PENGUKURAN KECEPATAN PUTAR BERBASIS REAL TIME LINUX

SKRIPSI

Oleh

HERMIN KOSASIH
04 04 03 04 66

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
GANJIL 2007/2008

PENGUKURAN KECEPATAN PUTAR BERBASIS REAL TIME LINUX

SKRIPSI

Oleh

HERMIN KOSASIH
04 04 03 04 66

SKRIPSI INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN

PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
GANJIL 2007/2008

PERNYATAAN KEASLIAN SKRIPSI

Saya menyatakan dengan sesungguhnya bahwa Skripsi dengan judul:

PENGUKURAN KECEPATAN PUTAR BERBASIS REAL TIME LINUX

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari skripsi yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 11 Juli 2008

(Hermin Kosasih)

NPM: 04 04 03 04 66

PENGESAHAN

Skripsi dengan judul:

PENGUKURAN KECEPATAN PUTAR BERBASIS REAL TIME LINUX

Dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada program studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan dalam sidang ujian skripsi.

Depok, 11 Juli 2008

Dosen Pemimbing,

(Dr. Ir. Feri Yusivar, M.Eng.)

NIP.132 090 912

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada :

Dr. Ir. Feri Yusivar, M. Eng

selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberi pengarahan, diskusi dan bimbingan serta persetujuan sehingga skripsi ini dapat selesai dengan baik.

Hermin Kosasih
NPM : 04 04 03 04 66
Departemen Teknik Elektro

Dosen Pembimbing
Dr. Ir. Feri Yusivar, M.Eng.

PENGUKURAN KECEPATAN PUTAR BERBASIS REAL TIME LINUX

ABSTRAK

Program *real time* lebih unggul dibandingkan dengan program yang non *real time* karena memiliki sifat *preemptive* dan *deterministic*. Karena kemampuan program *real time* yang dapat menentukan prioritas pekerjaan dan secara tepat dapat menentukan waktu suatu pekerjaan dilaksanakan maka program *real time* ini cocok diterapkan dalam aplikasi pengukuran.

Tulisan ini membahas perancangan dan penerapan pengukuran kecepatan putar berbasis *real time* Linux dengan menggunakan metode M sebagai metode pengukurannya. Pengukuran kecepatan putar ini dijalankan dalam sistem operasi RTLinux dan peralatan pengukur yang digunakan adalah *incremental rotary* encoder yang beresolusi 100 dan kartu akusisi data NI PCI 6024E. Sistem yang akan diukur kecepatan putarnya adalah sebuah motor DC merek Pittman Express model GM87 12-21.

Pengukuran kecepatan putar berbasis RTLinux ini terdiri dari tiga buah program yang saling berinteraksi satu sama lain. Tujuan keseluruhan dari ketiga program tersebut adalah melakukan perhitungan kecepatan putar dan menampilkan hasil perhitungan kecepatan putar dan posisi sudut dalam bentuk grafik. Selain itu, program tersebut dapat menyimpan nilai hasil perhitungan kecepatan putar dan posisi sudut dalam fungsi waktu ke dalam suatu *file*.

Berdasarkan hasil pengujian yang dilakukan pada pengukuran kecepatan putar berbasis RTLinux ini, dapat disimpulkan bahwa pengukuran kecepatan putar tersebut dapat diandalkan dalam mengukur kecepatan putar suatu motor. Pengukuran kecepatan putar berbasis RTLinux memiliki persen kesalahan pengukuran sebesar 6,26%. Pengujian ini dilakukan dengan membandingkan hasil pengukuran kecepatan putar oleh pengukuran berbasis RTLinux dengan pengukuran oleh tachometer. Dengan menggunakan metode perhitungan *least-squares regression*, diperoleh koefisien determinasi sebesar 99,999% untuk model regresi linear hubungan antara pengukuran kecepatan putar berbasis RTLinux dengan pengukuran kecepatan putar oleh tachometer. Ini membuktikan bahwa hasil pengukuran kecepatan putar berbasis RTLinux linear terhadap hasil pengukuran kecepatan putar oleh tachometer.

Kata kunci : Pengukuran Kecepatan Putar, Metode M, *Real Time* Linux

Hermin Kosasih
NPM : 04 04 03 04 66
Departement of Electrical Engineering

Supervisor
Dr. Ir. Feri Yusivar, M.Eng.

SPEED MEASUREMENT BASED ON REAL TIME LINUX

ABSTRACT

Real time program has become more powerful than non real time program because of its preemptive and deterministic characteristic. Hence the ability of real time program in deciding the priorities of works and accurately performing the specific work on the precise timing, real time program is suitable for the implementation of measurement applications.

This final assignment studied about the design and implementation of speed measurement based on real time Linux by using M Method as its measuring method. The speed measurement is worked on RTLinux operating system and its measuring device is a 100-resolution incremental rotary encoder and a data acquisition board NI PCI 6024E. This speed measurement will measure the speed of DC motor Pittman Express model GM87 12-21.

The RTLinux speed measurement consists of three programs which are interacting each other. The purpose of the whole three programs is to perform the calculation of speed and put the data of speed and pulse position on graph. Besides, those programs also save the data of speed and pulse position in the time domain to a file.

According to the experimental results, it is concluded that the RTLinux speed measurement is reliable on measuring the speed of motor. The RTLinux speed measurement has the small percentage of error in measurement which is 6.26%. This experiment is performed by comparing the results of the RTLinux speed measurement and the result of tachometer speed measurement. By using least-square regression method, the coefficient of determination is 99.999% for the linear model of relationship between RTLinux speed measurement and tachometer speed measurement. It is proved that the result of RTLinux speed measurement is linear to the result of tachometer speed measurement.

Keywords : *Speed Measurement*, M *Method*, *Real Time* Linux

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN

AI	Analog Input
AO	Analog Output
API	*Application Programming Interface*
COMEDI	*COntrol and MEasurement Device Interface*
CPU	*Central Processing Unit*
CTRSRC	*Counter Source*
DAQ	*Data Acquisition*
DIO	Digital Input Output
DMA	*Direct Memory Access*
GCC	*GNU C Compiler*
GPCT	*General-purpose Counter Timer*
GPCTR	*General-purpose Counter*
GTK+	*GNU Image Manipulation Program Toolkit*
I/O	Input/Output
NI	*National Instruments*
MITE	MXI *Interface to Everything*
PC	*Personal Computer*
PCI	*Peripheral Component Interconnect*
PFI	*Programmable Function Input*
POSIX	*Portable Operating System Interface*
RTLinux	*Real Time Linux*
RTSI	*Real-Time System Integration*
STC	*System Timing Controller*
TC	*Terminal Count*
TTL	*Transistor-transistor Logic*

BAB I

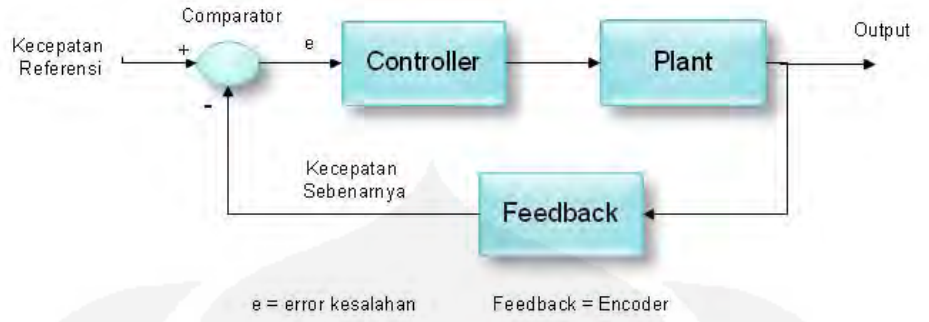
PENDAHULUAN

1.1 LATAR BELAKANG

Dewasa ini, program yang berkemampuan *real time* semakin banyak diminati dan dibutuhkan dalam bidang *manufacture*. Hal ini karena program *real time* lebih unggul dibandingkan dengan program yang non *real time* karena memiliki sifat *preemptive* dan *deterministic*. Program memiliki sifat *preemptive* yang berarti kemampuan program untuk menentukan prioritas-prioritas dari berbagai pekerjaan yang akan dikerjakannya. Program memiliki sifat *deterministic* berarti kemampuan program untuk memprediksi secara tepat waktu suatu pekerjaan dilaksanakan.

Pada skripsi ini digunakan sistem operasi yang dapat menunjang program yang bersifat *real time*. Sistem operasi yang digunakan tersebut bernama RTLinux. Selain sistem operasi RTLinux memiliki kelebihan dalam menunjang program yang bersifat *real time*, sistem operasi tersebut juga merupakan *free software* atau dengan kata lain *software* ini bebas digunakan tanpa perlu membayar/membeli lisensi atau royalti. Hal ini dapat menjadi pertimbangan yang menarik bagi kalangan perusahaan industri untuk mengurangi biaya belanja perangkat lunak.

Pada skripsi ini diterapkan contoh nyata penggunaan program *real time* dalam pengukuran kecepatan putar dari sebuah motor DC. Program pengukuran kecepatan putar berbasis *real time* Linux ini dijalankan dalam lingkungan RTLinux, dan melakukan perhitungan kecepatan putar dengan bantuan instrumen *incremental rotary encoder* yang beresolusi 100 dan kartu DAQ NI PCI 6024E. Metode pengukuran kecepatan yang diterapkan dalam skripsi ini adalah metode M. Prinsip dasar pengukuran kecepatan dengan metode M adalah membagi perubahan posisi sudut dengan selisih waktu pengukuran. Selisih waktu pengukuran pada metode M merupakan waktu deteksi pengukuran. Bila dilihat dari suatu blok diagram sistem *plant* yang memiliki pengendali kecepatan, maka bagian yang dibahas dalam skripsi ini meliputi hanya bagian *feedback*.



Gambar 1.1 Blok Diagram Pengendali Kecepatan Sistem Tertutup

1.2 TUJUAN

Skripsi ini bertujuan merancang dan menerapkan pengukuran kecepatan putar berbasis *real time* Linux dengan menggunakan metode M sebagai metode pengukurannya. Pengukuran kecepatan putar ini dijalankan dalam sistem operasi RTLinux dan peralatan pengukur yang digunakan adalah *incremental rotary encoder* beresolusi 100 dan kartu akusisi data (*Data Acquisition Board*) NI PCI 6024E. Sistem yang akan diukur kecepatan putarnya adalah sebuah motor DC merek Pittman Express model GM87 12-21. Pada skripsi ini dilakukan beberapa skenario percobaan untuk menguji keandalan dan keakuratan dari pengukuran kecepatan putar berbasis *real time* Linux. Dalam pengujian tersebut, dilakukan perbandingan dan analisa data hasil pengukuran kecepatan putar berbasis *real time* Linux dengan data hasil pengukuran kecepatan putar oleh tachometer.

1.3 BATASAN MASALAH

Batasan masalah penulisan ini sebagai berikut:

a. Pembahasan cara kerja *incremental rotary encoder* dan DAQ Board NI PCI 6024E terbatas pada pengukuran kecepatan putar berbasis *real time* Linux.
b. Tidak membahas API (*Application Programming Interface*) RTLinux, COMEDI dan GTK+ pada program pengukuran kecepatan putar berbasis *real time* Linux ini. Tulisan ini hanya membahas algoritma yang penting saja dari program yang digunakan untuk pengukuran kecepatan putar berbasis *real time* Linux.

c. Cara-cara yang digunakan dalam meng-*compile* program pengukuran kecepatan putar berbasis *real time* Linux tidak dibahas dalam penulisan ini. Selain itu, cara-cara memodifikasi *driver* COMEDI dan peng-*compile*-an ulang *driver* COMEDI tersebut juga tidak dibahas dalam penulisan ini.

1.4 SISTEMATIKA PENELITIAN

Pengukuran kecepatan putar berbasis RTLinux yang digunakan dalam skripsi ini akan dievaluasi unjuk kerjanya dalam mengukur kecepatan putar suatu motor DC. Dalam proses evaluasi tersebut, akan dilakukan tiga skenario percobaan untuk mengetahui keakuratan data hasil pengukuran kecepatan putar berbasis *real time* Linux ini dan menguji kemampuan algoritma pengukuran kecepatan putar berbasis *real time* Linux dalam menangani berbagai metode perhitungan yang dilakukan oleh counter seperti *count up* dan *count down*.

1.5 SISTEMATIKA PENULISAN

Untuk membuat pembahasan dalam laporan skripsi ini lebih sistematis, maka laporan skripsi ini akan dibagi dalam beberapa bab yang memiliki pokok bahasan masing–masing untuk mencapai tujuan laporan skripsi ini.

Bab satu adalah pendahuluan yang berisi latar belakang, tujuan, pembatasan masalah, sistematika penelitian dan sistematika penulisan dari laporan skripsi ini. Bab dua ini berisi tentang penjelasan teori-teori yang terpakai dalam skripsi ini yaitu konsep sistem operasi RTLinux, karakteristik kartu data *acquisition* terutama di dalam fungsi *counter*-nya, pembahasan singkat mengenai COMEDI dan juga mengenai instrumen *incremental rotary encoder*, serta pengenalan GTK+. Bab ketiga akan membahas mengenai perancangan pengukuran kecepatan putar berbasis RTLinux yang meliputi langkah-langkah kerja perancangan *hardware* dan *software*-nya. Kemudian, bab keempat berisi pengujian dan analisa data hasil pengukuran kecepatan putar berbasis RTLinux dalam dua skenario percobaan. Bab kelima memuat kesimpulan dari keseluruhan pembahasan pada skripsi ini.

