ISSN 1979-7451

Perancangan Sistem *Automatic Cruise Control* (ACC) Menggunakan Metode DePES (Development Process of Embedded System) Dengan Simulasi Java

Muhammad Iman Santoso[1]
[1]Electrical Engineering of Sultan Ageng Tirtayasa University, Serang-Indonesia
m.iman.santoso@ft-untirta.ac.id

ABSTRACT

Design is an important part of a system building. Design of the system could be existing system development or create from the scratch. This paper studies the system development in case of Automatic Cruise Control (ACC). DePES (Development Process of Embedded System) is used to design such system. This process focused on embedded software components, problem analysis, specifications, architechture and test. In order to illustrate and test the proposed system, JAVA simulation is provided.

ABSTRAK

Perancangan sangatlah penting dalam membuat sebuah sistem. Perancangan sistem sendiri bisa berdasarkan sistem yang sudah ada, dengan kata lain melakukan pengembangan sistem yang telah ada sebelumnya atau membuat sebuah sistem yang baru. Pada penelitian ini, perancangan sistem yang dilakukan yaitu mengambil studi kasus untuk sistem *Automatic Cruise Control* (ACC). Metode perancangan sistem yang dipakai menggunakan DePES (*Development Process of Embedded System*). Proses disini berfokus terhadap komponen *embedded software*, pemodelan terhadap permasalahan (*problem*), spesifikasi (*specifications*), arsitektur (*architecture*) dan pengujian (*test*). Untuk visualisasi dan pengujian dari perancangan sistem digunakan simulasi menggunakan JAVA.

Kata Kunci : Software embedded, DePES, Automatic Cruise Control (ACC), JAVA

1. PERANCANGAN SISTEM ACC

Dalam perkembangan dunia otomotif dewasa ini, salah satu aplikasi yang sedang hangat diperbincangkan yaitu sistem *Automatic Cruise Control* (ACC). Sistem ini dapat bekerja dengan cara menyesuaikan kecepatan kendaraan dan mengatur jarak kendaraan dengan kendaraan di depannya. Dengan dibenamkannya aplikasi sistem ACC ini pada kendaraan, tentunya dapat memberikan keselamatan dan kenyamanan baik bagi pengemudi, penumpang, dan pengendara kendaraan lain di sekitarnya [1]. Berdasarkan Peraturan Pemerintah (PP) Republik Indonesia no. 44 tahun 1993 tentang kendaraan dan pengemudi, Pasal 70 dan 71 ayat (1), (2), dan (3) menyebutkan bahwa komponen pendukung kendaraan bermotor salah satunya terdiri dari alat pengukur kecepatan, serta pengukur jarak [2]. Landasan hukum inilah yang mendasari ide dirancangnya sebuah sistem software terpadu (*embedded software*) untuk sistem *Automatic Cruise Control* (ACC).

Perancangan sistem ACC yang akan dibuat dalam penelitian ini dibuat berdasarkan metode perancangan sistem DePES (*Development Process of Embedded Sistem*). Dimana DePES merupakan metode baru perancangan pengembangan sistem *software embedded* yang didalamnya tergabung beberapa pemodelan sistem yang telah ada seperti UML(*Unified Modelling Language*), DFD (Data Flow Diagram), dan *Four variable model* [1] perancangan sistem ini akan disimulasikan menggunakan JAVA.

2. SISTEM AUTOMATIC CRUISE CONTROL (ACC)

Automatic Cruise Control (ACC) merupakan teknologi sistem kendali kemudi yang mampu menjaga jarak dan kecepatan dengan kendaraan di depan. Sistem ini akan mendeteksi kecepatan kendaraan di depan, selanjutnya sistem memerintahkan mobil untuk menyamakan kecepatan dan menjaga jarak dengan bantuan rem dan komputer pengatur kerja rem. [3].

Prinsip kerja sistem ACC pada perancangan modern saat ini, ACC dibuat dengan cara diaktifkan terlebih dahulu melalui tombol "on/off" yang biasanya diletakan didaerah stir, tentunya dalam hal ini, kendaraan harus menyala terlebih dahulu. Berbeda dengan perancangan sebelumnya, diamana ACC dibuat selalu dalam kondisi aktif. Dalam beberapa perancangan juga ACC ini dilengkapi dengan fungsi-fungsi tombol "set", "resume", "accelerate", dan "cancel". Selain itu, alternative lain menonaktifkan ACC adalah dengan cara menginjak pedal rem, sehingga pengemudi dapat meneruskan kemudi kendaraan tanpa harus menonaktifkan sistem ACC terlebih dahulu. Sistem ini dibuat semudah mungkin dan diletakan di sekitar stir kemudi, sehingga akan mudah terjangkau pada saat pengoperasian oleh seorang pengemudi. Umumnya tombol-tombolnya berada pada batang atau tangkai belok (*turn signal stalk*) yang dirancang pada kendaraan buatan *General Motors (GM), Toyota, BMW, dan Mercedes Benz*. Berbeda dengan kendaraan *Honda* yang meletakkan tombol ACC ini pada stir kemudi.

3. REKAYASAPIRANTI LUNAK

Rekayasa piranti lunak adalah pengembangan dan penggunaan prinsip pengembangan untuk memperoleh perangkat lunak secara ekonomis yang *reliable* dan bekerja secara efisien dan *real time* [4].

3.1. Pendekatan rekayasa perangkat lunak

3.1.1 Metode konvensional

Pengembangan sistem pada rekayasa piranti lunak dengan metode konvensional terdapat dua komponen yang digunakan untuk menganalisa sistem ketika membuat pemodelan, yaitu *Data-Flow Diagram* (menggambarkan fungsi sistem), dan *Four Variable Model* (menggambarkan 4 variabel yang menganalisa pada lapisan software arsitektur).

Data flow Diagram dalam hal ini berarti yang digunakan dalam *model environment*, yaitu *Context-Diagram* atau yang digunakan untuk penurunan level yaitu leveled. Untuk menganalisa sistem lebih efektif lagi, dapat digunakan suatu metode. Ada dua metode pendekatan dalam membangun sistem, yang pertama yaitu Top-Down. Pada metode ini sistem diturunkan dari pemetaan secara global yang kemudian akan menurun ke arah yang lebih deskriptif. Metode yang kedua yaitu Bottom-Up, dalam hal ini sistem dipetakan dari satuan terkecil sehingga kesatuan terbesar. *Four Variable Model* merupakan konsep pendekatan empat varibel yang menjadi fokus dalam melakukan observasi pada lapisan arsitektur software untuk komponen yang menjadi bagian program, variabel ini meliputi *Monitored variable*, *Controlled variable*, *Input data* dan *Output data* [4]

3.1.2. Konsep *Object Oriented*

Teknologi *Object Oriented* merupakan paradigma baru dalam rekayasa piranti lunak (*software*). Metode ini didasarkan pada objek dan kelas. *Object Oriented* memandang *software* bagian per bagian, dan menggambarkan satu bagian tersebut dalam satu objek. Satu objek dalam sebuah sebuah model merupakan suatu fokus selama dalam proses analisis, desain, dan implementasi dengan menekankan state, perilaku (*behavior*), dan interaksi objek-objek [4].

