihindari dari berhenti mendadak. Untuk itulah diperlukan variabel lain yaitu T(X) yang menunjukan lift terlambat untuk berhenti di lantai X. T(X) ini aktif bila ada permintaan untuk berhenti pada laintai X padahal tabung lift berada di state current state - 1 dan sedang naik atau berada di current state +1 dan sedang turun.

Dari logika di atas dapat dibuat tabel antara tombol-tombol dan variabel-variabel yang merupakan fungsi X dengan kombinasi yang ditemukan dari tabel sebelumnya.

Tabel 3. Logika kombinasi

Kom-binasi	Br(X)	T(X)	PK(X)	LN(X)	LT(X)	A(X)	B(X)
a	0	0	0	0	0	0	0
b	0	0	0	0	0	1	0
c	0	0	0	0	0	0	1
d	1	0	1	d	d	0	0
d	1	0	d	1	d	0	0
d	1	0	d	d	1	0	0
e	d	1	d	d	d	1	d
e	0	0	0	0	0	1	d
e	1	0	0	0	1	1	d
f	1	0	d	1	d	1	d
f	1	0	1	d	d	1	d
g	1	0	1	0	d	0	1
h	1	0	1	d	d	0	0
h	1	0	0	1	d	0	0
h	1	0	0	d	1	0	0
i	0	0	0	0	0	d	1
i	1	0	0	1	0	d	1
i	d	1	d	d	d	d	1
j	1	0	d	d	1	d	1
j	1	0	1	d	d	d	1
k	1	0	1	d	d	1	0
l	1	0	1	d	d	0	0
l	1	0	0	1	d	0	0
l	1	0	0	d	1	0	0

Tabel diatas menunjukan semua kombinasi yang mungkin dari ke-tujuh input (tombol dan variabel) dan setiap kombinasi input bila tidak tereliminasi harus cocok dengan salah satu dari kombinasi tabel sebelumnya (a s/.d. l) dengan syarat pada satu kelompok kombinasi tidak boleh ada dua yang cocok. Kelompok kombinasi yang dimaksud ialah kelompok kombinasi Disini, kelompok kombinasi Di bawah, dan kelompok kombinasi di atas.
Bila telah sampai tahap ini maka tahap selanjutnya ialah menerjemahkan logika yang telah ditemukan ke dalam software.

Logika pengendalian lift yang telah ditemukan sebelum diimplementasikan ke dalam software yang siap untuk ditulis ke mikrokontroler (software yang hardware dependent) terlebih dahulu dibuat versi yang hardware independent. Hal ini dilakukan untuk melakukan simulasi dan mempermudah debugging. Debugging akan lebih mudah bila software yang digunakan sudah menggunakan GUI (Graphic User Interface) sehingga gerakan lift akan terlihat jelas salah atau benarnya.

Akan sangat mudah bila software yang hardware independent menggunakan bahasa yang sama dengan software yang nantinya akan dituliskan ke mikrokontroler (software yang hardware dependent). Karena program yang akan digunakan untuk menulis software yang hardware dependent tersebut ialah Keil yang menggunakan bahasa C maka sebaiknya software yang hardware independent juga ditulis dalam bahasa C. Berhubung dengan keterbatasan fasilitas maka pada saat ini penulisan software yang hardware independent menggunakan Visual Basic (VB). Meskipun demikian diusahakan agar peralihan program tidak terlalu sulit misalnya meskipun ada type Boolean (TRUE or FALSE) pada variabel VB tapi pada software ini tidak digunakan. Untuk software ini pemeriksaan logika menggunakan byte / integer seperti pada bahasa C yaitu 0 untuk FALSE dan selain 0 untuk TRUE.