BAB II

LANDASAN TEORI

2.1 REAL TIME LINUX

Sistem operasi merupakan hal yang sangat penting dalam menyediakan sarana hubungan antara perangkat lunak dengan perangkat keras. Sebuah sistem operasi menyediakan sebuah lapisan *abstraction* di antara *platform* perangkat keras dan perangkat lunak dengan menggunakan *interface* yang terdefinisi dengan baik di antara sebuah program *user space, kernel space driver* dan perangkat keras.

Dalam domain komputasi, *kernel* adalah inti dari sistem operasi. *Kernel* ini dikelilingi oleh lapisan *software*, menyediakan hak akses pengguna dan memfasilitas interaksi seperti *shell, window manager*, dan program aplikasi. Untuk menjalankan program *real time*, maka diperlukan sistem operasi yang mana telah mengalami perubahan yang khusus pada *kernel* standarnya sehingga memiliki sifat yang *real time*. Oleh karena itu, diperkenalkan sistem operasi yang berbasis Linux dan bersifat *real time*. Sistem operasi *real time* Linux memiliki keuntungan lain seperti bebas untuk dimodifikasi bagian *kernel* sistem operasi tersebut tanpa perlu khawatir melanggar hak cipta dan juga memudahkan para *developer* dalam mencari informasi yang berhubungan dengan pengembangan *kernel* tersebut.

Linux dikembangkan pertama kali oleh Linus Torvalds di University of Helsinki sebagai bagian dari tugas proyek mahasiswanya. Versi pertama dari *kernel* Linux pertama kali dirilis pada bulan September 1991. Saat ini *kernel* Linux telah banyak dikembangkan oleh para *developer* di seluruh dunia. Sistem operasi *real time* Linux adalah jawaban bagi para pengguna aplikasi yang membutuhkan kemampuan *real time*.

2.1.1 Definisi Real Time

Sebuah sistem *real time* adalah suatu sistem yang menjamin kebenaran dari suatu komputasi perhitungan dan ketepatan waktu pelaksanaan komputasi

perhitungan tersebut. Jika kebutuhan akan ketepatan waktu tidak dapat terpenuhi, kesalahan pada sistem dapat terjadi. Dengan kata lain, sistem *real time* tidak hanya memperhatikan apakah hasil yang dihasilkan dari suatu proses benar, tetapi juga memperhatikan kapan hasil tersebut dihasilkan.

Dilihat dari definisi *real time* tersebut, sebuah sistem *real time* tidak hanya harus cepat, sebagaimana dipercaya oleh sebagian besar orang. Sebagai contoh untuk penuntun sistem dari sebuah kapal dapat saja dikatakan sebagai suatu sistem yang non *real time*. Akan tetapi, karena kecepatan kapal yang rendah, maka memungkinkan memiliki waktu yang cukup bagi sistem untuk mengambil keputusan kontrol. Menurut definisi dari kata *real time* yang dijelaskan sebelumnya, sistem ini dapat secara efektif dikatakan sebagai suatu sistem *real time*. Jadi perlu diketahui bahwa sebuah sistem *real time* adalah bukan sebuah sistem dalam *real time*. Sistem dalam *real time* secara umum adalah sistem yang memiliki kemampuan cepat demi memberikan kesan realitas misalnya pada aplikasi *game* simulasi.

Untuk mendefinisikan secara spesifik dari tingkatan yang berbeda-beda dari *real time*, hal ini sangat penting dalam mendefinisikan terlebih dahulu dua sifat sistem *real time* yaitu, *preemptive* dan *deterministic*.

- *Preemptive* berhubungan dengan kemampuan menentukan sebuah pekerjaan yang berprioritas tinggi untuk dilaksanakan terlebih dahulu dengan mencegah/menunda pelaksanaan pekerjaan yang berprioritas rendah, sehingga *resource* dapat tersedia.
- *Deterministic* berhubungan dengan kemampuan untuk memprediksi kapan sebuah kejadian yang spesifik akan muncul pada waktu yang tepat.

2.1.2 Soft Real Time dan Hard Real Time

Soft real time adalah sebuah respons yang tidak mementingkan seberapa tepat waktu responsnya terhadap suatu intstruksi yang diberikan kepadanya. *Soft real time* tidak memiliki sifat *preemptive* dan *deterministic*.

Hard real time membutuhkan sebuah respons yang bersifat *preemptive* dan *deterministic* terhadap suatu instruksi. Dalam sebuah sistem *hard real time*,

batas waktunya sudah tetap dan sistem harus menjamin responsnya dalam batasan waktu yang telah ditentukan.

2.1.3 Gambaran Umum Kernel Linux

Arsitektur sistem operasi Linux mempartisi memori fisik ke dalam dua bagian yaitu *kernel space* dan *user space*, di mana *kernel space* dialokasikan untuk kode *kernel* Linux. *Kernel* Linux mengatur alokasi memori untuk dipakai oleh program pengguna yang berjalan secara bersamaan di dalam *user space*. Komunikasi di antara program *user space* dengan kode *kernel* dapat dicapai dengan menggunakan *system call* ke kode *kernel*. Secara umum contoh dari penggunaan *system call* adalah ketika saat mengakses *disk drive, keyboard, mice* dan *monitor*. Jadi kode *kernel* berfungsi dalam menjembatani antara aplikasi program pengguna dengan perangkat keras.

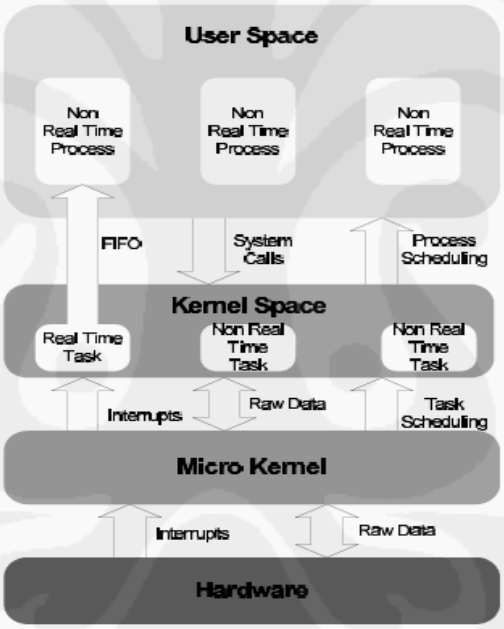
Sistem operasi Linux merupakan sistem operasi yang *multi-threaded* yaitu mendukung prioritas *thread* dan menyediakan mekanisme sinkronisasi *thread* yang dapat diprediksi. *Kernel* Linux memiliki sifat yang tidak *preemptible*. Hal ini yang menyebabkan kecepatan waktu respons operasi *kernel* pada linux yang standar lebih lambat bila dibandingkan dengan sistem operasi *hard real time*. Perlu diketahui bahwa Linux bukan merupakan sistem operasi yang *real time* karena ketidak-mampuannya dalam menjamin kinerja yang *deterministic* dan pewaktuannya yang bersifat pukul rata sehingga memiliki dampak yang buruk terhadap program *real-time*.

Ada dua alasan yang menyatakan bahwa *kernel* Linux memiliki kinerja yang buruk pada sistem *uniprosesor* karena *kernel* menon-aktifkan *interrupt* dan juga kernel tidak cocok dalam melakukan aktivitas *preemption*. Bila *interrupt* dinon-aktifkan, sistem tidak berdaya dalam menjawab terhadap *interrupt* yang datang. Semakin lama penundaan *interrupt*, maka semakin lama penundaan waktu dari suatu program dalam menjawab *interrupt* yang diberikannya. Kurangnya kemampuan *preemption* pada *kernel* memiliki arti bahwa *kernel* tidak mampu memberikan fasilitas kepada tugas yang berprioritas tinggi untuk dijalankan terlebih dahulu dan menunda tugas yang berprioritas rendah. Hal ini dapat menyebabkan waktu penundaan yang cukup berarti.

2.1.4 Sejarah Singkat Real Time Linux (RTLinux)

RTLinux dikembangkan pertama kali oleh Victor Yodaiken di *Department of Computer Science of the Institute for Mining and Technology of New Mexico* dan diimplementasi pertama kali oleh Michael Barabanov pada tahun 1996. Dengan mengambil konsep untuk tidak memodifikasi *kernel* dari Linux secara ekstrim dalam membuat suatu sistem operasi yang *real time*, maka diajukan suatu solusi dengan membuat *kernel-kernel* kecil yang terpisah dari *kernel* linux dan sebuah *scheduler*. *Kernel* Linux yang dihasilkan dari strategi modifikasi ini disebut sebagai *micro kernel*.

Implementasi dengan strategi ini menyediakan sebuah *kernel* kedua yang mana sebagai lapisan *interface* antara *kernel* standar dan lapisan perangkat keras sebagaimana ditampilkan dalam gambar 2.1.

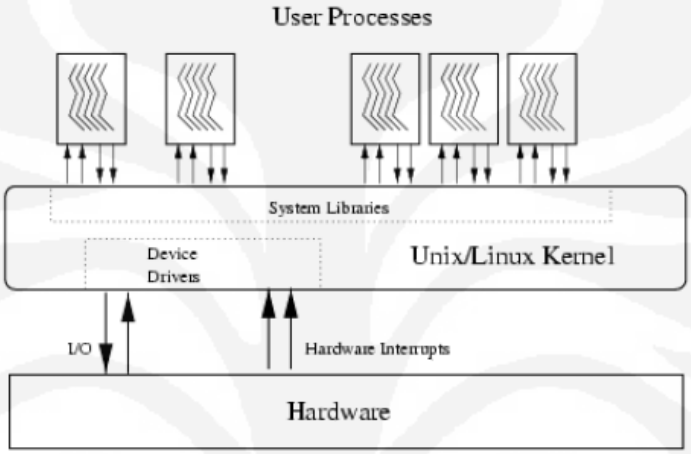


Gambar 2.1 Arsitektur Real Time Micro Kernel [4]

2.1.5 Konsep Dasar Real Time Linux

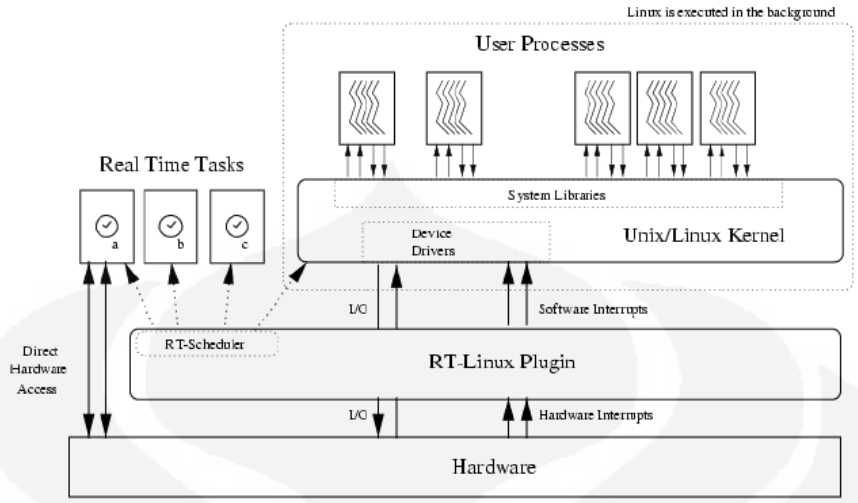
Dasar pemikiran dalam mendesain RTLinux adalah tidak ada kemungkinan untuk mengidentifikasi dan menghilangkan seluruh aspek dari operasi *kernel* yang memunculkan hal-hal yang tidak dapat diprediksi. Sumber-sumber dari hal-hal yang tidak dapat diprediksi terdiri dari algoritma *Linux scheduling, device driver, uninterruptible system call,* penggunaan dari *interrupt disabling* dan operasi *virtual memory*.

Hal terbaik dalam menghindari permasalahan ini adalah membentuk kernel kecil (*micro kernel*) yang dapat diprediksi dan terpisah dari *kernel* linux. Dengan membuat *micro kernel* ini secara sederhana, maka operasinya dapat diukur dan dapat diprediksi. *Micro kernel* ini memiliki fungsi mengontrol eksekusi tugas-tugas *real time* dan menjalankan *kernel* Linux standar sebagai tugas yang berjalan dalam mode *background*.



Gambar 2.2 Kernel Linux Standar [3]

Gambar 2.2 menunjukkan *kernel* Linux standar tanpa dukungan *hard real time*. Dalam gambar 2.2 ditunjukkan bahwa *kernel* Linux memisah *hardware* dari *user-level task*. *Kernel* memiliki kemampuan dalam menunda *user-level task* apapun, bilamana *task* tersebut telah melebihi penggunaan waktu yang dialokasikan kepadanya oleh CPU. Asumsi, sebagai contoh, sebuah *user task* yang mengontrol sebuah tangan robot. *Kernel* linux yang standar akan berpotensi menunda tugas pengontrolan tangan robot dan memberikan alokasi CPU ke tugas-tugas lain yang kurang genting. Konsekuensinya, tangan robot tersebut tidak dapat memenuhi suatu kriteria waktu yang ketat yang telah ditentukan. Oleh karena itu, dalam mencoba berlaku adil terhadap semua *task*, *kernel* tersebut telah menghambat tugas (*task*) yang genting.