3.1.3. Konsep Unified Modelling Language (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. Unified Modelling Language (UML) merupakan sistem arsitektur yang bekerja dalam OOAD (*Object Oriented Analysis And Design*) dengan satu bahasa yang konsisten untuk menentukan, menvisualisasi, mengonstruksi, dan mendokumentasikan program-program yang terdapat dalam sistem software [5-13].

3.2. Metode *DePES (Development Process Of Embedded system)*

Metode DePES (*Development Process of Embedded System)* [14] yang berarti adalah metode pengembangan proses untuk sistem komputer tertanam, bersifat menjelaskan bagian-bagian dari perancangan suatu sistem. Tahapan-tahapan dari mulai apa dan bagaimana dijelaskan secara menyeluruh sehingga membentuk pola arsitektur sistem dan program yang dapat dikembangkan. Metode ini sangat efektif sekali dalam hal memberikan informasi terpadu dari suatu sub-sub program, sehingga memudahkan para *programmer* untuk dapat mengetahui kesalahan-kesalahan dan pengembangan program. Langkah-langkah perancangan sistem dengan metode DePES yaitu:

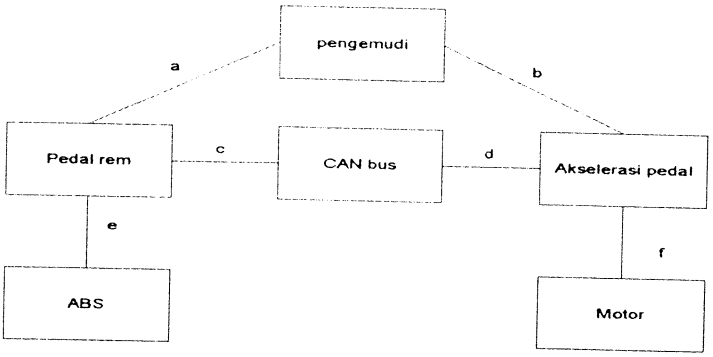
1. Menggambarkan permasalahan sistem (*describe problem*)
2. Membuat persyaratan sistem (*consolidate requirements*)
3. Menguraikan permasalahan dalam sistem (*decompose problem*)
4. Menghubungkan kebiasaan spesifikasi dari *machine* untuk tiap sub masalah (*derive a machine behavior specifications for each sub problem*)
5. Merancang sistem arsitektur secara global (*design global system architecture*)
6. Menghubungkan spesifikasi setiap komponen untuk sistem arsitektur global (*derive specifications for all components of the global system acrhitecture*)
7. Merancang sebuah arsitektur untuk setiap komponen yang dapat diprogram dari sistem arsitektur global yang dibangun, kemudian di implementasikan ke dalam bagian software. *(design an architechture for all programmable component of global system architecture that will be implemented in software).*
8. Menjelaskan behavior dari seluruh komponen yang menjadi bagian software arsitektur menggunakan sequence diagram. (*Specify the behavior of all components of all software architectures, using sequence diagrams).*
9. Mengimplementasikan komponen menjadi program, dan melakukan test environtment. (*Implement software components and test environment)*
10. Mengintegrasikan seluruh komponen yang telah diprogram menjadi software yang utuh dan test keseluruhan sistem software.

(Integrate and test software components)

4. PERANCANGAN SISTEM

4.1 Tahap pertama: *Describe Problem*

Pada fase pertama ini mendepenelitiankan tahapan perancangan sistem awal Automatic Cruise Control (ACC), berikut dijelaskan dalam *context diagram.*



Gambar 1. Context Diagram sistem yang ada

Phenomena hubungan antar domain di atas yaitu :

a: (Pengemudi menekan pedal rem, Pengemudi melepaskan pedal rem)

b: (Pengemudi menekan pedal akselerasi, Pengemudi melepaskan pedal akselerasi)

c: (Pedal rem memberikan informasi posisi pedal rem ke CAN Bus)

d: (Pedal akselerasi memberikan informasi posisi pedal akselerasi ke CAN Bus)

e: (mengaktifkan mekanisme rem, menonaktifkan mekanisme rem)

f: (menambah kecepatan motor, mengurangi kecepatan motor)

4.2 Tahap kedua: *Consolidate Requirement*

Pada fase kedua ini, akan diperkenalkan *context diagram* sistem yang dikembangkan (*developed system*),

Phenomena hubungan antar domain diatas adalah :

a : (Pengemudi menekan pedal rem, Pengemudi melepaskan pedal rem)

b: (Pengemudi menekan pedal akselerasi, pengemudi melepaskan pedal akselerasi)

c: (Pedal rem memberikan informasi posisi pedal rem ke CAN Bus)

d: (Pedal akselerasi memberikan informasi posisi pedal akselerasi ke CAN Bus)

e: (mengaktifkan mekanisme rem, menonaktifkan mekanisme rem)

f: (meningkatkan kecepatan motor, mengurangi kecepatan motor)

g: (tekan tombol OFF, tekan tombol RESUME, tekan tombol PLUS, tekan tombol MINUS, tekan tombol SET, peringatan visual)

h: (perintah panel kontrol)

i: (informasi radar untuk nilai jarak)

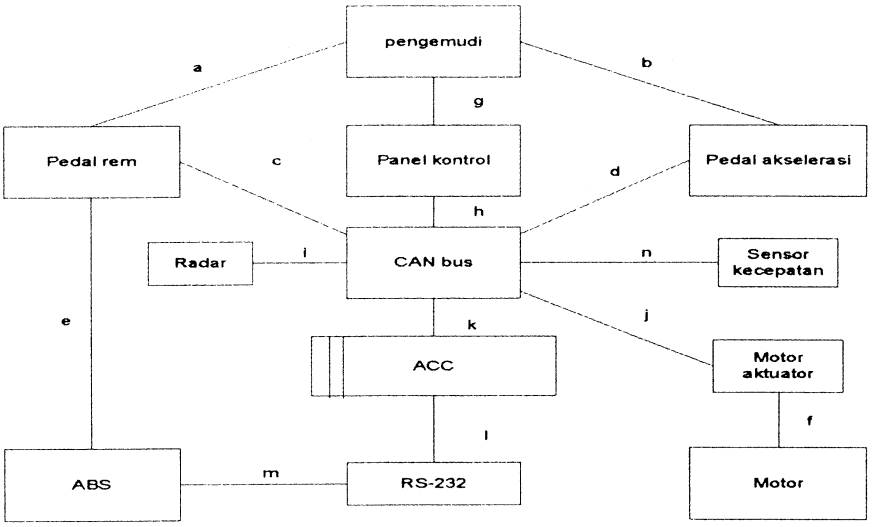
j: (sinyal akselerasi)

k: (menerjemahkan/membaca pesan CAN, menulis pesan CAN)

l: (sinyal pengereman)

m: (mengaktifkan ABS)

n: (kecepatan motor)



Gambar 2 Context Diagram sistem yang ada

4.3 Tahap ke tiga: *Decompose Problem*

Pada fase ini merupakan penyusunan kelompok permasalahan kedalam bentuk *problem diagram.*

Phenomena hubungan antara domain untuk problem diagram di atas adalah :