6. Diagram Alir

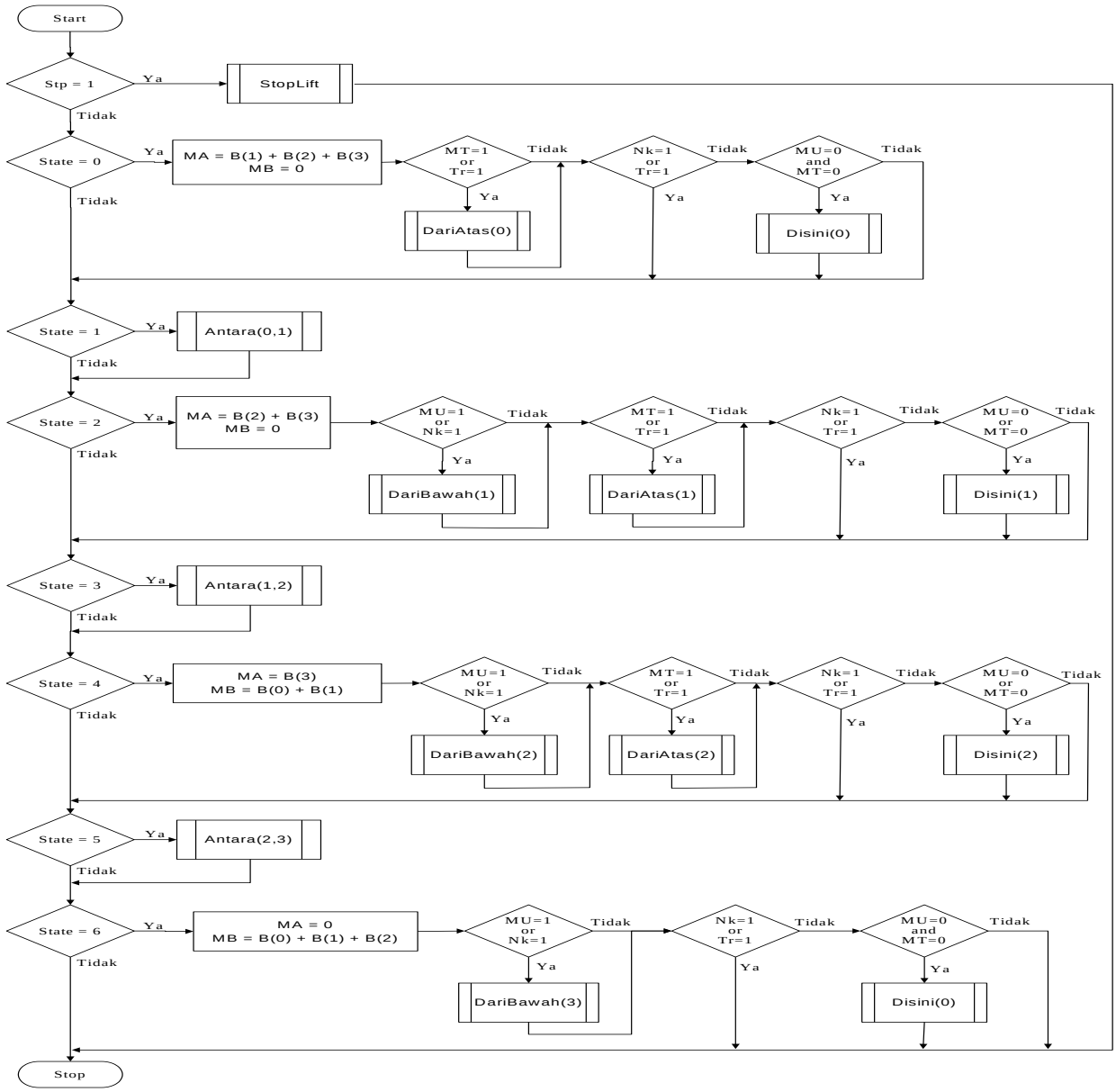
Program ini terdiri atas beberapa subroutine, yaitu :

- Subroutine inti yang mengarahkan pergerakan dari satu state ke state lainnya : KendaliLift().
- Subroutine yang mengatur apa yang harus dilakukan pada suatu state : Disini(), Antara(), DariBawah(), Dari Atas(), StopLift().
- Subroutine yang membantu pengendalian lift : Naik(), Turun(), StopReq().
- Subroutine pembacaan tombol: BacaTombol()

KendaliLift()

Flowchart untuk KendaliLift() bisa dilihat pada Gambar 2. Begitu masuk ke subroutin ini terdapat pilihan apakan Stp=1 atau bukan. Variabel Stp ialah variabel yang menyatakan bahwa tabung dalam keadaan stop (membuka dan menutup pintu) sehingga bila Stp=1 maka akan langsung ke StopLift() tanpa memeriksa tabung berada di state mana karena di state manapun proses stop tetap sama.