Gambar 2.3 Kernel RTLinux [3]

Gambar 2.3 menunjukkan sebuah *kernel* linux yang dimodifikasi untuk mendukung *hard real time*. RTLinux berbentuk sebagai sebuah lapisan tambahan berada di antara *kernel* Linux standar dan perangkat keras komputer. Dalam pandangan *kernel* Linux standar, lapisan baru ini muncul sebagai perangkat keras yang sebenarnya. RTLinux memperkenalkan *fixed-priority* pada *scheduler*-nya sendiri. *Scheduler* (penjadwalan) menentukan prioritas terendah untuk *kernel* Linux standar dan menjalankannya sebagai tugas yang terpisah dengan tugas-tugas *real time*. Hal ini menyebabkan *micro kernel* mampu menginterupsi perangkat keras dan menjamin *kernel* Linux standar tidak menunda interupsi apapun yang bekerja dalam *micro kernel* sehingga waktu penundaan yang dapat terjadi pada tugas-tugas *real time* dapat diminimalisir.

2.2 COMEDI

COMEDI, singkatan dari *Control and Measurement Device Interface*, adalah suatu proyek *free software* yang mengembangkan *driver-driver*, *tool* dan *library* untuk berbagai jenis DAQ (*Data Acquisition*) *board*. Dengan menggunakan COMEDI, maka DAQ *board* dapat digunakan dalam melakukan aktivitas seperti membaca dan menulis sinyal analog, membaca dan menulis masukan/keluaran digital, melakukan proses *encoder* dan sebagainya. COMEDI terdiri dari dua paket yaitu paket “comedi” (yang mengimplementasi fungsi *kernel space*) dan “comedilib” (yang mengimplementasi akses *user space* ke fungsi

device driver). COMEDI dapat berjalan pada *kernel* Linux standar dan juga pada RTLinux. Berikut ini adalah macam-macam paket yang didistribusi oleh pihak COMEDI:

a. Paket Comedi merupakan kumpulan *driver-driver* untuk berbagai macam kartu data akusisi.
b. Paket Comedilib merupakan paket distribusi yang terdiri dari sebuah *user-space library* yang menyediakan antar muka bagi Comedi *device*. Hal-hal lain yang turut dicantumkan dalam paket Comedilib adalah dokumentasi, fasilitas kalibrasi dan program-program demo.
c. Paket Kcomedilib adalah sebuah modul kernel Linux (didistribusi bersamaan dengan paket comedi) yang menyediakan antar muka yang sama dengan comedilib dalam *kernel space*, dan cocok diterapkan untuk program *real time*. Jadi kcomedilib adalah *kernel library* untuk penggunaan Comedi pada program *real time*.

2.2.1 Definisi Device Driver

Sebuah kartu DAQ dapat diakses oleh program tidak terlepas karena adanya *driver* atau *device driver*. *Device driver* adalah sekumpulan perangkat lunak yang menjembatani ke sekumpulan perangkat keras misalnya *printer*, *sound card* atau *motor drive*. *Device driver* berfungsi menerjemahkan perintah-perintah yang diberikan dari perangkat keras untuk dapat dikonfigurasi, dibaca dan diolah ke dalam fungsi-fungsi umum dan struktur data untuk program.

2.2.2 Hirarki Device

COMEDI mengorganisasi semua perangkat keras berdasarkan hirarki umum berikut ini:

- **Channel**: Komponen perangkat keras *low level*, yang mewakili suatu saluran data tunggal. Misalnya sebuah masukan analog atau sebuah masukan digital. Setiap *channel* memiliki beberapa parameter seperti *voltage range, reference voltage,* dan *channel polarity* (unipolar,bipolar).
- **Sub-device**: sekumpulan *channel* yang memiliki fungsi yang sama yang secara fisik diimplementasi pada kartu *interface* yang sama.

- **Device**: sekumpulan *sub-device* yang secara fisik diimplementasi pada kartu *interface* yang sama. Sebagai contoh, peralatan "National Instruments PCI 6024E" memiliki empat *sub-device* yang terdiri dari 16 buah analog input channel, dua buah analog output channel, 8 buah digital input/output channel dan dua buah *counter*.

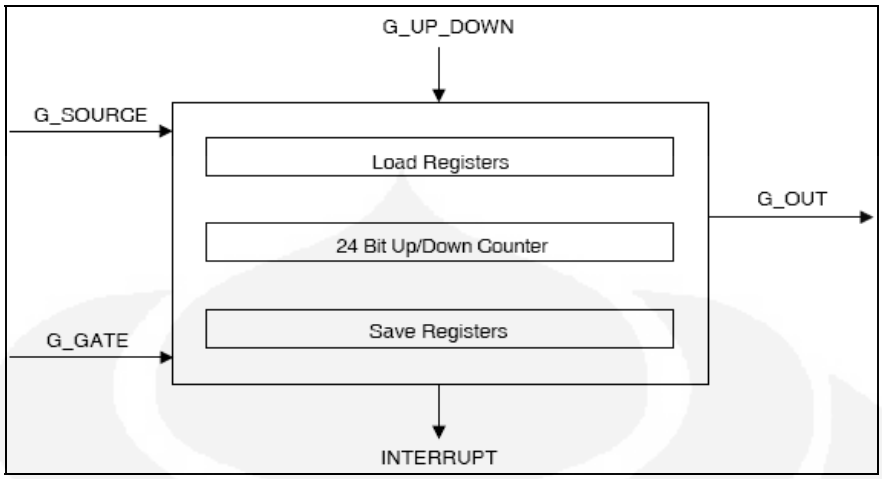
2.3 DATA ACQUISITION BOARD

Data acquisition board merupakan instrumen yang dipakai dalam melakukan suatu proses pengukuran terhadap suatu kejadian, misalnya mengukur tegangan pada sensor temperatur. Secara umum, DAQ *board* melakukan kegiatan mendigitalisasi signal masukan analog, melakukan konversi dari sinyal digital ke analog, mengukur dan mengontrol sinyal digital input atau output. Sebagai informasi tambahan, untuk kartu DAQ NI PCI 6024E menggunakan sistem *timing controller* dari National Instruments untuk fungsi yang berhubungan dengan waktu dan terdiri dari tiga kelompok *timing* yaitu mengontrol masukan analog, keluaran analog dan fungsi *general-purpose counter*/*timer*.

2.3.1 Konsep Dasar General Purpose Counter/Timer

General-purpose counter/*timer* (GPCT) adalah counter yang memiliki kemampuan untuk menghitung secara *count up* ataupun secara *count down*, memiliki *load* dan *save register*, memiliki sebuah struktur pengontrol untuk mengimplementasi beberapa perhitungan umum dan memiliki fungsi *timing* I/O. Fungsi *timing* ini terdiri pengukuran periode, dan *pulse-train generation* dengan frekuensi dan *duty cycle* yang dapat diprogram. Kebanyakan fungsi dapat dioperasi dengan menggunakan hanya satu buah *general-purpose counter*. Ada dua mode operasi pengukuran pada kartu NI PCI 6024E yaitu *single mode* dan *buffered mode*. Dalam *single mode*, counter hanya melakukan satu buah pengukuran. Dalam *buffered mode*, counter melakukan serangkaian pengukuran yang berurutan.

Modul GPCT memiliki dua buah *counter* 24-bit *binary up*/*down* yang identik dengan *general-purpose counters* 0 and 1. Gambar 2.4 menunjukkan model sederhana dari sebuah *counter*.



Gambar 2.4 Model Sederhana dari General-Purpose Counter/Timer [2]

Setiap counter GPCT memiliki sebuah input *source* (G_SOURCE), sebuah input *gate* (G_GATE) dan sebuah input *control up*/*down* (G_UP_DOWN). Ketika *counter* diaktifkan untuk menghitung, *rising edges* pada input G_SOURCE dapat menyebabkan counter menambah atau mengurang isi *save register*-nya. Masukan G_GATE berlaku sebagai sebuah sinyal pengontrol *general-purpose* dan dapat dioperasi sebagai sebuah sinyal pen-*trigger* counter, sinyal pengaktif counter, sinyal *save*, sinyal *reload*, sinyal *interrupt*, sinyal kontrol output, sinyal pemilih *register load*, dan sinyal penghenti counter. Input G_UP_DOWN bertujuan menentukan metode perhitungan pada counter yaitu metode perhitungan *count up* (penambahan) atau metode perhitungan *count down* (pengurangan). Keluaran counter adalah sinyal G_OUT dan sinyal *interrupt*. Pin G_OUT berfungsi mengeluarkan sinyal dari modul counter ke port keluaran pada kartu NI PCI 6024E.

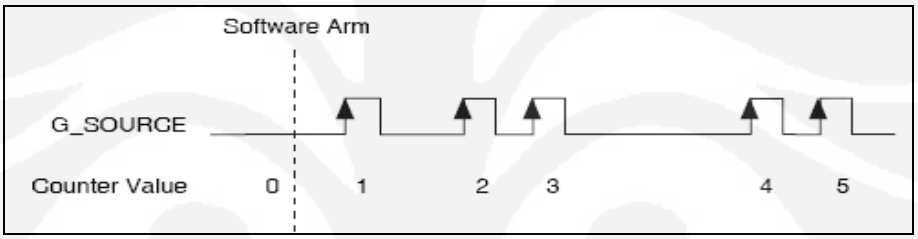
2.3.2 Event-Counting pada GPCT

Dalam *event-counting*, counter menghitung kejadian pada input G_SOURCE setelah counter diaktifkan. Kondisi-kondisi yang dialami oleh counter pada saat *event counting* adalah:

a. Pin G_SOURCE berfungsi menerima pulsa-pulsa informasi.

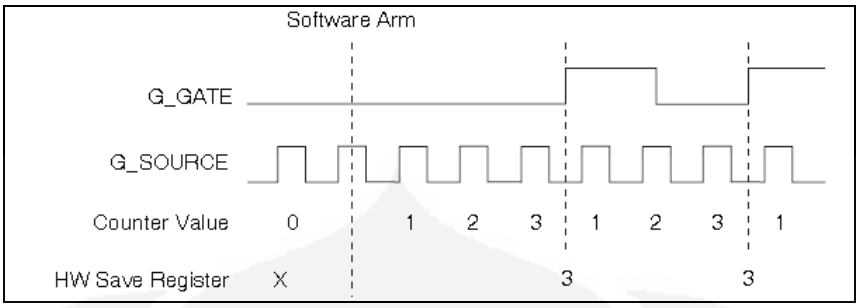
b. Pin G_GATE untuk mengindikasi kapan untuk memulai dan menghentikan interval penghitungan dan kapan untuk menyimpan isi *counter* pada *save register*.
c. *Software* dapat membaca nilai *counter* secara asinkron atau membaca nilai simpanan di *save register* setiap saat.
d. Pin G_UP_DOWN mengontrol metode perhitungan pada counter yaitu penambahan atau pengurangan.

Dalam mode *simple event counting*, counter menghitung jumlah pulsa yang diberikan pada G_SOURCE ketika counter diaktifkan. Software dapat membaca isi counter kapan saja tanpa mengganggu proses perhitungan. Gambar 2.5 menunjukkan sebuah contoh dari *simple event counting* di mana counter menghitung lima kejadian pada G_SOURCE.



Gambar 2.5 Simple Event Counting [2]

Dalam mode *buffered non-cumulative event counting*, counter secara umum memiliki karakteristik yang mirip dengan mode *simple event counting* kecuali adanya interval perhitungan secara berkelipatan. Sinyal G_Gate mengindikasi adanya limit antara serangkaian interval perhitungan. Counter menghitung pulsa yang diberikan pada sinyal G_SOURCE sesudah *software* mengaktifkan counter tersebut. Setiap terjadi *active edge* pada sinyal G_GATE, maka isi counter akan diisi ke *hardware save register* dan *counter* akan mengisi kembali isi counter ke nilai inisialnya untuk memulai interval perhitungan berikutnya. Sebuah *interrupt* akan memberitahukan CPU sesudah setiap interval perhitungan sehingga *interrupt software* dapat membaca hasil yang disimpan dalam *hardware save register*. Gambar 2.6 menunjukkan sebuah contoh perhitungan counter pada mode *buffered non-cumulative event counting*.



Gambar 2.6 Buffered Non-Cumulative Event Counting dengan dua interval [2]

2.4 INSTRUMEN INCREMENTAL ROTARY ENCODER

Instrumen *incremental rotary encoder* atau dikenal juga sebagai *incremental shaft encoder* adalah salah satu tipe dari peralatan *encoder* yang memberikan keluaran dalam format digital. Hal ini menyebabkan instrumen sejenis ini lebih nyaman digunakan pada aplikasi pengontrolan yang menggunakan komputer, sebagaimana pengukuran dibutuhkan dalam bentuk digital. Oleh karena itu, proses konversi dari sinyal analog ke digital tidak perlu lagi dilakukan.