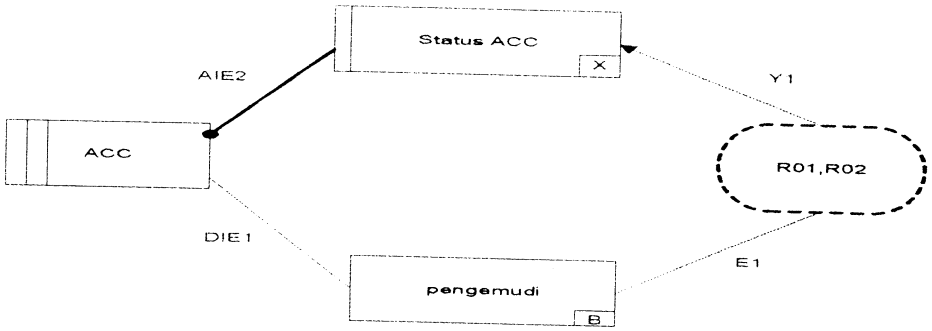
E1: (perintah "ON", perintah "OFF")

4.3.1 *Problem Diagram*

4.3.1.1 *Problem Diagram* untuk "ACC-Switching"

E2: (status ditetapkan ON, status ditetapkan OFF)

Y1: (mengaktifkan ACC, menonaktifkan ACC)



Gambar 3 Problem Diagram ACC Switching

4.3.1.2 Problem Diagram "Pengemudi mengendalikan kecepatan melalui ACC"

Phenomena hubungan antar domain diatas yaitu:

E1: (Perintah "+", perintah "-", perintah "SET")

E2:(Tambah kecepatan yang dikehendaki, kurangi kecepatan yang dikehendaki, tetapkan kecepatan yang dikehendaki)

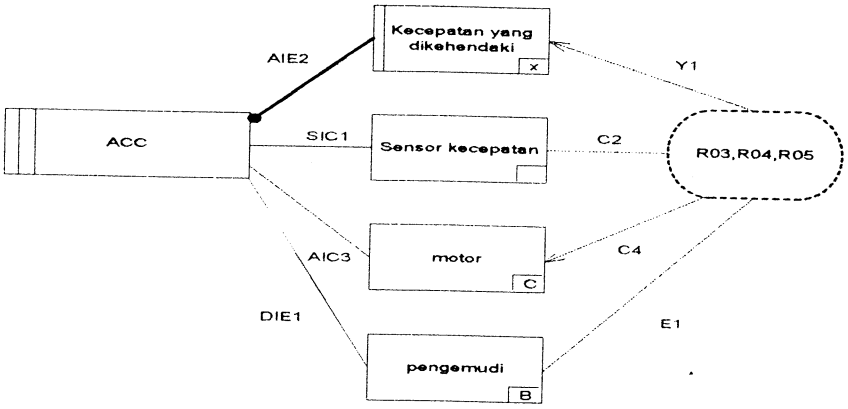
Y1: (Ubah kecepatan yang dikehendaki)

C1: (Sinyal sensor kecepatan)

C2: (Kecepatan motor saat ini)

C3: (Tambah kecepatan motor, kurangi kecepatan motor)

C4: (Ubah kecepatan motor)



Gambar 4 Problem Diagram Pengemudi Menggunakan ACC

3.3.1.3 Problem Diagram "ACC mengendalikan mobil"

Phenomena hubungan antar domain di atas adalah :

C1: (Tambah kecepatan motor, kurangi kecepatan motor, Aktifkan ABS, nonaktifkan ABS)

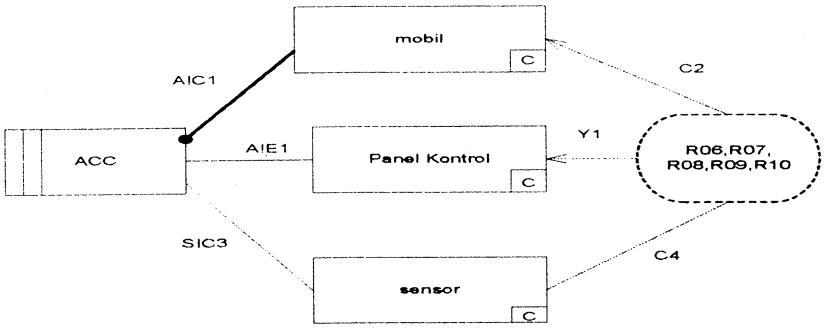
C2: (Ubah kecepatan motor, rem kendaraan)

C3: (Sinyal sensor kecepatan, sinyal sensor radar)

C4: (Kecepatan motor, jarak dengan mobil di depan)

E1: (Peringatan visual)

Y1: (Informasi pengemudi)



Gambar 5 Problem Diagram ACC Mengendalikan Mobil

4.3.1.4 Problem Diagram "interrelasi Driver dan ACC"

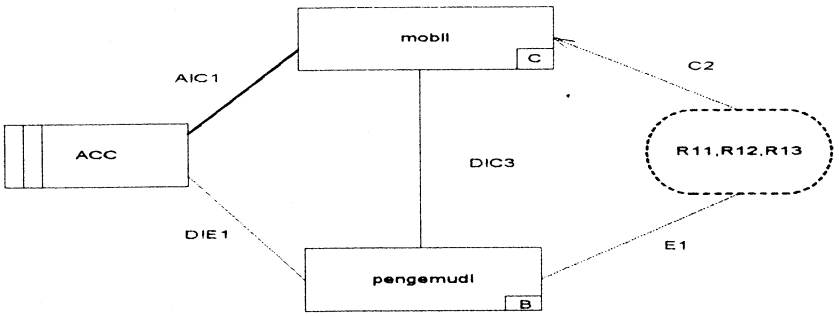
Phenomena hubungan antar domain di atas yaitu:

E1: (Perintah panel kontrol)

C1: (Tambah kecepatan motor,kurangi kecepatan motor, aktifkan ABS, nonaktifkan ABS)

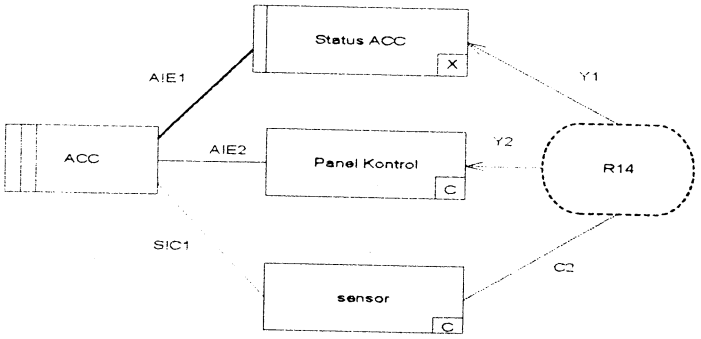
C2 : (Ubah kecepatan motor, rem kendaraan)

C3: (Akselerasi melalui pedal, Rem melalui pedal)



Gambar 6 Problem Diagram Interrelasi Driver dan ACC

4.3.1.5 Problem Diagram "Reaksi jika Sensor Rusak"



Gambar 7 Problem Diagram Reaksi Bila Sensor Rusak

Phenomena hubungan antar domain di atas yaitu:

E1: (Tetapkan status ON, tetapkan status OFF)

Y1: (Aktifkan ACC, Nonaktifkan ACC)

E2: (Sinyal peringatan visual)