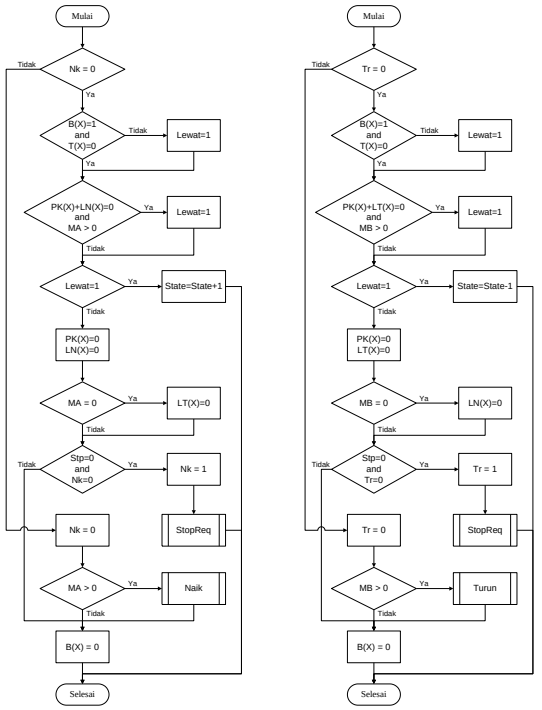
Bila tidak dalam proses stop (Stp=0) maka akan diperiksa posisi current state. Terdapat dua kelompok proses yaitu pada state ganjil (tabung berada di antara dua lantai) dan pada state genap (tabung berada di lantai tertentu). Pada state ganjil aliran proses akan dialihkan ke Antara() dengan masukan lantai dibawah tabung dan lantai di atas tabung. Pada state genap diperiksa variabel MU dan MT kecuali pada lantai terbawah (state 0) MU tidak diperiksa dan pada lantai teratas (state=6) MT tidak diperiksa. Bila MU=1 (TRUE) yang berarti motor sedang menggerakkan tabung naik maka subroutine DariBawah() dipanggil. Bila MT=1 (TRUE) yang berarti motor sedang menggerakkan tabung turun maka subroutine DariAtas() dipanggil. Bila MU=0 dan MT=0 yang berarti tabung dalam keadaan Diam maka subroutine Disini() dipanggil. Variabel Nk dan Tr digunakan untuk memisahkan posisi Diam dan Stop karena pada kedua posisi ini MU dan MT = 0 tapi proses yang dilakukan berbeda.



Gambar 2. Flowchart KendaliLift()

DariBawah() & DariAtas()

Flowchart untuk DariAtas() dan DariBawah() bentuknya sama (Gambar 3) hanya variabel-variabelnya yang berbeda oleh karena itu keterangan untuk kedua folwchart tersebut disatukan. Pertama yang diperiksa ialah Nk/Tr bila 1 maka Nk/Tr berarti state sebelumya ialah PT atau sebelumnya ialah proses Stop, yang harus dilakukan kemudian ialah mereset Nk/Tr dan B(X) dan memeriksa apakah ada permintaan berhenti dari atas/bawah kalau ada berarti dilanjutkan dengan Naik()/Turun() kalau tidak berarti selanjutnya ialah proses Diam