Dewasa ini, instrumen ini banyak digunakan dalam industri terutama pada mesin-mesin seperti mesin pengemasan, tangan robot, pengontrol gerakan motor derek, mesin penggiling. Biasanya instrumen ini digunakan dalam menghitung sudut, posisi, revolusi, kecepatan, akselerasi dan jarak. Ada beberapa macam jenis *incremental rotary encoder* yang telah dikembangkan seperti jenis magnetis, kontak, resistif dan optis. Akan tetapi, jenis *incremental rotary encoder* yang akan dibahas dalam sub bab ini adalah jenis optis saja.

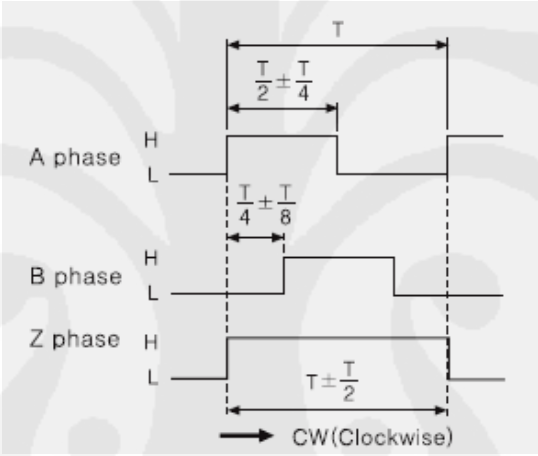
2.4.1 Konsep Dasar Instrumen Incremental Rotary Encoder

Konsep dasar operasi instrumen *incremental rotary encoder* adalah instrumen ini mengukur nilai sesaat posisi angular dari sebuah *shaft* yang sedang berotasi dan menghasilkan pulsa-pulsa pada *channel-channel*-nya. Pulsa-pulsa yang dihasilkan ini berbentuk gelombang *square*.

Instrumen *incremental rotary encoder* biasanya memiliki tiga buah sinyal keluaran, yaitu sinyal A, sinyal B, dan sinyal Z, ditunjukkan dalam gambar 2.7. Untuk sinyal A dan sinyal B, masing-masing sinyal keluaran tersebut saling *quadrature* yang berarti terjadi pergeseran fasa 90^{O} satu sama lain. Kedua sinyal

tersebut selain memberikan nilai posisi *shaft* dari *encoder*, juga mampu menyediakan informasi mengenai arah putaran dari *shaft* misalnya berputar searah jarum jam atau berputar berlawanan arah jarum jam. Hal penting yang perlu diperhatikan hubungan antara sinyal A dan sinyal B adalah bahwa pergeseran fasa satu sama lain antara kedua sinyal tersebut harus berada dalam batas toleransi yang dapat diterima biasanya tidak melebihi 90^{O} sehingga proses perhitungan dapat berlangsung dengan akurat.

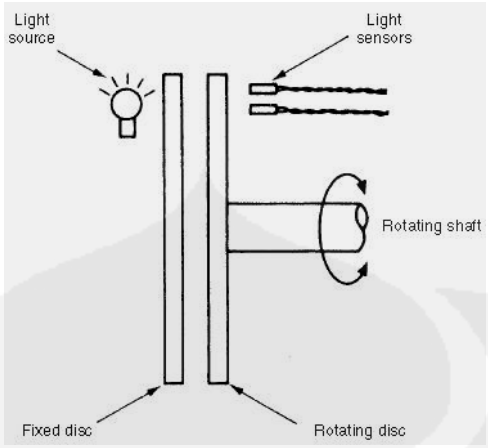
Untuk kebanyakan peralatan mesin motor atau aplikasi *positioning*, sinyal Z dikenal sebagai *index signal*, yang memiliki peranan penting dalam menentukan *zero position* dengan cara memberikan sebuah pulsa keluaran tunggal per satu revolusi.



Gambar 2.7 Tiga Buah Sinyal Keluran Encoder[6]

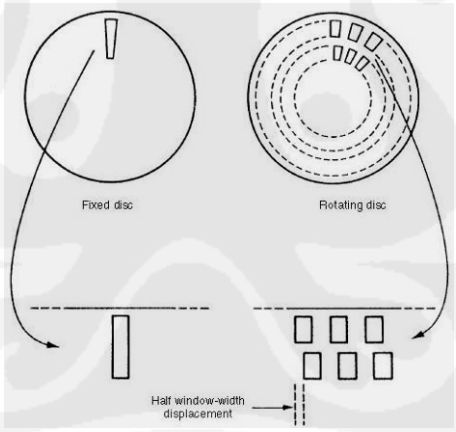
2.4.2 Prinsip Kerja Instrumen Optical Incremental Position Encoder

Sebuah contoh dari sebuah *optical incremental rotary encoder* ditunjukkan dalam gambar 2.8. Pada gambar tersebut, instrumen ini terdiri dari sebuah sumber cahaya yaitu biasanya adalah LED, sebuah *disc encoder* (*rotating disc*), sebuah *fixed disc*, dan *photo-detector*. Letak posisi antara LED dan *photo-detector* disusun dengan sejajar, sehingga cahaya dari LED dapat masuk ke *detector* tersebut secara tegak lurus.



Gambar 2.8 Optical Incremental Shaft Encoder [6]

Disc yang ditunjukkan dalam gambar 2.9 adalah elemen kunci dari *encoder* dan secara umum terbuat dari bahan kaca dan diberi cap yang terbuat dari bahan logam sehingga membentuk slot-slot jendela yang mengelilingi bagian tepi dari *disc encoder* tersebut. Slot-slot ini bersifat tembus cahaya. Jumlah slot tersebut adalah sama dengan jumlah dari pulsa per satu revolusi. Sebagai contoh, sebuah *disc* kaca yang dicap dengan 1000 slot, bila *disc* tersebut telah bergerak 180^{O} jika dan hanya jika encoder tersebut telah mengeluarkan 500 pulsa.



Gambar 2.9 Susunan Jendela dalam Incremental Rotary Encoder [6]

Selama *disc* berotasi, LED secara konstan diaktifkan dan cahaya dari LED tidak secara terus-menerus mencapai *photo-detector*, tetapi hanya saat cahaya tersebut melewati slot-slot yang berada pada *disc* encoder tersebut. Saat cahaya masuk ke *photo-detector*, *detector* tersebut menghasilkan pulsa-pulsa keluaran berupa gelombang *square*.

Kebanyakan dari *incremental rotary encoder* menyediakan sebuah tanda tunggal pada *disc*, disebut *channel* Z atau *marker*. Pulsa dari *channel* ini menyediakan sebuah referensi yang menandakan per satu revolusi.

2.5 GTK+ (GIMP Toolkit)

GTK+ merupakan singkatan dari GIMP Toolkit, asal mulanya GTK+ didesain untuk *raster graphics editor* yang dikenal dengan sebutan GNU Image Manipulation Program (GIMP). GTK+ dibuat pertama kali di tahun 1997 oleh Peter Mattis, Spencer Kimball, dan Josh MacDonald. Dengan berlisensi *Lesser General Public License* (LGPL), GTK+ telah memberikan keuntungan berupa tidak perlunya membayar lisensi atau royalti dalam menggunakan GTK+ untuk penggembangan software baik *open software*, *free software* maupun *software* komersial. Dewasa ini, GTK telah digunakan dalam sejumlah besar proyek *software* termasuk di dalamnya proyek GNU Network Object Model Environment (GNOME), yang saat ini merupakan lingkungan desktop linux yang amat populer.

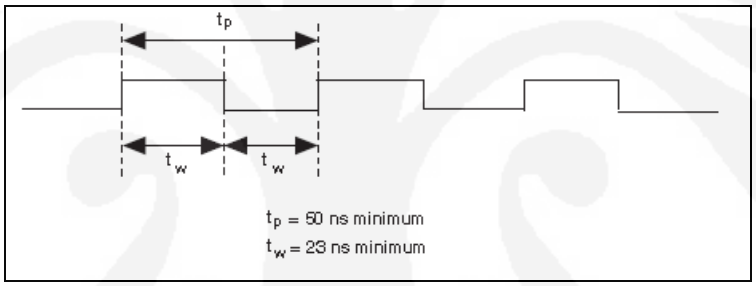
GTK+ adalah sebuah *object-oriented application programming interface* (API) yang ditulis dalam bahasa pemrograman C. GTK+ diimplementasi dengan konsep kelas-kelas dengan tujuan membuat suatu sistem yang luas yang dibangun atas dirinya. GTK selain menyediakan elemen-elemen yang biasa digunakan dalam membuat *interface* suatu program misalnya *button*, *label*, *text box* dan *window*, juga menyediakan komponen-komponen yang lebih abstrak yang digunakan untuk *application layout*.

BAB III

PERANCANGAN PENGUKURAN KECEPATAN PUTAR BERBASIS REAL TIME LINUX

3.1 KONSEP DASAR PENGUKURAN KECEPATAN

Konsep dasar pengukuran kecepatan secara sederhana adalah membagi jarak yang ditempuh dengan waktu tempuh. Dalam perancangan pengukuran kecepatan putar yang menggunakan *incremental rotary encoder*, sangat perlu diperhatikan jangkauan frekuensi masukan yang dapat didukung oleh kartu DAQ. Berikut ini adalah gambar pewaktuan sinyal gerbang *source* pada counter untuk kartu DAQ NI PCI 6024E.



Gambar 3.1 Pewaktuan Sinyal Gerbang Source pada Counter [1]

Frekuensi maksimum yang diizinkan sebagai masukan ke gerbang *source* pada counter di kartu NI PCI 6024E, berdasarkan referensi dari buku manual kartu NI PCI 6024E, sebesar 20MHz, dengan sebuah lebar pulsa minimum sebesar 23ns (*nanoseconds*). Tidak ada batasan frekuensi minimum. Menurut referensi dari *datasheet incremental rotary* encoder tipe E50S, encoder yang digunakan dalam pengukuran kecepatan putar berbasis *real time* Linux ini memiliki frekuensi keluaran maksimum sebesar 300kHz, sehingga dapat digunakan pada kartu NI PCI 6024E.

Selain itu, juga perlu diperhatikan lebar pulsa minimum dari sinyal masukan ke gerbang *gate* pada counter. Lebar pulsa minimum yang diizinkan oleh counter adalah 6ns (*nanoseconds*). Untuk jenis encoder yang digunakan dalam

pengukuran kecepatan putar berbasis *real time* Linux ini, lebar pulsa keluaran encoder tersebut adalah 3,33±1,665 μs (*microseconds*).

Pengukuran kecepatan putar dengan menggunakan *incremental rotary encoder* perlu memperhatikan beberapa hal penting berikut ini:

- Resolusi encoder. Tinggi rendahnya resolusi sebuah encoder yang dibutuhkan disesuaikan dengan kebutuhan dari aplikasi yang hendak diterapkan.
- Kecepatan putar suatu motor. Kecepatan putar suatu motor tidak boleh melebihi maksimum respons frekuensi yang dapat didukung oleh encoder. Hal ini dapat menyebabkan tidak dapat diukurnya keluaran sinyal pulsa dari encoder.
- Frekuensi keluaran encoder. Frekuensi keluaran encoder yang digunakan sebagai masukan pada gerbang-gerbang counter harus di bawah nilai frekuensi masukan yang diizinkan oleh counter agar counter dapat men-*scanning* frekuensi masukan tersebut dengan akurat.

Metode pengukuran yang digunakan pada pengukuran kecepatan putar berbasis *real time* Linux ini tergolong metode yang konvensional, yang mana banyaknya pulsa yang dihasilkan dibagi dengan waktu deteksi. Metode ini dikenal sebagai metode M.

3.1.1 Metode M

Konsep dasar dari metode M ini adalah selisih dari perubahan posisi sudut ($\Delta\theta$) dibagi dengan selisih waktu (Δt). Keluaran dari encoder dimasukkan ke dalam gerbang *source* counter kemudian program akan membaca nilai posisi sudut dari isi counter untuk setiap Δt dan menselisihkan nilai posisi sudut pembacaan saat ini dengan nilai posisi sudut pembacaan sebelumnya ($\Delta\theta$) dan terakhir hasil selisih tersebut dibagi dengan selisih waktu (Δt). Hasil pembagian tersebut akan diperoleh kecepatan sudut, dan berikut ini adalah perumusan dari kecepatan sudut:

$$\omega = \frac{\Delta\theta}{\Delta t} \qquad \text{(rad/s)} \qquad (3.1)$$

dengan ω adalah kecepatan sudut (rad/s), $\Delta\theta$ merupakan perubahan posisi sudut (rad), dan Δt merupakan perubahan waktu (s).

Akan tetapi, hasil kecepatan putar yang diperoleh belum dalam bentuk rpm (*revolutions per minute*) sehingga perlu adanya rumus tambahan untuk mencari kecepatan putar tersebut. Berikut ini adalah rumus untuk kecepatan putar dalam satuan rpm:

$$\theta = \frac{2\pi m}{P} \quad \text{(rad)} \tag{3.2}$$

$$\omega = \frac{2\pi m}{PT_c} \quad \text{(rad/s)} \tag{3.3}$$

atau:

$$\omega = \frac{60m}{PT_c} \quad \text{(rpm)} \tag{3.4}$$

dengan θ adalah posisi sudut (rad), m merupakan banyaknya pulsa yang dihasilkan selama T_c , P merupakan banyaknya pulsa yang dihasilkan oleh encoder dalam satu revolusi, dan T_c merupakan waktu deteksi (s).