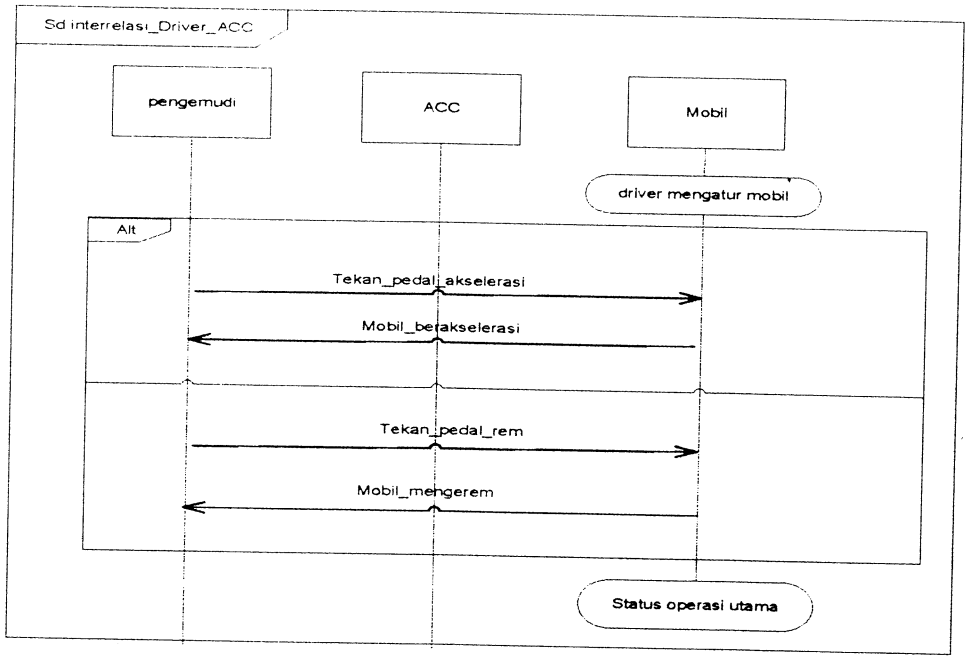
Y2: (Informasi pengemudi)

C1: (Sinyal sensor radar, sinyal sensor kecepatan)

C2: (Informasi status sensor)

4.4 Tahap keempat: *Derive a Machine behavior specification for each sub problem.*

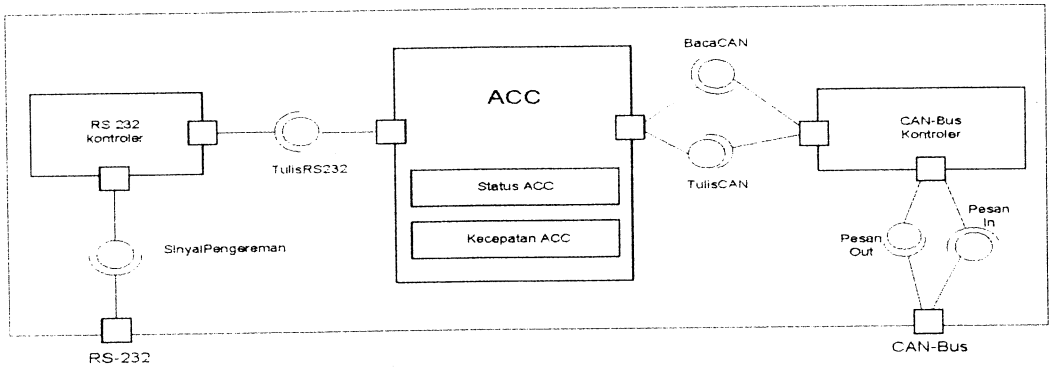
Pada fase ini, akan dibuat *sequence diagram* untuk menjelaskan masing-masing sub problem pada fase ke 3. Salah satu contoh dari *sequence diagram* ini terlihat pada gambar 7.



Gambar 8 Problem Diagram Interrelasi Driver dan ACC

4.5. Tahap kelima: *Design of Global System Architecture (GSA)*

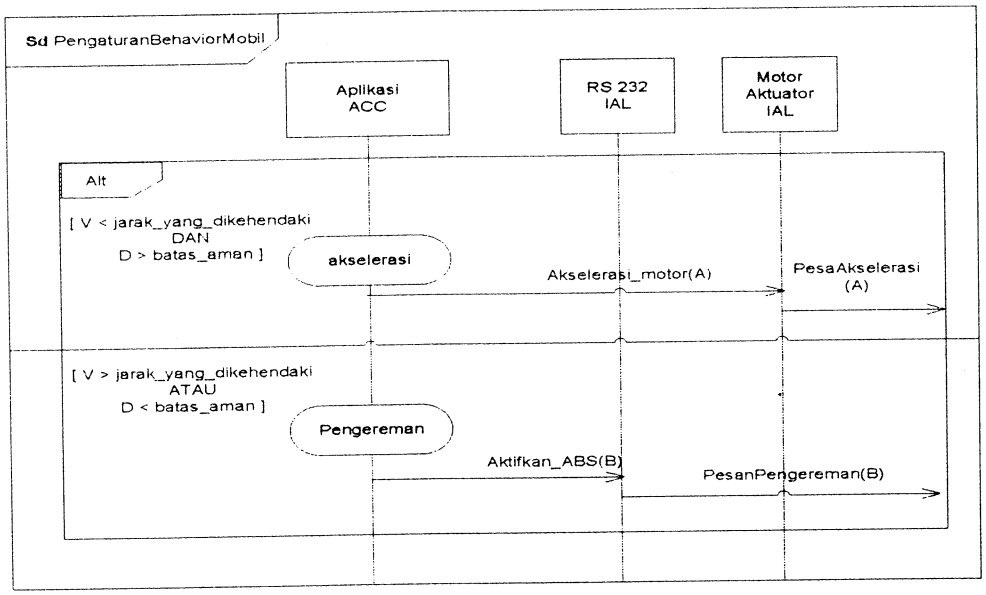
Pada fase ini, dijelaskan perancangan arsitektur global dari sistem ACC.



Gambar 9 Sistem Arsitektur Global

4.6 Tahap keenam: *Derive specifications for all component of GSA*

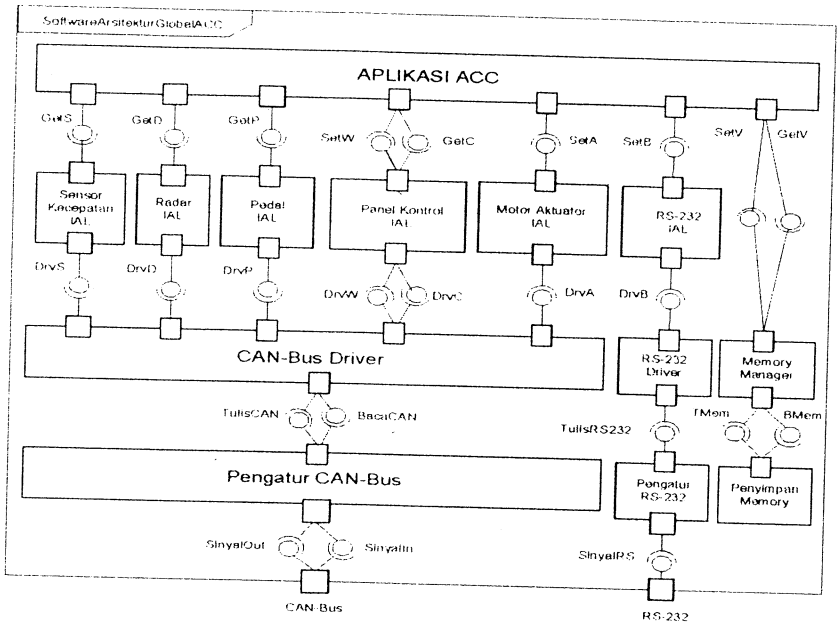
Tahapan ini merupakan detail perilaku komponen *interface* pada arsitektur sistem.