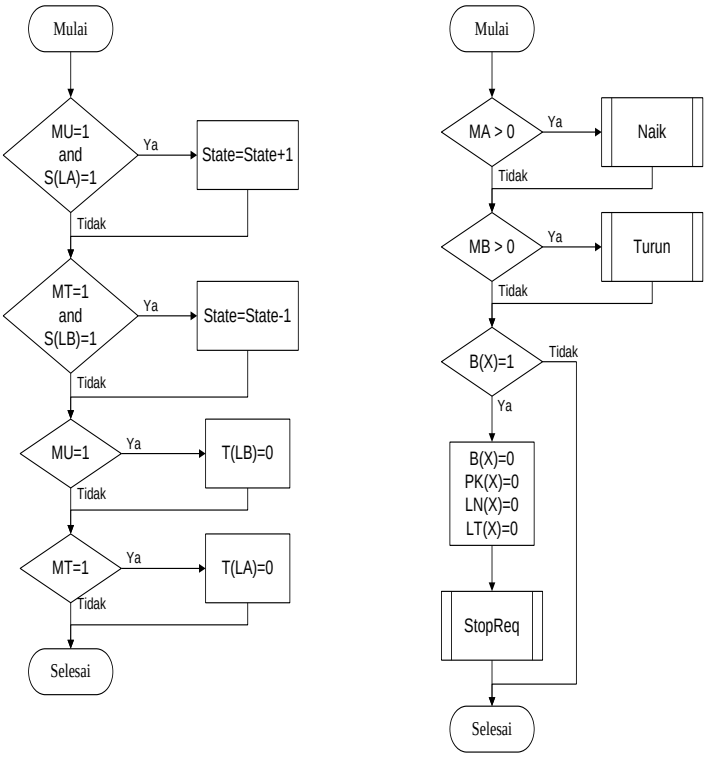


Gambar 3. Flowchart DariBawah() dan DariAtas()

Bila Nk/Tr = 0 berarti tabung berasal dari bawah/atas dan yang pertama diperiksa ialah apakah ia lewat atau tidak. Ada dua syarat yang diperiksa, yang pertama ialah apakah ada permintaan berhenti dari lantai tersebut dan permintaan tersebut tidak terlambat. Bila syarat yang pertama tidak terpenuhi maka tabung lewat tapi bila terpenuhi masih ada syarat ke dua yaitu bila permintaan tersebut bukan dari arah yang sama (LN(X) untuk naik dan LT(X) untuk turun) dan ada permintaan berhenti di atas/bawah maka tabung akan tetap lewat.
Bila tabung berhenti di state tersebut maka semua yang menyebabkan berhenti harus direset (PK, LN/LT, dan B). Setiap berhenti harus diawali oleh proses Stop jadi nilai Stp dan Nk/Tr harus disesuaikan.

Antara()
Pada subroutine Antara() (Gambar 4) ada dua hal yang dilakukan. Pertama menunggu sinyal S; bila ada sinyal S dari lantai di atasnya maka current state pindah ke state di atasnya, sebaliknya bila ada sinyal S dari lantai di bawahnya maka current ctate pindah ke state di bawahnya. Hal yang ke dua ialah mereset T (Terlambat) bila sedang naik maka T dari lantai di bawahnya direset dan bila sedang turun maka T dari lantai di atasnya direset.
Ada hal ketiga yang bisa ditambahkan di subroutine ini yaitu pengecekan apakah lift macet dengan cara menghitung waktu aktifnya state ini. Bila terlalu lama maka alarm dibunyikan.