3.2 PLATFORM UJICOBA

Dalam perancangan pengukuran kecepatan putar berbasis *real time* Linux, tidak terlepas dari dibutuhkannya dukungan perangkat keras dan perangkat lunak. Perangkat keras dan perangkat lunak sangat menentukan berjalannya pengukuran kecepatan putar berbasis RTLinux dengan baik.

3.2.1 Perangkat Keras yang Diperlukan

Perangkat keras yang diperlukan agar pengukuran kecepatan putar berbasis RTLinux ini dapat berjalan dengan baik yaitu:

- Dua buah PC (*Personal Computer*) dengan spesifikasi sebagai berikut:

Tabel 3.1 Tabel Konfigurasi Peralatan PC

	Yang dipakai di Skripsi
CPU	3 GHz
Hard Drive	15 Gigabytes
RAM	512 Megabytes
Video RAM	128 Megabytes
Keyboard	Ada

Tabel 3.2 Tabel Konfigurasi Peralatan PC (lanjutan)

Mice	Ada
CD-ROM	Ada
Floopy Drive	Ada
PCI Slot yang tidak terpakai	4 buah

- Dua buah DAQ *Board,* di mana DAQ *Board* adalah kartu *Data Acquisition* (yang dipakai dalam skripsi ini adalah National Instruments PCI 6024E). Alasan dipakai kartu NI PCI 6024E ini adalah karena kartu PCI seri E tidak memiliki saklar DIP, *jumper* atau potentiometer, sehingga dapat dengan mudah dikonfigurasi dan dikalibrasi dengan menggunakan *software*. Kartu PCI seri E ini adalah kartu I/O multifungsi analog, digital dan *timing* yang *Plug* and *Play* untuk PCI *bus* pada komputer. Karakteristik kartu seri E ini memiliki enam belas buah AI channel dengan resolusi 12-bit, dua buah AO channel dengan resolusi 12-bit, delapan buah saluran yang TTL-*compatible* DIO, dan dua buah 24-bit *counter*/*timers* untuk *timing* I/O.
- Dua buah konektor I/O 68-pin untuk kartu DAQ.
- Satu buah *incremental rotary encoder* merek Autonics® tipe E50S8-100-3-N-24, yang mana memiliki tiga sinyal keluaran A, B dan Z.
- Dua buah DC Power Supply.
- Rangkaian voltage regulator untuk keluaran DC 5 volt.

Perangkat yang dipakai dalam skenario percobaan pengujian pengukuran kecepatan putar berbasis RTLinux sebagai berikut:

- Satu buah DC Gearmotor merek Pittman Express model GM87 12-21.
- Tachometer merek Lutron Electronic model DT-2236.
- Satu buah DC Power Supply.
- Digital Multimeter merek Kyoritsu tipe Kew Mate model 2000.

3.2.2 Perangkat Lunak yang Diperlukan

Perangkat lunak yang diperlukan dalam pengukuran kecepatan putar berbasis *real time* Linux sebagai berikut:

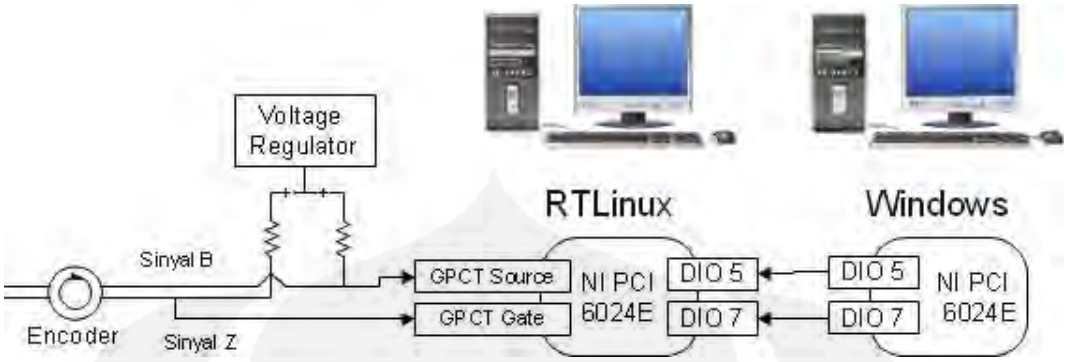
- Sistem operasi Linux dengan distro Redhat 9.0
- *Real Time* Linux (RTLinux) versi 3.2-rc1
- Comedi versi 0.7.73 dan comedilib versi 0.7.22
- Sistem operasi Windows 2000.
- *Software Measurement* & *Automation Explorer* dari National Instruments.
- GTK+ (GIMP Toolkit) versi 2.0.
- GtkDataBox versi 0.4.0.2 adalah *widget* tambahan bagi GTK+ untuk memplot grafik.

3.3 PERANCANGAN PERANGKAT KERAS DAN PERANGKAT LUNAK

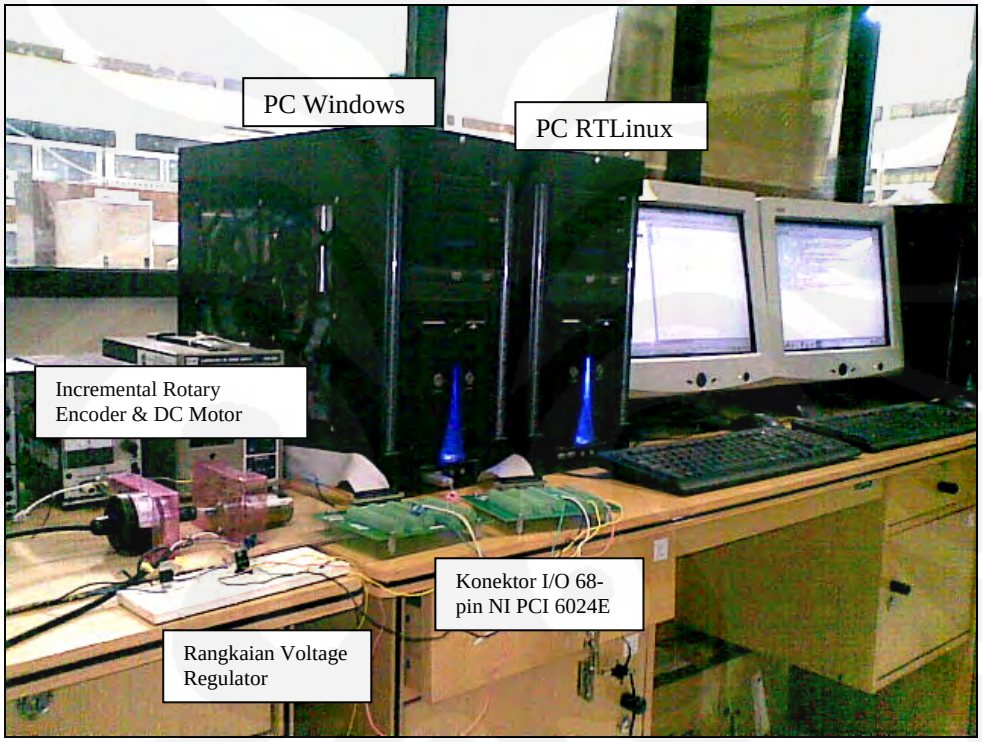
Pengukuran kecepatan putar berbasis RTlinux harus melewati beberapa tahapan pengerjaan sebelum siap diuji coba. Tahapan ini terdiri dari dua bagian yaitu bagian perancangan perangkat keras dan bagian perancangan perangkat lunak pengukuran kecepatan putar berbasis RTLinux.

3.3.1 Perancangan Perangkat Keras Pengukuran Kecepatan Putar Berbasis Real Time Linux

Pengukuran kecepatan putar berbasis *real time* Linux menggunakan dua komputer dengan satu komputer berbasis Windows dan satu komputer lagi berbasis RTLinux. Kedua komputer tersebut terintegrasi dengan kartu NI PCI 6024E. Komputer Windows bertujuan memberikan sinyal *trigger* yang merubah metode perhitungan pada counter dan sinyal *trigger* yang mengaktifkan dan menon-aktifan counter. Counter yang di-*trigger* oleh komputer Windows adalah counter yang berada pada komputer RTLinux. Secara umum bentuk rancangan dari keseluruhan pengukuran kecepatan putar berbasis RTLinux ini ditampilkan dalam gambar 3.2 dan hasil implementasinya ditampilkan dalam gambar 3.3.



Gambar 3.2 Rancangan Pengukuran Kecepatan Putar Berbasis Real Time Linux



Gambar 3.3 Perangkat Ujicoba Pengukuran Kecepatan Putar Berbasis Real Time Linux

Pengukuran kecepatan putar berbasis RTLinux menggunakan 4 buah pin pada counter. Fungsi dari masing-masing pin tersebut sebagai berikut:

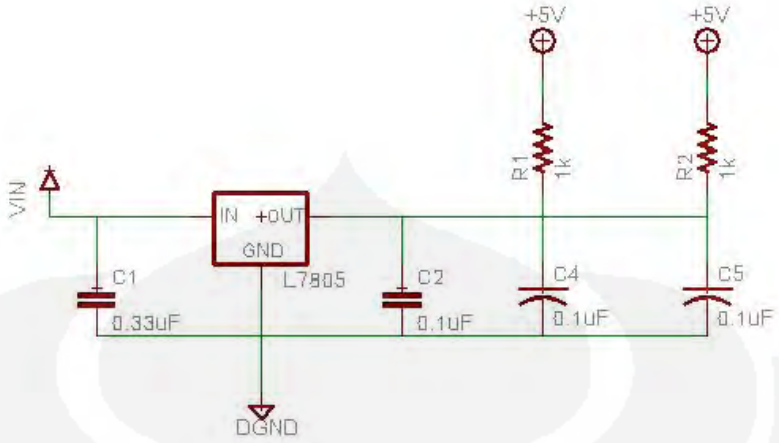
a. Pin GPCT Source. Pin ini bertujuan menerima sinyal dari port B keluaran encoder yang memberikan informasi mengenai peningkatan/penurunan kecepatan putar.
b. Pin GPCT Gate. Pin ini bertujuan menerima sinyal dari port Z keluaran encoder yang bertugas memberikan pemberitahuan telah terjadinya satu revolusi dalam suatu putaran.

c. Pin DIO 5. Pin ini bertujuan menerima sinyal *trigger* untuk mengaktifkan dan menon-aktifkan counter. Bila pulsa yang diterima oleh counter adalah sinyal *up* maka counter tidak dapat diaktifkan. Bila pulsa yang diterima oleh counter adalah sinyal *down* maka counter dapat diaktifkan, dan pengaktifan counter ini terus terjadi selama sinyal *up* tidak diberikan ke counter.
d. Pin DIO 7. Pin ini bertujuan menerima sinyal *trigger* pemberitahuan untuk merubah metode perhitungan pada counter. Informasi yang diterima berupa sinyal *up* atau sinyal *down*. Sinyal *up* berarti counter melakukan perhitungan secara menambah, sedangkan sinyal *down* berarti counter melakukan perhitungan secara mengurang.

Pada perangkat komputer Windows tersebut di-*install software Measurement* & *Automation Explorer* dari National Instruments. Di dalam bagian *software* ini, terdapat bagian yang bertujuan mengatur keluaran sinyal pada pin-pin DIO di kartu DAQ yang dipasang di komputer tersebut. Pin yang digunakan pada komputer Windows ini adalah pin DIO 5 dan pin DIO 7.

Hal penting yang tidak boleh dilupakan dalam merangkai peralatan pengukur kecepatan putar berbasis RTLinux ini adalah *ground* dari kedua kartu DAQ NI PCI 6024E, beserta *ground* dari rangkaian *voltage regulator* dan encoder harus dihubungkan satu sama lain. Tujuannya untuk menghindari terjadinya rangkaian *open loop* saat proses pengukuran kecepatan putar berbasis *real time* Linux.

Pada pengukuran kecepatan putar berbasis RTLinux digunakan rangkaian *voltage regulator* dengan tujuan untuk mem-*pull-up* tegangan keluaran encoder sebelum dialirkan ke pin-pin pada counter. Hal ini dilakukan karena keluaran dari encoder merupakan keluaran yang berasal dari *open collector* TTL. Tegangan keluaran encoder tersebut di-*pull-up* sebesar 5V karena tegangan maksimum yang diizinkan sebagai masukan ke pin counter sebesar 5,5V (*sumber: 6023E/6024E/6025E User Manual [1]*). Berikut ini ditampilkan gambar skema rangkaian *voltage regulator* yang digunakan pada pengukuran kecepatan putar berbasis RTLinux:



Gambar 3.4 Skema Rangkaian Voltage Regulator

3.3.2 Perancangan Perangkat Lunak Pengukuran Kecepatan Putar Berbasis Real Time Linux

Perangkat lunak yang digunakan dalam pengukuran kecepatan putar berbasis RTLinux ini terdiri atas tiga buah program yaitu SET (Set Counter), RTPR (Real Time Pulse Reading) dan VSGP (Very Simple Graph Plotter). Ketiga program tersebut dijalankan dalam lingkungan RTLinux. Pembahasan mengenai fungsionalitas dari program-program tersebut sebagai berikut:

a. Program SET (Set Counter).