Gambar 10 Spesifikasi detail perilaku dari komponen RS232 dan Aktuator Motor

4.7 Tahap ketujuh: *Design software Architecture for all component of GSA*

Fase ini menjelaskan perancangan software untuk komponen sistem ACC.



Gambar 11 Arsitektur Global Software

4.8. Tahap ke delapan: *specify the behavior of all component from software architecture, using sequence diagram.*

Menjelaskan semua perilaku komponen pada arsitektur *software* fase ke tujuh kedalam *sequence diagram*. Perilaku komponen-komponen yang ada dalam arsitektur *software*, diuraikan dalam *sequence diagram* IAL (interface abstraction layer) untuk Sensor Kecepatan IAL, Radar IAL, Pedal IAL, Panel Kontrol IAL, Motor Aktuator IAL, RS-232 IAL.

4.9 Tahap ke sembilan: *Implement software component and test environment*

Pada fase ini mengimplementasikan bagian-bagian dari keseluruhan komponen software.

4.10 Tahap ke sepuluh: *Integrate and Test software components*

Pada fase ini menjelaskan proses penyatuan beberapa komponen yang telah diprogram pada arsitektur software fase ke 4.7, dan behaviour komponen fase ke 4.8 yang diimplementasikan kedalam sub-sub kelas dan interfaces. Proses integrasi ini bertujuan untuk menyempurnakan arsitektur software kedalam program JAVA, dan menerapkannya menjadi sebuah software yang utuh sehingga kemudian bisa dilakukan pengujian.

5. Pengujian dan Analisa Sistem

Setelah melakukan integrasi menjadi program yang utuh, langkah selanjutnya adalah melakukan pengujian. Pengujian dilakukan agar mengetahui apakah sistem tersebut berjalan sesuai dengan yang diinginkan atau tidak. Pengujian menggunakan metode *black box* yang mengutamakan penilaian terhadap pemenuhan kebutuhan fungsionalitas perangkat lunak.

5.1 Prosedur pengujian

Pada perancangan sistem *Automatic Cruise Control* (ACC) ada 2 aktivitas/proses utama yang difokuskan untuk pengujian, yaitu:

1. Perilaku antarmuka (*interfaces*)
   Mengacu pada fase ke 4.6 menjelaskan proses bagaimana interaksi *interface*, mulai dari inisialisasi tampilan awal, input perintah dari panel control, RESUME, PLUS, MINUS, SET, OFF, dll. diwakili oleh sebuah tampilan pada simulasi aplikasi program ACC.
2. Perilaku komponen
   Mengacu pada fase ke 3.8 dan *sequence diagram* 3.4 menjelaskan relevansi skenario yang dilakukan untuk mendepenelitiankan sistem keseluruhan. Bagaimana komponen-komponen yang dirancang menyesuaikan behaviornya dengan aplikasi program ACC.

5.2 Faktor-faktor yang diuji

Faktor-faktor yang diuji dalam software yang dibuat meliputi [15] :

1. Pengujian komponen, adalah pengujian pada setiap komponen pembentuk modul
2. Pengujian modul adalah pengujian terhadap kesatuan komponen pembentuk modul
3. Pengujian sistem adalah pengujian sistem operasional setelah menyatukan modul.

Implementasi sistem ACC dibagi menjadi 2 modul (lihat 3.9):

1. Modul inti ACC
   Modul inti ACC terdiri dari beberapa komponen sebagai berikut :
   a. Pedal akselerasi
   b. Pedal rem
   c. Sensor kecepatan
   d. Sensor radar
   e. Panel control
2. Modul CAN-Bus Driver
   Modul CAN-Bus terdiri dari motor aktuator dan RS-232.

Modul-modul ini merupakan komponen yang masing-masing telah diimplementasikan ke dalam program sub-sub kelas JAVA (lihat tahap 3.9).

komponen-komponen yang telah terprogram diintegrasikan kedalam kelas utama, barulah proses pengujiannya dapat dilakukan. Proses pengujiannya dilakukan dengan membuka, *script* tersebut pada JAVA NetBean IDE 6.7, kemudian dilakukan *debug* file untuk mengetahui apakah *script* java tersebut sudah benar atau masih ada kesalahan [16].

Selain itu, untuk pengujian sistem secara keseluruhan ada 6 skenario pengujian. Tujuannya adalah untuk menguji apakah modul di atas dapat berjalan dan sesuai dengan rancangan. Berikut skenario pengujiannya adalah :

1. Skenario 1 "inisialisasi awal"
2. Skenario 2 "ACC *switching*"
3. Skenario 3 "Pengemudi mengendalikan kecepatan via ACC"
4. Skenario 4 "ACC mengendalikan Mobil"
5. Skenario 5 "hubungan Pengemudi dan ACC".
6. Skenario 6 "Reaksi ACC jika sensor rusak"

5.3 Cara pengujian dan konfigurasi pengujian

Pengujian sistem dilakukan dengan cara mengaplikasikan dan menjalankan program pada sebuah komputer. Konfigurasi *hardware* yang digunakan adalah sebuah komputer yang diharapkan dapat mewakili dan penyusun mengharapkan dengan model seperti ini waktu pengujian tidak terlalu lama dan tanpa mengurangi kualitas pengujian. Konfigurasi yang digunakan dalam pengujian adalah: PC/Laptop dengan sistem operasi Windows XP professional dengan JAVA NetBeans IDE 6.7+JDK 5.0 dan minimum prosessor Intel Pentium III, kapasitas harddisk 20 GB, dan RAM 128 MB.

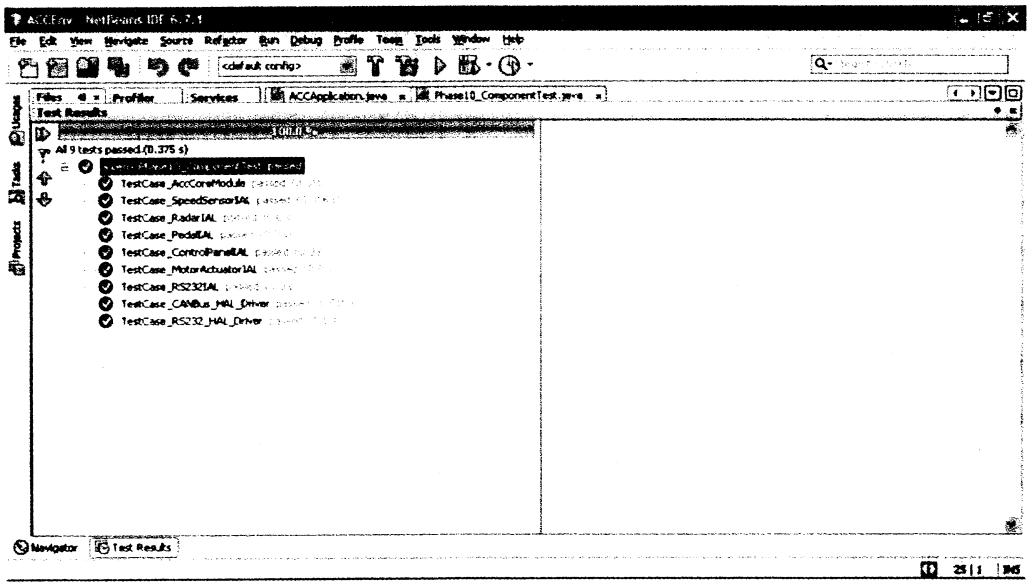
5.4 Hasil dan Analisa Pengujian

Hasil dan analisa pengujian dapat dilihat dalam tabel-tabel berikut ini :