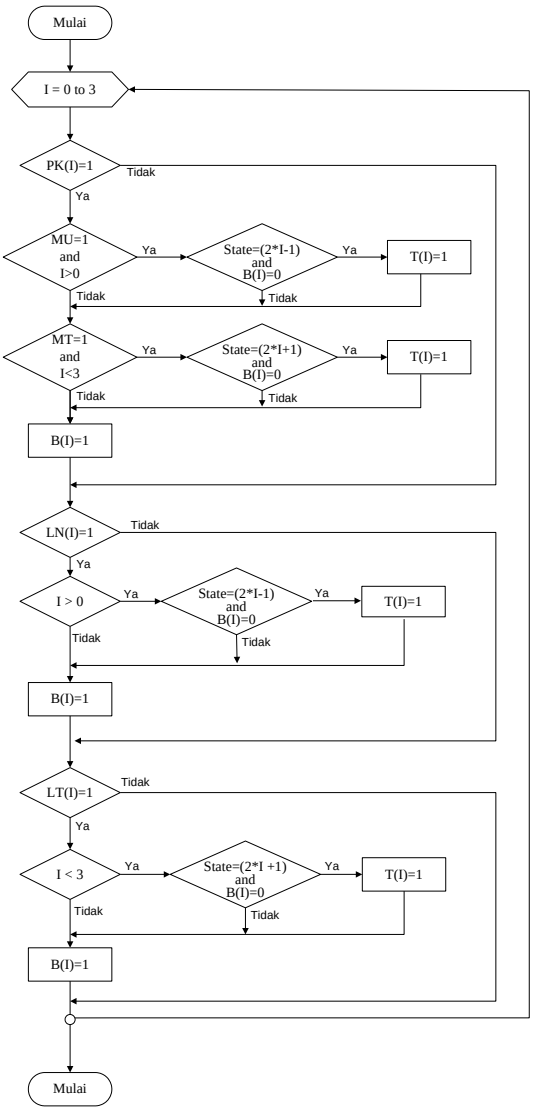
Disini()
Setiap permintaan ada berhenti pasti akan melalui proses Stop. Oleh karena itu Subroitine Disini() memeriksa apakah ada perimtaan berhenti di lantai ini, bila ada maka StopReq() dipanggil dan semua yang menyebabkan permintaan berhenti direset. Setelah itu subroutine ini menunggu adanya permintaan di atas atau di bawahnya dan state siap untuk pindah ke atas atau ke bawahnya dengan cara memanggil Naik() atau Turun().



Gambar 4. Flowchart Antara() dan Disini()

StopLift()
Subroutine ini mengatur proses Stop atau membuka dan menutup pintu. Ditinjau dari sudut FSM proses ini ialah perpindahan dari suatu state ke PB kemudian dari PB setelah waktu tertentu pindah ke PT dan setelah waktu tertentu pula kembali ke state sebelum PB.

BacaTombol()
Fungsi dari subroutine ini ialah mengeset nilai B(X) (permintaan berhenti di lantai X) dan T(X) (terlambat untuk berhenti di lantai X). Setiap penekanan tombol LN(X), LT(X), dan PK(X) maka B(X) di set. Sedangkan T(X) diset bila penekanan tersebut tepat pada saat state berada satu langkah sebelum state lantai X tersebut.



Gambar 5. Flowchart Baca Tombol()

Subroutine Lainya
Subroutine lainya sangat sederhana, flowchart-nya tidak digambarkan. Subroutine-subroutine tersebut ialah :

- Naik() : mengeset nilai MU, state pindah ke atas
- Turun() : mengeset nilai MT, state pindah ke bawah
- StopReq : Stop Request, mengeset nilai Stp dan WkTg (time counter).

7. Implementasi Pada Visual Basic

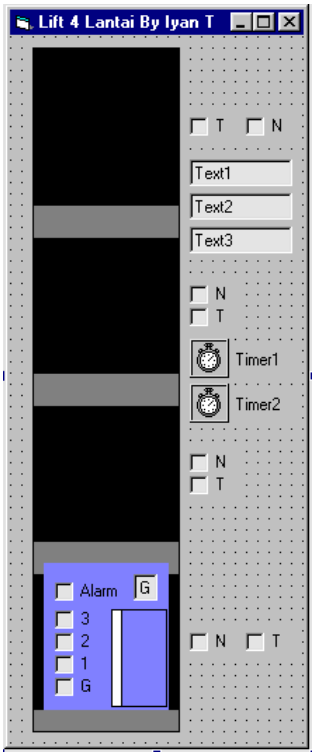
Program simulasi yang dibuat menggunakan bahasa Visual Basic ini merepresentasikan lift pada suatu grafik seperti pada Gambar 6. Bangunan dipresentasikan sebagai kotak hitam dengan kotak abu-abu sebagai batas lantai. Tombol permintaan naik/turun dari setiap lantai dibuat dengan menggunakan CheckBox. Tabung lift dibuat dengan menggunakan frame yang posisinya bisa diubah-ubah naik/turun terhadap bangunan (kotak hitam), didalamnya ada beberapa CheckBox dan ada pintu yang dibuat dari kotak putih yang lebarnya bisa diatur. Timer1 dan Timer2 digunakan untuk pengaturan software dan pergerakan lift / pintu.