   Program SET (Set Counter) bertujuan mengaktifkan counter NI PCI 6024E untuk melakukan proses pencacahan pulsa-pulsa informasi yang diberikan kepadanya. Program ini juga menentukan metode perhitungan yang dilakukan oleh counter. Selain berfungsi mengaktifkan counter, program SET juga berfungsi dalam menon-aktifkan counter tersebut. Perlu diketahui juga bahwa syarat agar counter dapat diaktifkan adalah pin DIO 5 pada counter harus diberikan sinyal *down*, dan bila untuk menon-aktifkan counter tersebut maka pin DIO 5 harus diberikan sinyal *up*. Program SET ini bukan merupakan jenis program *real time* Linux.

b. Program RTPR (Real Time Pulse Reading).

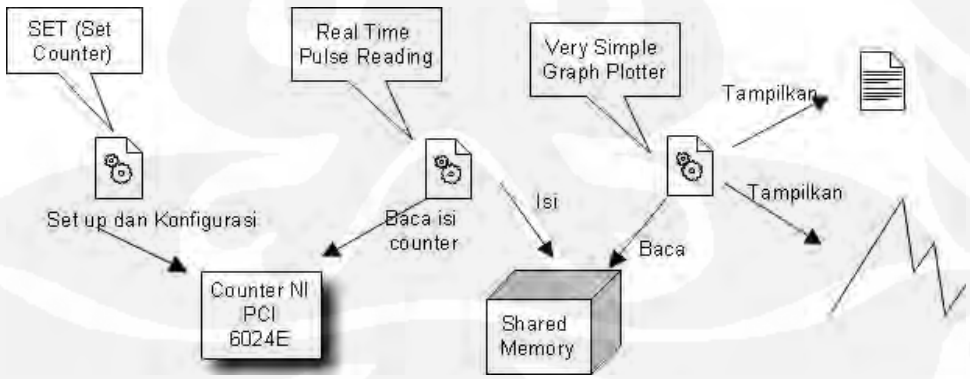
   Program RTPR ini bertujuan untuk membaca isi counter secara periodik dan melakukan perhitungan perubahan posisi sudut selama waktu deteksi. Hasil perhitungan perubahan posisi sudut tersebut diisi ke dalam suatu *shared memory* untuk kelak dibaca oleh program VSGP. Nilai posisi sudut

yang dibaca dari isi counter juga diisi ke dalam *shared memory*. Selain menghitung perubahan posisi sudut dan membaca isi counter, program RTPR juga mencatat waktu pembacaan isi counter ke dalam *shared memory*. Program RTPR ini merupakan jenis program *real time* Linux.

c. Program VSGP (Very Simple Graph Plotter).

Program VSGP berfungsi melakukan perhitungan kecepatan putar dan posisi sudut serta menampilkan hasil perhitungan kecepatan putar dan posisi sudut dalam bentuk grafik. Program ini membaca nilai perubahan pulsa selama waktu deteksi dan posisi sudut dari *shared memory* yang diisi oleh program RTPR. Nilai kecepatan yang ditampilkan oleh program VSGP dalam satuan rpm (*revolutions per minute*) dan nilai posisi sudut yang ditampilkan oleh program VSGP dalam satuan derajat. Bila menerima instruksi untuk melakukan penulisan data perhitungan ke suatu *file*, maka program VSGP akan menyimpan hasil perhitungan kecepatan putar dan posisi sudut dalam fungsi waktu ke dalam suatu *file*. Program VSGP ini bukan merupakan jenis program *real time* Linux.

Untuk memahami interaksi program-program pengukur kecepatan putar berbasis *real time* Linux ini, maka ditampilkan skema interaksi program-program tersebut pada gambar 3.5:



Gambar 3.5 Skema Interaksi Program-Program pada Pengukuran Kecepatan Putar Berbasis Real Time Linux

3.3.2.1 Algoritma Program SET (Set Counter)

Program SET (Set Counter) secara garis besar memiliki fungsi mengaktifkan dan menon-aktifkan counter. Gambar diagram alir algoritma program SET dapat dilihat pada gambar 3.6 berikut ini:

Gambar 3.6 Diagram Alir Algoritma Program SET

Penjabaran secara singkat untuk diagram alir algoritma program SET sebagai berikut:

1. **Inisialisasi.**

   Variabel-variabel yang akan digunakan pada program SET harus diinisialisasi sebelumnya. Kemudian program SET akan mencoba mendapatkan akses penggunaan kartu NI PCI 6024E dengan menggunakan driver COMEDI. Bila akses tersebut berhasil, pengaktifan dilanjutkan pada pin-pin pada counter yang akan digunakan pada pengukuran kecepatan putar berbasis RTLinux. Counter dinon-aktifkan terlebih dahulu sebelum counter tersebut di-*reset*. Tujuan *reset* adalah menghapus konfigurasi mekanisme kerja pada counter yang ditentukan sebelumnya. Setelah counter di-*reset*, maka ditentukan sumber masukan bagi gerbang *source* dan gerbang *gate* berasal dari sumber eksternal atau sumber yang bukan berasal dari kartu NI PCI 6024E itu sendiri. Lalu, isi nilai yang ada di *save register* counter dengan nilai nol. Setelah itu, mekanisme perhitungan pada counter yang dikonfigurasi adalah *buffered non-cumulative event counting*. Dengan dipilihnya mekanisme *buffered non-cumulative event counting* pada counter, maka pin DIO7 pada kartu NI PCI 6024E akan digunakan sebagai *controller* yang mengontrol metode perhitungan counter secara menambah atau secara mengurang.

2. **Pengaktifan Counter.**

   Setelah program SET berhasil melakukan proses inisialisasi, counter siap diaktifkan.

3. **Pengendali Aktivasi Counter.**

   Counter akan dinon-aktifkan bila pin DIO 5 pada kartu NI PCI 6024E diberikan sinyal *up*. Ketika sinyal *up* diberikan, proses selanjutnya adalah program SET akan menon-aktifkan counter. Setelah counter dinon-aktifkan, counter akan direset dan selanjutnya nilai *save register* counter akan diisi dengan nilai nol. Setelah itu, program SET akan dihentikan.

3.3.2.2 Konfigurasi Register-Register pada Driver NI PCI 6024E

Mekanisme *buffered non-cumulative event counting* belum tersedia pada *driver* COMEDI untuk NI PCI 6024E. Oleh karena itu, mekanisme tersebut harus dibuat sendiri dan ditambahkan ke dalam *library driver* COMEDI. Mekanisme *buffered non-cumulative event counting* yang hendak dibuat tersebut harus memiliki karakteristik sebagai berikut:

- Memberikan kemampuan pada counter untuk me-*reset* sendiri dirinya ke nilai inisialnya. Kemampuan ini diperlukan untuk menandakan telah terjadinya satu kali revolusi dalam pengukuran kecepatan putar. Gerbang pada counter yang berfungsi memberitahukan telah terjadi satu kali revolusi dalam suatu putaran adalah gerbang *gate*.
- Perhitungan jumlah pulsa dapat dilakukan secara menambah (*count up*) dan secara mengurang (*count down*). *Count up* berarti isi nilai counter ditambah seiring dengan adanya sinyal masukan di gerbang *source*, sedangkan *count down* berarti isi nilai counter dikurang seiring dengan adanya sinyal masukan di gerbang *source*.
- Isi counter hasil perhitungan jumlah pulsa dapat dibaca setiap saat tanpa perlu menunggu terlebih dahulu *interrupt* yang diberikan oleh counter.

Berikut ini adalah konfigurasi register-register pada *driver* NI PCI 6024E yang harus dilakukan dalam pembuatan mekanisme *buffered non-cumulative event counting* pada *driver* COMEDI tersebut:

- Untuk register G_Input_Select_Register, bit-bit register yang akan dikonfigurasi pada register ini, ditampilkan dalam tabel berikut ini:

Tabel 3.2 Tabel Bit-Bit Register G_Input_Select_Register

No.	Bit Register	Deskripsi	Nilai Bit
1.	G_Source_Polarity	Bit ini menentukan jenis *active edge* apakah yang akan men-*trigger* counter untuk melakukan perhitungan. Ada dua macam *active edge* yang tersedia yaitu *rising edge* dan *falling edge*.	0 (*rising edge*)
2.	G_Gate_Select_ Load_Source	Bit ini bertujuan untuk membolehkan pemilihan *load register* oleh gerbang *gate* dari counter. Ketika bit ini ditetapkan bernilai 1, sebuah *active gate level* akan memilih *load register*	0 (*disable*)

Tabel 3.2 Tabel Bit-Bit Register G_Input_Select_Register (lanjutan)

		A, dan sebuah *inactive gate level* akan memilih *load register* B. Fitur ini dapat digunakan hanya bersamaan dengan *level gating*.	
3.	G_OR_Gate	Bit ini menentukan apakah sinyal *gate* yang dipilih akan di-OR-kan dengan output dari counter yang lain. Penetapan nilai nol untuk tidak mengaktifkan fungsi ini, sedangkan penetapan nilai satu untuk mengaktifkan fungsi ini.	0

- Untuk register G_Mode, bit-bit register yang akan dikonfigurasi pada register ini ditampilkan dalam tabel berikut ini:

Tabel 3.3 Tabel Bit-Bit Register G_Mode

No.	**Bit Register**	**Deskripsi**	**Nilai Bit**
1.	G_Reload_Source_ Switching	Bila Gi_Gate_Select_Load_Source ditetapkan bernilai nol, maka bit ini membolehkan pemilihan load register dalam sifat sebagai berikut ini: - 0: Selalu menggunakan *load register* yang sama. - 1: Berselang-seling di antara dua *load register*.	1
2.	G_Loading_On_TC	Bit ini menentukan sifat counter pada TC (Terminal Count). Sifat-sifat yang dapat diterapkan pada counter sebagai berikut: - 0: *roll over* on TC. - 1: *reload* on TC.	0
3.	G_Gate_Polarity	Bit ini memilih polaritas input dari masukan sinyal pada gerbang gate. Setting nilai nol untuk memilih *active high* dan setting nilai satu untuk memilih *active low*.	0 (*active high*)
4.	G_Loading_On_ Gate	Bit ini menentukan apakah sinyal gate menyebabkan terjadinya *reload* pada counter atau tidak. Bila bit ini ini diatur bernilai 1, maka isi counter akan diinisialisasi ke nilai awal saat counter menerima *trigger* dari *gate*.	1

Tabel 3.3 Tabel Bit-Bit Register G_Mode (lanjutan)

5.	G_Gate_On_Both_ Edges	Bit ini membolehkan penggunaan kedua gate edges (*rising edge* dan *falling edge*) untuk menghasilkan *interrupt* gate dan/atau untuk mengontrol operasi counter. *Setting* nilai nol untuk tidak mengaktifkan fungsi ini dan *setting* nilai satu untuk mengaktifkan fungsi ini.	0
6.	G_Counting_Once	Bit ini menentukan apakah *hardware* yang menon-aktifkan perhitungan pada counter atau tidak. Bila bit ini diatur bernilai 0, maka counter tidak pernah dinon-aktifkan oleh *interrupt* dari *gate*.	0
7.	G_Stop_Mode	Bit ini menentukan kondisi apakah counter akan berhenti berhitung. Bila bit ini diatur bernilai 0, maka counter akan berhenti berhitung bila adanya perubahan kondisi sinyal *gate*.	0
8.	G_Gating_Mode	Bit ini menentukan pemilihan metode *gating* pada counter. Bila bit ini diatur bernilai 2, maka metode *gating* pada counter adalah *edge gating*.	2
9.	G_Trigger_Mode_ For_Edge_Gate	Bit ini menentukan metode pemicu pada counter, bila *gating* diaktifkan. Bila bit ini diatur bernilai 3, maka *gate* digunakan untuk memicu terjadinya aktivitas *reload*, *load* dan *save* pada counter, bukan aktivitas pemberhentian kerja counter.	3

Pembuatan mekanisme *buffered non-cumulative event counting* pada *driver* NI PCI 6024E meliputi modifikasi pada *file-file* berikut ini:

- comedi.h

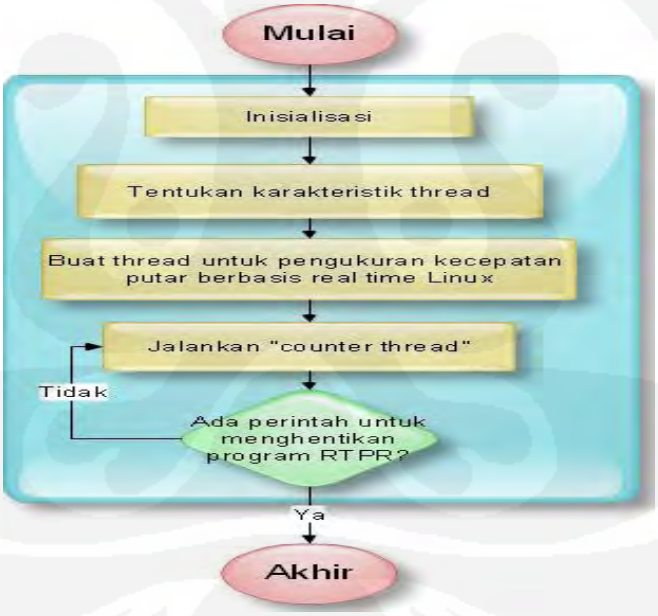
  Tujuan dari *file* ini adalah sebagai *header file* yang menyimpan deklarasi variabel-variabel yang digunakan oleh COMEDI. Agar mekanisme *buffered non-cumulative event counting* dikenali oleh COMEDI, perlu dideklarasi mekanisme *buffered non-cumulative event counting* pada *file* ini. Modifikasi pada *file* comedi.h dilakukan pada comedi 0.7.73 dan comedilib 0.7.22.