Tabel 1 Hasil pengujian untuk proses insialisasi *RESUME, PLUS, MINUS, SET, OFF*

No	NAMA PROSES	SKENARIO PENGUJIAN	YANG DIHARAPKAN	HASIL
1	RESUME	tekan tombol RESUME	>sistem mengaktifkan ACC >CAN menerima kode 504 dari panel kontrol > CAN mengirimkan kode pesan 601 peringatan visual (ACC-ON)	memenuhi
2	PLUS	tekan tombol PLUS	>kecepatan yang dikehendaki bertambah (v+10) >CAN menerima kode 501 dari panel kontrol	memenuhi
3	MINUS	tekan tombol MINUS	>kecepatan yang dikehendaki bertambah (v-10) >CAN menerima kode 502 dari panel kontrol	memenuhi
4	SET	tekan tombol SET	>tetapkan kecepatan sebagai kecepatan yang dikehendaki >CAN menerima kode 503 dari panel kontrol	memenuhi
5	OFF	tekan tombol OFF	>sistem menonaktifkan ACC >CAN menerima kode 505 dari panel kontrol >CAN mengirimkan kode 601 untuk menginformasikan status ACC-OFF	memenuhi

Pengujian komponen dilakukan cara *men-debug*, untuk mengetahui kesalahan dalam program. Berikut *debug* file untuk *component test* :



Gambar 12 Hasil debug tes komponen

Pada Gambar 12 menunjukan hasil pengujian komponen dengan cara melakukan *debug*, dan hasilnya 100% test tidak mengalami kesalahan, dengan catatan waktu total untuk men-*debug* adalah 0,375 detik. Berikut rincian waktu untuk komponen-komponen yang diuji :

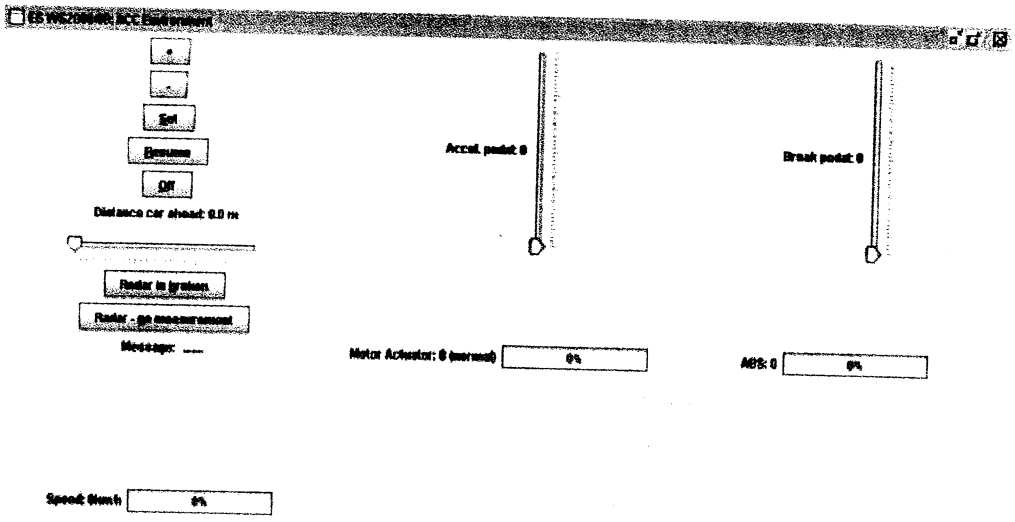
Tabel 2 hasil pengujian komponen

Jenis test komponen	Status	Lamanya proses (detik)
AccCoreModule	passed	0,015
speedsensorIAL	passed	0,016
RadarIAL	passed	0
PedalIAL	passed	0
ControlPanelIAL	passed	0
MotorAktuatorIAL	passed	0
RS232IAL	passed	0
CANBusHALDriver	passed	0,015
RS232HALDriver	passed	0

Selain itu, untuk pengujian sistem secara keseluruhan ada 6 skenario pengujian. Tujuannya adalah untuk menguji apakah modul diatas dapat berjalan dan sesuai dengan rancangan. Berikut skenario pengujiannya adalah :

1. Skenario pertama "inisialisasi awal"

Skenario ini merupakan pengujian tahap pengenalan tampilan awal, pada saat ACC akan digunakan. Dengan mengacu *sequence diagram* 3.6.1, Pada proses "insialisasi awal", kedua pedal (akselerasi dan rem) dilepaskan, jarak dengan mobil di depan belum diterima. Kecepatan mobil sama dengan 0 km/jam, kecepatan ACC yang dikehendaki belum ditentukan. Dengan parameter tersebut, maka output sistem di atas akan ditampilkan oleh JAVA, sebagai berikut :



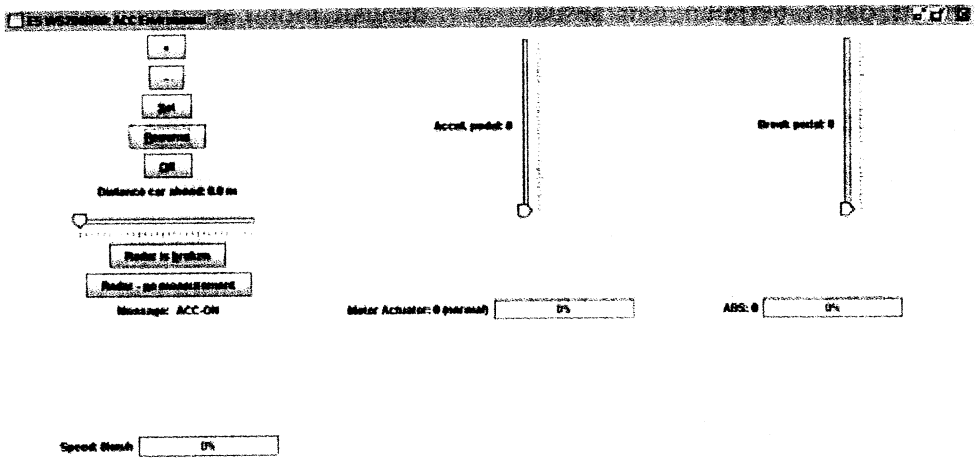
Gambar 13 Setting Inisiasi ACC

Output gambar 13 di atas menjelaskan bahwa insialisasi awal tampilan rancangan program ACC, dengan kondisi ACC belum diaktifkan, pedal akselerasi dan pedal rem berada pada posisi dilepaskan.