Ada beberapa komponen yang digunakan untuk debugging dan tidak akan ada pada software yang sebenarnya yaitu ; CheckBox turun di lantai Ground dan CheckBox naik di lantai 3 yang ada hanya untuk penyederhanaan software (supaya tidak error) dan dibuat tidak visible. Selain itu ada TextBox di dalam tabung yang menunjukan lantai dan 3 buah TextBox di luar tabung. Text1 menunjukan State dan variabel naik (Nk) serta Turun (Tr). Text2 menunjukan permintaan berhenti. Text3 menunjukan input posisi tabung (SG, S1, S2, S3)

Code dari Form

Yang dimaksud dengan code pada VB ialah program yang menempel pada suatu form. Jadi semua flowchart di atas diterjemahkan ke dalam bahasa BASIC dan ditulis di code tersebut.

Selain dari subroutine yang berasal dari flowchart di atas ada dua subroutine yang merupakan adaptasi dengan software VB yaitu Timer1_Timer() dan Timer2_Timer(). Keduanya merupakan komponen VB yang aktif secara otomatis dengan perioda tertentu.

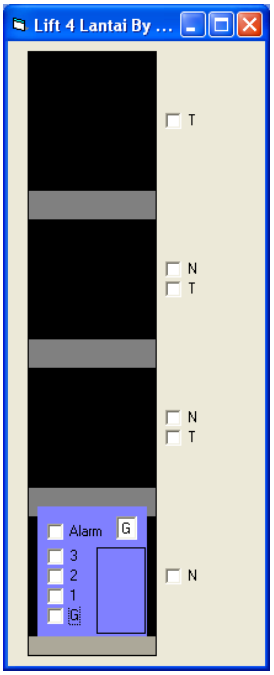


Gambar 6, Bentuk dan Komponen Form

Subroutine Timer1_Timer() digunakan untuk menghidupkan simulasi seolah-olah ia adalah hardware dari pengendali lift ini. Ia menaikan tabung lift bila ada masukan MU dan menurunkan lift bila ada masukan MT. Selain itu ia juga mengeluarkan sinyal S sesuai dengan posisi tabung lift. Subroutine Timer2_Timer() digunakan sebagai scheduler untuk menjalankan subroutine-subroutine yang sudah di tulis.

8. Pengujian dengan VisualBasic

Hasil pengujian dengan Visual Basic adalah sebagai berikut.



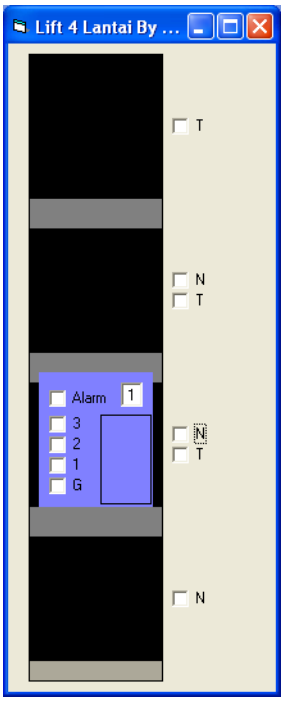
Gambar 7. Lift berada pada posisi lantai G

Posisi awal lift berada di lantai G. Pada posisi ini, lift menunggu permintaan untuk bergerak. Permintaan dapat dilakukan dari lantai mana saja. Jika CheckBox N pada lantai G di-klik maka pintu lift akan terbuka selanjutnya lift akan bergerak keatas sesuai dengan CheckBox (pilihan lantai) mana yang dipilih.

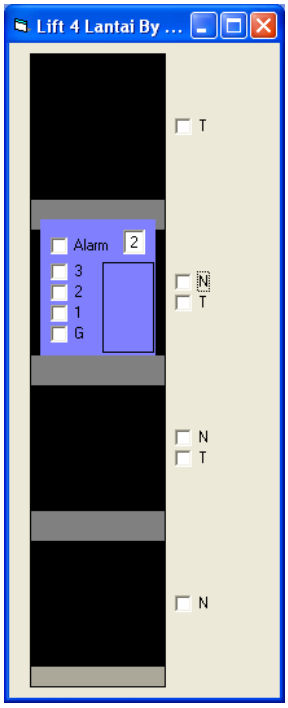
Pada saat lift bergerak keatas dari lantai G misalnya, untuk naik ke lantai 3, tiba-tiba saat berada di tengah antara lantai G dan lantai 1 CheckBox N pada lantai 1 di-klik, lift tidak akan berhenti di lantai 1 kare na penekanan CheckBox N sudah terlambat. Lift akan naik terlebih dahulu ke lantai 3 baru kemudian lift turun kelantai 1 untuk melayani permintaan. Hal ini berlaku juga untuk proses kebalikan yakni dari lantai atas ke lantai bawah.