- ni_mio_common.c

  Tujuan dari *file* ini adalah sebagai *file* yang menyimpan kode-kode eksekusi mekanisme-mekanisme yang dapat dijalankan oleh kartu NI PCI 6024E. Pada *file* ini, perlu ditambahkan mekanisme *buffered non-cumulative event counting* agar counter dapat menjalankan fungsi *buffered non-cumulative event counting*. Modifikasi pada file ni_mio_common.c hanya dilakukan pada comedi 0.7.73.

3.3.2.3 Algoritma Program RTPR (Real Time Pulse Reading)

Program RTPR merupakan jenis program *real time* yang berarti program ini dijalankan dalam level *kernel space*. Pengaktifkan program yang dijalankan di level *kernel space* berbeda dengan program yang dijalankan di level *user space*. Untuk mengaktifkan program RTPR, digunakan perintah "insmod" pada *terminal* Linux. Gambar diagram alir algoritma program RTPR dapat dilihat pada gambar 3.7.



Gambar 3.7 Diagram Alir Algoritma Program RTPR

Penjabaran secara singkat untuk diagram alir algoritma program RTPR sebagai berikut:

1. **Inisialisasi.**

   Program RTPR merupakan juga *module* karena program ini dijalankan di dalam *kernel space*. Saat program RTPR dijalankan, variabel-variabel

yang dibutuhkan oleh program ini didefinisikan terlebih dahulu. Kemudian, program RTPR mencoba mendapatkan akses penggunaan kartu NI PCI 6024E melalui driver COMEDI. Bila berhasil mendapatkan akses penggunaan kartu tersebut, program RTPR membuat *shared memory* agar dapat digunakan secara bersama-sama dengan program VSGP.

2. **Penentuan Karakteristik Tread.**

   Thread adalah suatu proses yang dijalankan dalam *module kernel* Linux yang memiliki kemampuan untuk berkomunikasi, ber-*interrupt*, dan berkirim pesan antar *thread* yang lain serta berbagi memori antar *thread* yang lain. *Thread* yang akan dibuat harus memiliki prioritas tertinggi dibandingkan program lainnya yaitu program SET dan program VSGP. Selain itu, *thread* yang dibuat ini akan dijalankan berulang-ulang secara periodik.

3. **Pembuatan Thread untuk Pengukuran Kecepatan Putar berbasis Real Time Linux**

   Setelah ditentukan karakteristik *thread* yang akan dibuat, maka tahap selanjutnya adalah membuat sebuah *thread* dengan nama “counter thread”.

4. **Pengoperasian Counter Thread**

   Setelah “counter thread” dibuat, maka thread tersebut sudah siap dioperasikan. Thread tersebut terus dijalankan selama tidak ada perintah dari pengguna untuk menon-aktifkan program RTPR. Cara menon-aktifkan program RTPR berbeda dengan cara menon-aktifkan program umumnya yang dijalankan dalam *user space*. Untuk menon-aktifkan program RTPR, digunakan perintah “rmmod” pada *terminal* Linux. Saat program RTPR dinon-aktifkan, *thread* dan *shared memory* akan dihapus.

Berdasarkan *datasheet* DC Gearmotor GM8712-21, kecepatan maksimum motor DC yang digunakan tanpa beban sebesar 396rpm. Oleh karena itu, pengukuran kecepatan putar berbasis RTLinux ini harus mendukung pengukuran kecepatan putar hingga 400 rpm. “Counter thread” memiliki fungsi membaca nilai posisi sudut secara periodik dari isi counter. Waktu pembacaan yang dirancang untuk “counter thread” sebesar 10*ms*. Berdasarkan metode M, nilai kecepatan

putar diperoleh dari selisih dari perubahan posisi sudut dibagi dengan selisih waktu. Untuk menentukan besarnya selisih waktu/waktu deteksi (T_c) dalam pengukuran kecepatan putar berbasis RTLinux ini, digunakan persamaan (3.4) sebagai berikut:

$$T_c = \frac{60m}{P\omega}$$

Pada pengukuran kecepatan putar berbasis RTLinux ini ditentukan bahwa untuk mengukur kecepatan putar sebesar 400 rpm maka nilai untuk banyaknya pulsa yang dihasilkan selama waktu deteksi sebesar 60 buah. Oleh karena itu, nilai waktu deteksi yang diperoleh melalui perhitungan dengan persamaan (3.4) sebesar 0,090s (*seconds*) atau 90*ms*. Hal ini berlaku untuk pemakaian instrumen *incremental rotary* encoder E50S8-100-3-N-24 yang beresolusi 100.

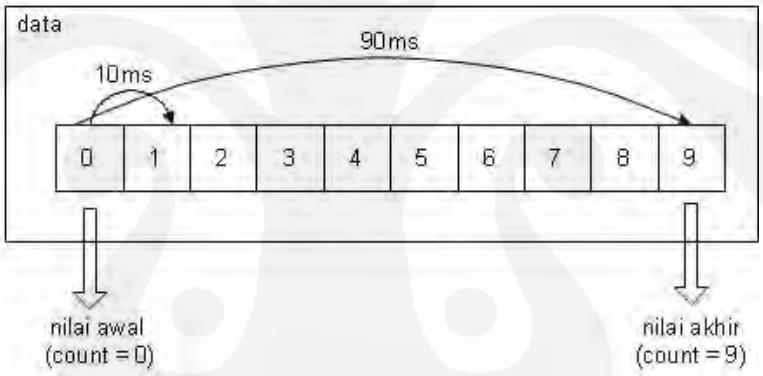
Sebelum memasuki diagram alir algoritma "counter thread" pada program RTPR, berikut ini adalah variabel-variabel yang digunakan pada "counter thread" tersebut:

- **count**, variabel yang berfungsi sebagai penunjuk nomor urut variabel yang diakses dalam array "data".
- **data**, merupakan variabel jenis *array*. *Array* adalah kumpulan dari beberapa variabel. *Array* "data" memiliki fungsi menyimpan hasil pembacaan nilai posisi sudut.
- **dummy**, variabel yang berfungsi memberitahukan adanya perubahan nilai delta.
- **delta**, variabel yang berisi nilai perubahan posisi sudut selama waktu deteksi.
- **tanda**, variabel yang berfungsi menandakan kondisi perubahan metode perhitungan pada counter. Bila counter yang semula melakukan perhitungan *count up* dan kemudian berhenti selama selang waktu tertentu, dan selanjutnya melakukan perhitungan secara *count down*, maka variabel "tanda" menandakan nilai 1. Bila counter yang semula melakukan perhitungan *count down* dan kemudian berhenti selama selang waktu tertentu, dan selanjutnya melakukan perhitungan secara *count up*, maka variabel "tanda" menandakan nilai 2. Bila counter melakukan metode

perhitungan yang sama untuk waktu sebelumnya dan saat ini maka variabel “tanda” menandakan nilai 0.

- **MAX** adalah nilai hasil pembagian nilai waktu deteksi dengan nilai *sampling*. Max bernilai 9 diperoleh dari pembagian nilai 90*ms* dengan nilai 10*ms*.
- **nilai awal**, variabel yang akan diisi dengan nilai *array* “data” urutan yang pertama.
- **nilai akhir**, variabel yang akan diisi dengan nilai *array* “data” urutan yang terakhir.
- **REVOLUTION,** variabel yang menandakan banyaknya pulsa yang dihasilkan oleh encoder dalam satu revolusi.

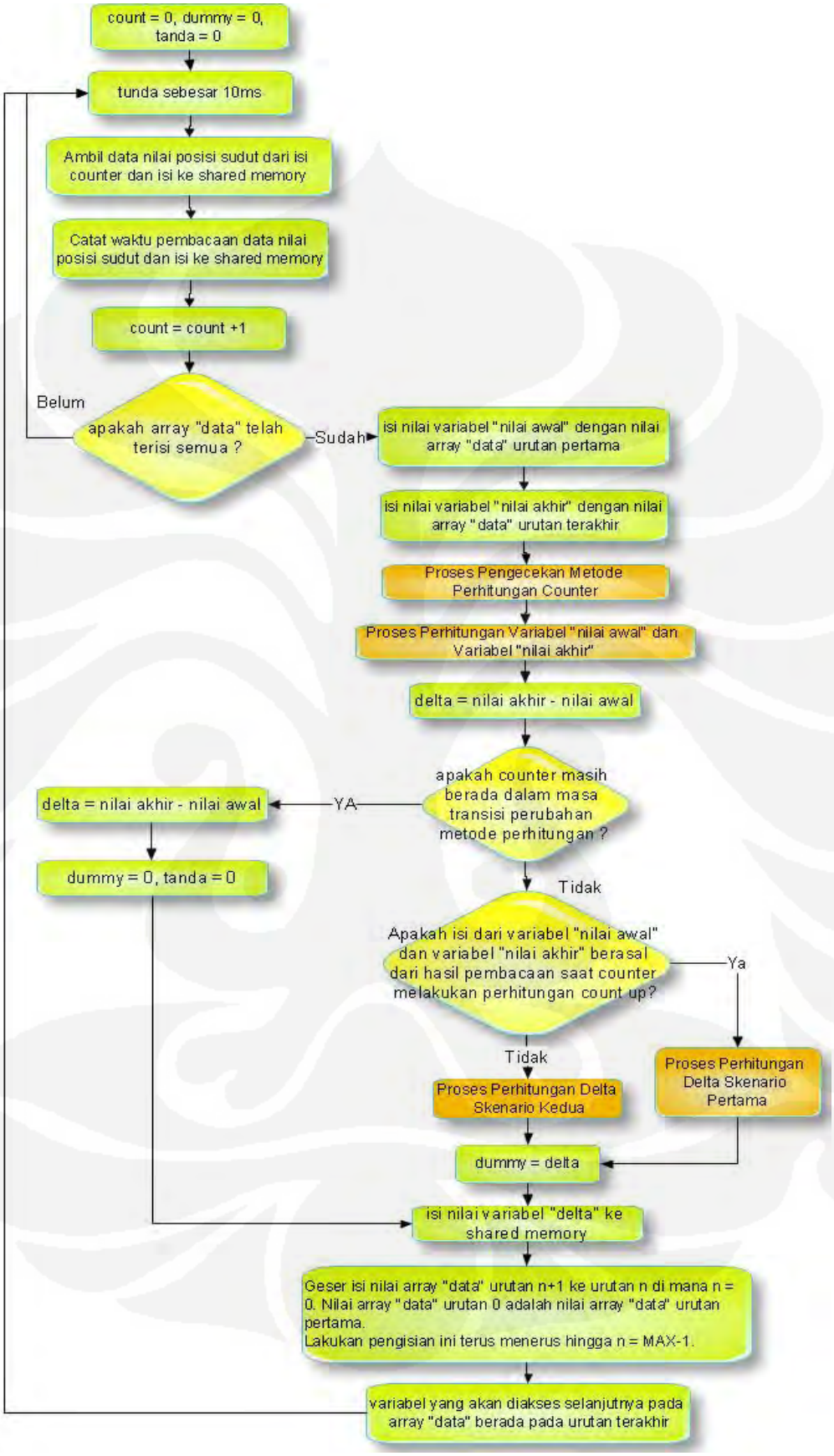
Untuk memperjelas pemahaman mengenai variabel “data” yang digunakan pada program RTPR, berikut ini ditampilkan gambar isi variabel “data”:



Gambar 3.8 Isi Variabel “data”

Pada gambar 3.8 tersebut, terlihat bahwa variabel “data” memiliki sepuluh buah variabel. Isi pada variabel “data” tersebut diisi untuk setiap 10*ms*. Nilai variabel “nilai awal” merupakan isi dari nilai variabel “data” yang pertama, di mana nilai variabel “count” bernilai 0. Nilai variabel “nilai akhir” merupakan isi dari nilai variabel “data” yang terakhir, di mana nilai variabel “count” bernilai 9. Jadi, perbedaan waktu antara hasil pembacaan nilai posisi sudut untuk variabel “nilai awal” dengan hasil pembacaan nilai posisi sudut untuk variabel “nilai akhir” sebesar 90*ms*.

Diagram alir algoritma “counter thread” pada program RTPR dapat dilihat pada gambar 3.9.



Gambar 3.9 Diagram Alir Algoritma Counter Thread pada Program RTPR

Diagram alir pada gambar 3.9 menunjukkan proses pencarian nilai perubahan posisi sudut selama waktu deteksi dan proses pembacaan nilai posisi sudut dari isi counter untuk setiap 10*ms*. Penjelasan mengenai diagram alir algoritma "counter thread" pada program RTPR sebagai berikut:

1. **Inisialisasi.**

   Pertama-tama, variabel "count", "tanda" dan "dummy diisi dengan nilai nol. Variabel-variabel yang digunakan dalam *array* "data" berjumlah 10 buah. Jumlah ini diperoleh dari pembagian antara waktu deteksi yaitu sebesar 90*ms* dengan waktu pembacaan isi counter yaitu sebesar 10*ms* dan kemudian hasil pembagian tersebut ditambah dengan satu.