2.Skenario kedua "ACC-switching"

Skenario ini merupakan pengujian bagaimana respon software terhadap pengaturan *switching* (penukaran) kondisi status ACC, mengacu pada sequence diagram 3.4.1

Pada proses "ACC-switching", skenarionya adalah kedua pedal (akselerasi dan rem) dilepaskan, jarak dengan mobil didepan belum diterima. Kecepatan mobil sama dengan 0 km/jam, kecepatan ACC yang dikehendaki belum ditentukan. Pengemudi mengaktifkan ACC dengan menekan tombol RESUME pada panel kontrol. Dengan parameter tersebut, maka output sistem diatas akan ditampilkan oleh JAVA, sebagai berikut:



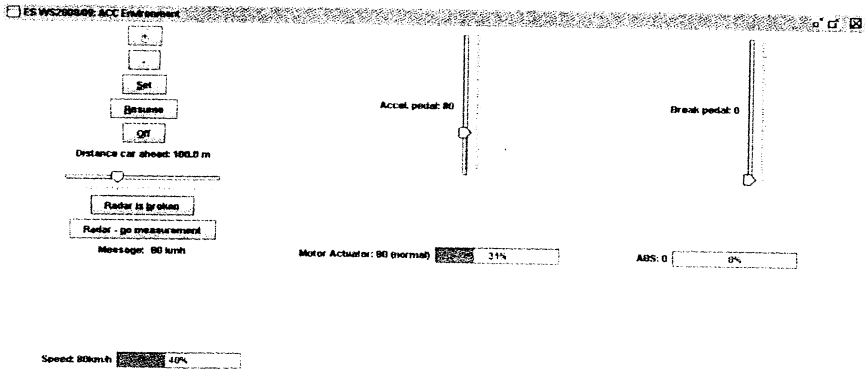
Gambar 14 *ACC Switching*

3. Skenario ketiga "Pengemudi Mengendalikan Kecepatan melalui ACC"

Skenario ini merupakan pengujian bagaimana respon software terhadap perintah pengemudi untuk mengendalikan ACC, mengacu pada sequence diagram 3.4.2

Pada proses ini, skenarionya adalah pedal akselerasi ditekan sampai kecepatan mobil menunjukan 80 km/jam, jarak dengan mobil didepan dalam jarak aman. Kecepatan ACC

yang dikehendaki ditetapkan 80 km/jam. Pengemudi telah mengaktifkan ACC. Dengan parameter diatas, maka output sistem diatas akan ditampilkan oleh JAVA, sebagai berikut :

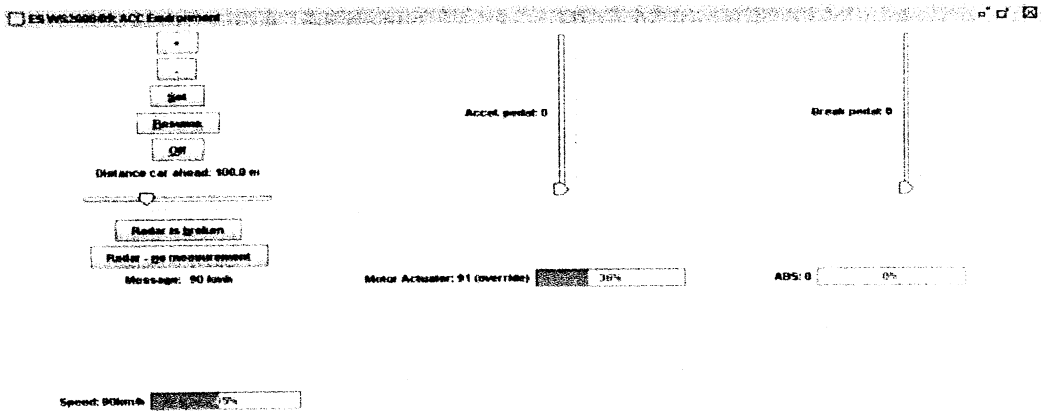


Gambar 15 Pengemudi Mengeset Kecepatan 80 Km/Jam

Output gambar 15 memperlihatkan bahwa pengemudi memberikan perintah PLUS, untuk menambah kecepatan mobil (v +10 km/jam), dan memberikan perintah SET untuk menetapkan kecepatan ACC.

4. Skenario keempat "ACC Mengendalikan Mobil"

Skenario ini merupakan pengujian bagaimana respon software terhadap mobil ketika diberi perintah oleh ACC, mobil menerjemahkan pesan ACC untuk mendeteksi jarak, dan mendeteksi kecepatan mobil mengacu pada sequence diagram 3.4.1.4. Pada proses ini, skenarionya adalah pedal akselerasi ditekan sampai kecepatan mobil menunjukan 100 km/jam, jarak dengan mobil didepan dalam jarak aman. kecepatan ACC yang dikehendaki ditetapkan 90 km/jam. Pengemudi mengaktifkan ACC. Kemudian pedal akselerasi dilepaskan. Dengan parameter diatas, maka output sistem di atas akan ditampilkan oleh JAVA, sebagai berikut:

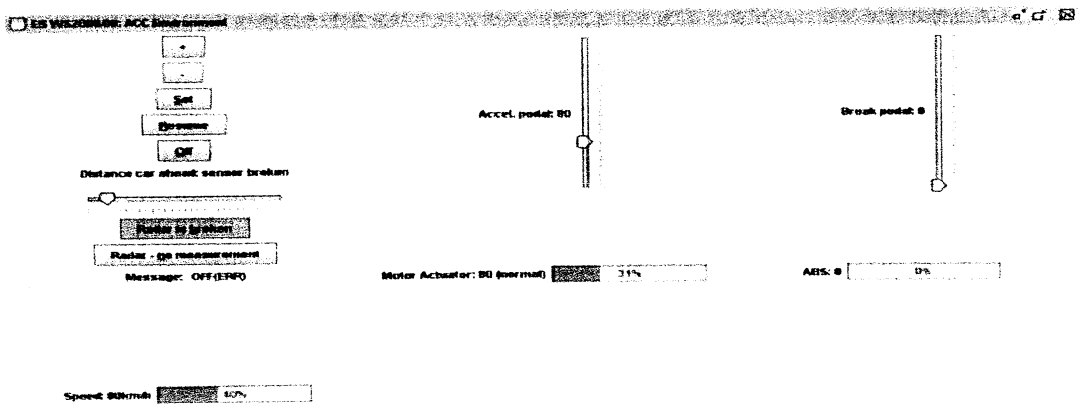


Gambar 16 ACC Mengendalikan Mobil pada 90 Km/Jam

Output ini menggambarkan bahwa pengemudi memberikan perintah PLUS, untuk menambah kecepatan mobil (v +10 km/jam), dan memberikan perintah SET untuk menetapkan kecepatan ACC. Kemudian Pedal akselerasi dilepaskan oleh pengemudi, yang terjadi adalah kecepatan kendaraan akan tetap sesuai kecepatan yang dikehendaki pada ACC. Dengan kata lain ACC menjaga kecepatan meskipun pedal akselerasi dilepaskan. Hal ini sesuai dengan sistem misi perancangan.