Setelah pintu lift tertutup sehabis melayani permintaan dan bila tidak ada permintaan lain, lift bebas bergerak keatas atau kebawah tanpa mempedulikan permintaan naik atau turun yang dilakukan sebelumnya (salah satu CheckBox N atau T yang ditekan).

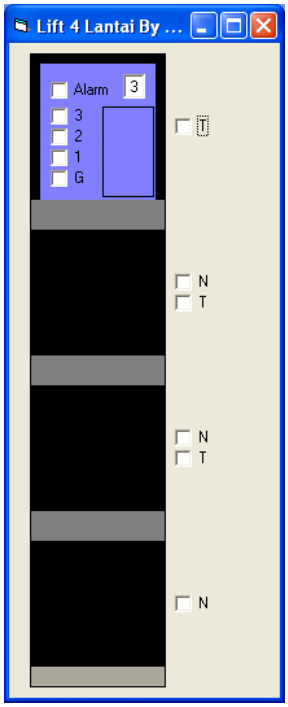
Apabila lift sedang bergerak keatas dari lantai G ke lantai 3, kemudian dilantai 2 dilakukan permintaan turun (CheckBox T di-klik), lift akan mengabaikan permintaan tersebut, maksudnya lift tidak akan berhenti di lantai 2. Dengan kata lain yang akan diprioritaskan adalah permintaan yang searah dengan gerakan lift. Lift akan berhenti di lantai 2 untuk melayani permintaan turun setelah selesai melayani permintaan naik dari lantai G ke lantai 3. Kecuali pada lantai 2 dilakukan permintaan untuk naik, Lift akan singgah di lantai 2 sebelum naik ke latai 3.



Gambar 8. Lift berada pada posisi lantai 1



Gambar 9. Lift berada pada posisi lantai 2



Gambar 10. Lift berada pada posisi lantai 3

9. Kesimpulan

Dalam perancangan software suatu sistem embedded apalagi sistem itu cukup kompleks hampir tidak mungkin tidak ada debugging. Meskipun secara logika sudah dibuat sesempurna mungkin, tetapi dalam penerjemahannya ke dalam software tetap masih bisa ada kesalahan.

Untuk proses debugging tersebut di atas sebaiknya dibuat suatu software yang hardware independent. Berdasarkan pengujian terhadap software yang hardware independent yang telah dibuat dapat disimpulkan bahwa software tersebut berjalan dengan baik sesuai dengan spesifikasi yang ditentukan. Hal ini sedikitnya telah membantu dalam membuat software yang hardware dependent.

Sebaiknya software yang hardware independent tersebut menggunakan bahasa yang sama dengan software yang hardware dependent. Hal ini disimpulan berdasarkan kesuitan yang kami alami dalam penerjemahan dari software yang hardware independent ke software yang hardware dependent.

Daftar Pustaka

1. David. E. Simon, (1999), ***An Embedded Software Primer***, Addison-Wesley
2. Kolman, Bernard. Busby, Robert C., (1984), ***Discreate Mathematcal Strucrures for Computer Science***, Prentice-Hall.
3. Markus Robijanto Kusuma, (1990), ***Belajar Turbo C Dengan Cepat dan Mudah***, Elex Media Komputindo.
4. Michael J.Pont, **(**2001), ***Patern for Time Triggered Embedded System,*** Addison-Wesley.
5. Microsoft, (1999), ***Mastering Visual Basic 6 Development***, Microsoft Press.
6. Microsoft, (1999), ***Mastering Visual Basic 6 Fundamentals***, Microsoft Press.
7. http://www.atmel.com