5. Skenario kelima "Interrelasi Pengemudi dan ACC"

Skenario ini merupakan pengujian bagaimana respon software terhadap hubungan ACC sesuai perintah pengemudi untuk mengendalikan mobil, mengacu pada sequence diagram 4.4.1.4 Pada proses ini, skenarionya adalah pedal akselerasi ditekan sampai kecepatan mobil menunjukan 100 km/jam, jarak dengan mobil didepan dalam jarak aman. kecepatan ACC yang dikehendaki ditetapkan 100 km/jam. Akan tetapi pengemudi menambah kecepatan hingga pedal akselerasi menunjukkan posisi 120 km/jam. Dengan parameter tersebut, maka output sistem diatas akan ditampilkan oleh JAVA, sebagai berikut:



Gambar 18 Reaksi ACC Jika Sensor Rusak

Output program di atas menggambarkan ketika terjadi sensor rusak atau sensor tidak menerima sinyal informasi jarak, kecepatan mobil akan dikendalikan oleh pengemudi dengan mengacu kecepatan terakhir yang ditentukan pengemudi. kemudian ACC akan memberikan sinyal peringatan berupa peringatan visual dengan pesan ACC OFF(ERR). Dengan kemungkinan terjadinya kondisi ini, sistem telah dirancang agar mobil dapat terus beroperasi melanjutkan perjalanan sesuai perintah pengemudi.

5.5 Analisa Manfaat

Metode *Development Process Of Embedded System* (DePES) merupakan metode baru yang menyajikan suatu proses pengembangan sistem berbasis sistem *embedded*. Dimana keunggulan dari metode ini yaitu dalam proses semua tahapan yang dikerjakan memiliki sebuah pola yang terstruktur rapi dan jelas. DePES ini memfokuskan terhadap pola pengembangan berbasis komponen untuk sistem *embedded* [14]. Kompleksitas dalam pembuatan sistem embedded, seperti halnya pembuatan yang dibuat dalam skala besar, dan membutuhkan tingkat ketelitian untuk meminimalisasi kesalahan-kesalahan dalam sistem, memicu lahirnya metode DePES ini. Mudah-mudahan diharapkan dengan metode ini, dapat membantu mempercepat dalam siklus perencanaan software (*software life cycle*) [1], mengelola dan mendefinisikan permasalahan dalam tiap bingkai masalah (*problem frames*), menjelaskan secara terperinci prinsip-prinsip proses pengembangan berikut :

a. Menentukan seluruh aktivitas utama

b. Menggunakan sumber daya, dengan batasan tertentu, untuk menghasilkan produk awal dan produk akhir
c. Tersusun dari beberapa sub proses
d. Setiap aktivitas mempunyai kriteria masukan dan keluaran
e. Aktivitas diorganisasikan secara teratur
f. Mempunyai panduan yang menjelaskan tujuan
g. Aturan validasi dipakai pada tiap aktivitas, sumberdaya dan produk.

6. KESIMPULAN

Telah dilakukan perancangan sistem *software* tertanam (*embedded system*) dengan menggunakan metode DePES (*Development Process of Embedded System*) untuk studi kasus *Automatic Cruise Control* (ACC). Aktivitas perancangan sistem berjalan dengan baik. Tiap tahapan dari proses pengembangan benar-benar dilakukan dengan teliti dan sesuai prosedur, divalidasi serta dikaji sesuai *requirement* dan misi perancangan sistem.

Dari hasil pengujian, dapat disimpulkan bahwa fungsi-fungsi *software* berjalan dengan baik, sesuai input dan output yang diinginkan. Fungsi-fungsi ini meliputi fungsi *RESUME, PLUS, MINUS, SET,* dan *OFF*. Pengujian sistem secara keseluruhan dengan menerapkan fungsi-fungsi tersebut kedalam skenario-skenario berikut:

1. Inisialisasi awal; menghasilkan output tampilan utama program, dengan kondisi status ACC belum aktif, nilai akselerasi dan nilai pengereman yang diharapkan tidak ada.
2. ACC *switching*; menghasilkan ouput tampilan utama program, dengan mengindikasikan perubahan status ACC. Status ACC ON, sistem diaktifkan, nilai akselerasi dan nilai pengereman yang diharapkan tidak ada.
3. Pengemudi mengendalikan kecepatan melalui ACC; menghasilkan output simulasi program, dengan status ACC diaktifkan, pengemudi menetapkan kecepatan ACC, nilai akselerasi yang diharapkan normal, dengan nilai pengereman tidak ada.

   ACC mengendalikan mobil; menghasilkan ouput simulasi program, dengan kondisi pedal akselerasi dilepaskan, nilai akselerasi yang diharapkan normal dan nilai pengereman tidak ada.
4. Interrelasi ACC dan pengemudi; menghasilkan output simulasi program, yang mengindikasikan perintah pengemudi lebih prioritas dibandingkan perintah ACC. Nilai

akselerasi yang diharapkan 120 km/jam dan nilai pengereman tidak ada.

5. Reaksi ACC jika sensor rusak; menghasilkan output simulasi program yang menunjukan kegagalan penerimaan sinyal informasi jarak, ACC akan memberikan sinyal peringatan berupa peringatan visual dengan pesan ACC OFF (ERR).

DAFTAR PUSTAKA

[1] M, Heisel and H, Dennis, "*A Model-Based Development Process for Embedded System*" Universit at Duisburg-Essen, Fachbereich Ingenieurwissenschaften, Institut f ur Medientechnik und Software-Engineering, Germany,2005

[2] www.legalitas.org, tentang "*Peraturan pemerintah republik Indonesia Nomor 44 tahun 1993 Tentang Kendaraan dan pengemudi*", Jakarta, 1993.

[3] H, Kazi,T, A., Stanton, N, A., Harrison, D. "*The interaction between drivers conceptual models of automatic-cruise -control and level of trust in the system* ". 2004. Brunel University, Design and Systems Engineering.

[4] Pressman, Roger S., *Software Enggineering: A practitioners Approach* (sixth ed.), McGraw-Hill, Newyork, 2001.

[5] Laurent Doldi: UML 2.0 Illustrated. TMSO, 2003. http://www.tmso-systems.com

[6] M. Jeckle, C. Rupp, J. Hahn, B. Zengler, S. Queins: UML2 glasklar. Hanser, 2004.

[7] B. Aldrich, "*Using model coverage analysis to improve the controls development process*," AIAA 2002

[8] Gomaa, Hassan., *Designing Concurrent, Distributed, And Real-Time Applications With UML*, Addison-Wesley, Newyork, 2001

[9] Jacobson,Ivar., Boosch, Grady., Rumbaugh, James., *The Unified Software Development Process*, (UML series ed) Addison Wesley, 1999.

[10] Unified Modeling Language Specification, Object Management Group, www.omg.org, 1999.

[11] Architecture and Design: Unified Modeling Language (UML), [http://www.cetus-links.org/oo_uml.html]

[12] Darwiyanti, Sri.,Wahono, Satria Romi,. *Pengantar UML*, www.ilmukomputer.com

[13] Suprapti, Iswanti, "Perancangan Mula Sistem Informasi Penerimaan Siswa Baru Online Dengan Unified

Modelling Language (UML)", thesis magister, ITB, Bandung, 2004

[14] Maritta Heisel, Prof., Dr., "*Development Process (DePES) For Embedded Systems*" Departement of Computer Science Software Engineering. Duisburg Essen Univeritat Germany.2005

[15] Hatebur, Denis, "*A Pattern and Component-Based Process for Embedded Systems Development*" Master thesis, Duisburg Essen, Universitat German.

[16] Siallagan, Sariadin.,"*Pemograman Java, Dasar-dasar pengenalan dan pemahaman*" Andi, Yogyakarta: 2009