2. **Proses Pembacaan Nilai Posisi Sudut.**

   Setelah proses inisialisasi, "counter thread" akan terlebih dahulu melakukan penundaan kerja *thread* tersebut untuk selama 10*ms*. Setelah itu, "counter thread" membaca nilai posisi sudut dari isi counter dan diisikan ke *array* "data". Sambil mengisi nilai variabel dalam *array* "data", "counter thread" juga mengisi hasil pembacaan nilai posisi sudut tersebut ke dalam *shared memory*. Selain itu, "counter thread" juga mencatat waktu pembacaan data nilai posisi sudut dan mengisi nilai waktu tersebut ke *shared memory*.

3. **Proses Penghitungan Nilai Variabel "nilai awal" dan Variabel "nilai akhir".**

   Bila variabel dalam *array* "data" belum telah terisi penuh, selanjutnya "counter thread" akan melakukan pembacaan isi counter kembali. Dalam kasus ini, bila kesepuluh variabel yang digunakan tersebut semuanya telah terisi oleh nilai posisi sudut, maka selanjutnya dilakukan proses pencarian perubahan posisi sudut. Proses pencarian perubahan posisi sudut akan dimulai dari pengisian nilai variabel "nilai awal" dengan nilai *array* "data" urutan pertama dan nilai variabel "nilai akhir" dengan nilai *array* "data" urutan terakhir. Setelah itu, proses selanjutnya adalah proses pengecekan metode perhitungan counter yang bertujuan untuk memastikan bahwa perhitungan yang dilakukan counter saat ini bukan berada dalam masa transisi. Kemudian "counter thread" akan mengakses ke bagian proses

perhitungan variabel “nilai awal” dan variabel “nilai akhir” bila counter berada dalam kondisi transisi.

4. **Proses Penghitungan Nilai Perubahan Posisi Sudut.**

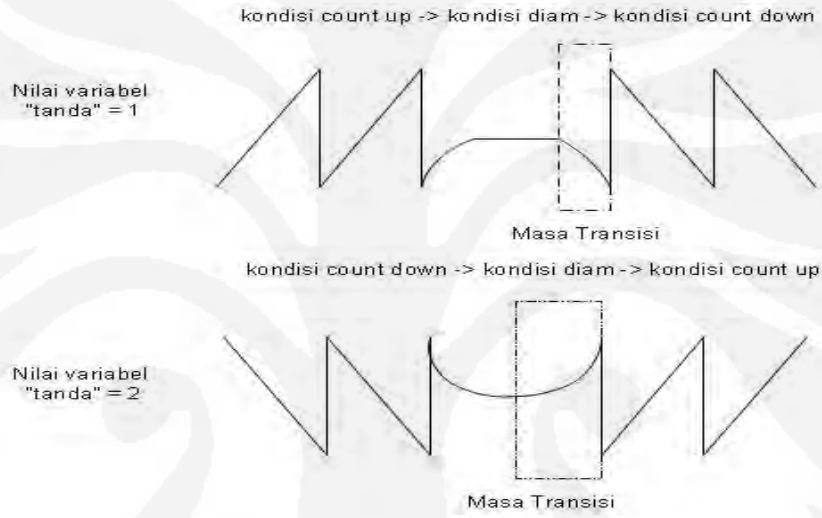
   Hasil selisih dari nilai posisi sudut saat ini dengan nilai posisi sudut untuk 90*ms* yang lalu diisikan ke variabel “delta”. Selanjutnya, bila counter masih berada dalam masa transisi, maka nilai delta diperoleh dari pengurangan “nilai akhir” dengan “nilai awal”. Bila counter tidak berada dalam masa transisi, maka dilanjutkan ke algoritma pemeriksaan nilai posisi sudut saat ini dan nilai posisi sudut untuk 90*ms* yang lalu dibaca saat counter sedang *count up*. Penentuan counter sedang *count up* dapat diketahui dari bila nilai kedua posisi sudut baik yang dibaca saat ini maupun yang dibaca 90*ms* yang lalu lebih kecil dari 8388608. Karena counter NI PCI 6024E memiliki resolusi 24-bit maka nilai maksimum counter tersebut adalah 16777216. Jadi, nilai 8388608 diperoleh dari pembagian antara nilai 16777216 dengan nilai 2. Alasan dipilihnya nilai 8388608 adalah bahwa saat counter dalam kondisi *count up*, perhitungan yang dilakukannya tidak dapat melewati nilai batasan tersebut karena encoder yang digunakan beresolusi 100. Bila kondisi tersebut terpenuhi maka “counter thread” akan melanjutkan ke algoritma yang bernama “Proses Perhitungan Delta Skenario Pertama”, sedangkan bila kondisi tersebut tidak terpenuhi maka “counter thread” akan melanjutkan ke algoritma yang bernama “Proses Perhitungan Delta Skenario Kedua”. Hasil perhitungan nilai variabel “delta” yang baru yang diperoleh baik dari proses perhitungan dengan menggunakan algoritma “Proses Pertama” maupun dengan algoritma “Proses Kedua” diisi ke variabel “dummy” dan ke *shared memory*.

5. **Menggulang Kembali Proses Perhitungan Counter Thread.**

   Proses selanjutnya adalah melakukan pergeseran isi pada variabel dalam *array* “data”. Proses pergeseran isi ini dimulai dari mengganti nilai variabel urutan yang pertama dalam *array* “data” dengan nilai variabel urutan yang kedua dalam *array* “data”, kemudian nilai variabel urutan yang kedua dengan nilai variabel urutan yang ketiga dan demikian

seterusnya hingga nilai variabel urutan kedua dari terakhir diganti dengan nilai variabel urutan yang terakhir. Setelah selesai melakukan proses pergeseran isi variabel pada array “data”, counter kembali menunggu selama 10*ms* sebelum melakukan proses pembacaan nilai posisi sudut dari isi counter. Nilai posisi sudut untuk pembacaan selanjutnya akan diisi ke variabel urutan terakhir dalam *array* “data”.

Berikut ini adalah gambar masa transisi perubahan metode perhitungan counter:



Gambar 3.10 Masa Transisi Perubahan Metode Perhitungan Counter

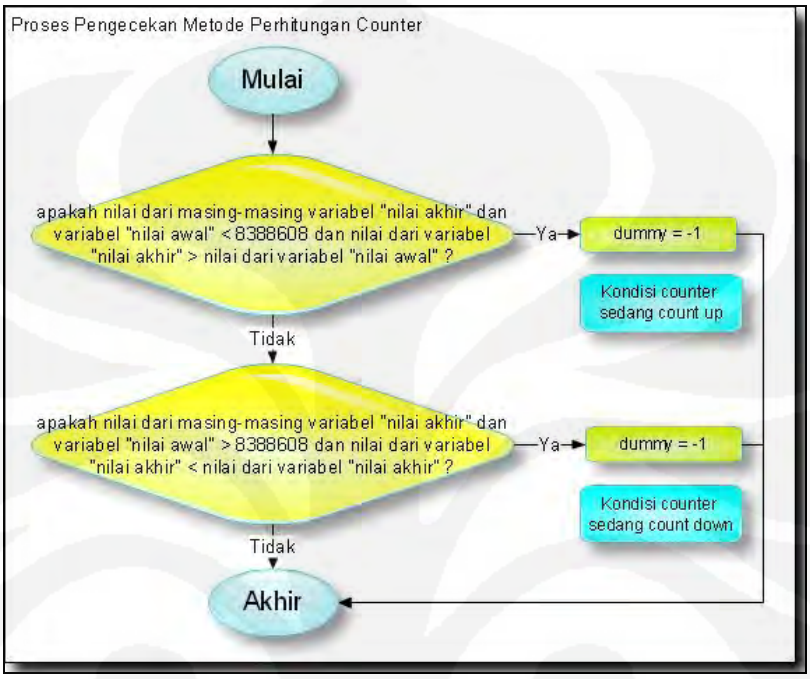
Pada masa transisi tersebut, counter yang semula tidak melakukan perhitungan atau dalam kondisi diam, saat ini mulai melakukan perhitungan baik secara *count up* atau secara *count down*. Persamaan yang digunakan dalam perhitungan nilai variabel “delta” untuk masa transisi adalah

$$\text{delta} = \text{nilai akhir} - \text{nilai awal} \qquad (3.5)$$

Pada masa transisi variabel “dummy” akan terus bernilai nol. Masa transisi tersebut berakhir hingga pembacaan terakhir nilai posisi sudut sebesar 0 derajat.

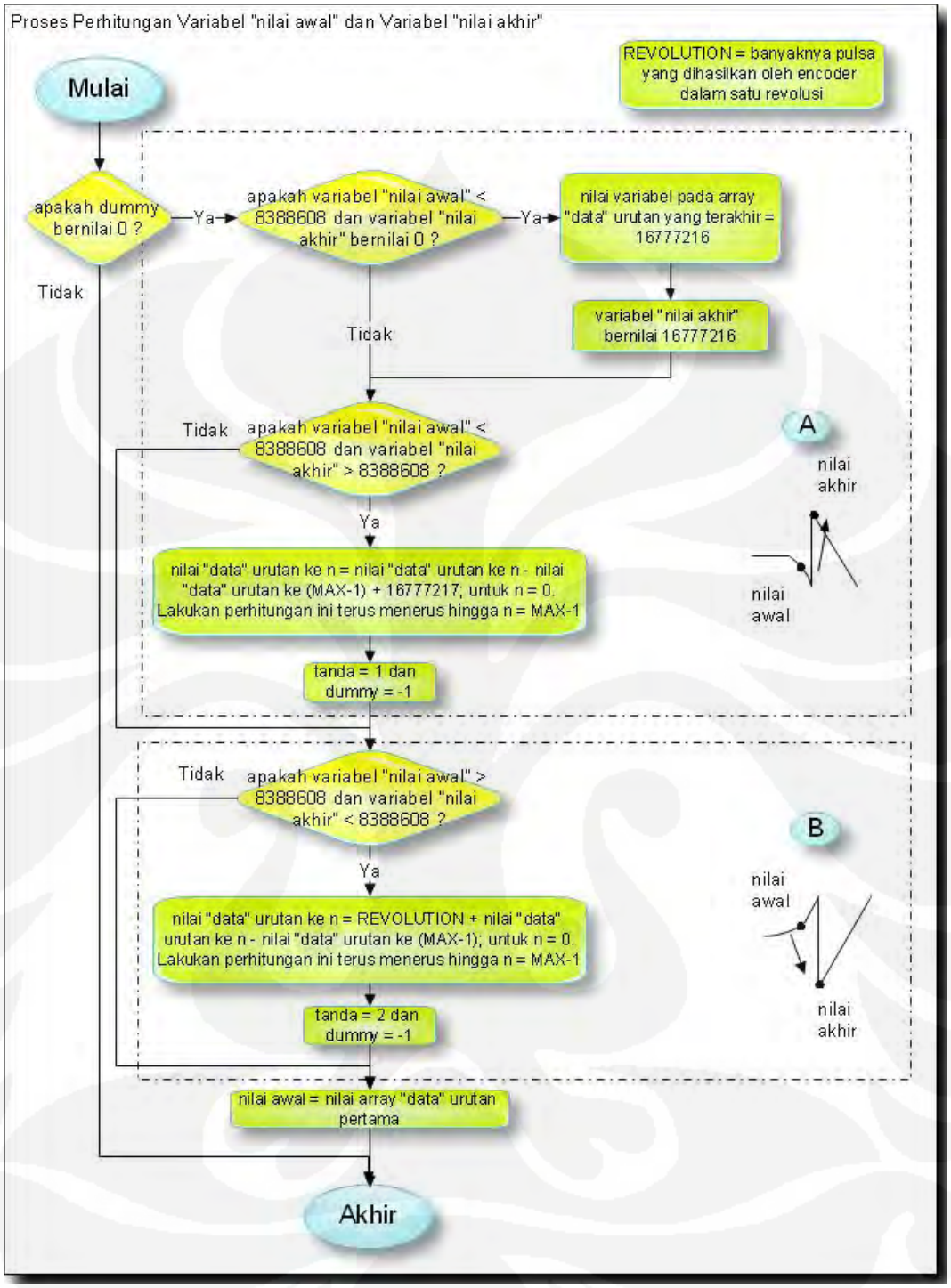
Algoritma “Proses Pengecekan Metode Perhitungan Counter” bertujuan untuk memastikan bahwa perhitungan yang dilakukan counter saat ini bukan berada dalam masa transisi. Bila perhitungan tersebut bukan berada dalam masa transisi maka variabel “dummy” bernilai -1. Kondisi ideal counter saat *count up* adalah nilai pembacaan saat ini lebih besar dari nilai pembacaan sebelumnya,

sedangkan kondisi ideal counter saat *count down* adalah nilai pembacaan saat ini lebih kecil dari nilai pembacaan sebelumnya. Gambar diagram alir algoritma "Proses Pengecekan Metode Perhitungan Counter" pada "counter thread" diilustrasikan dalam gambar 3.11.



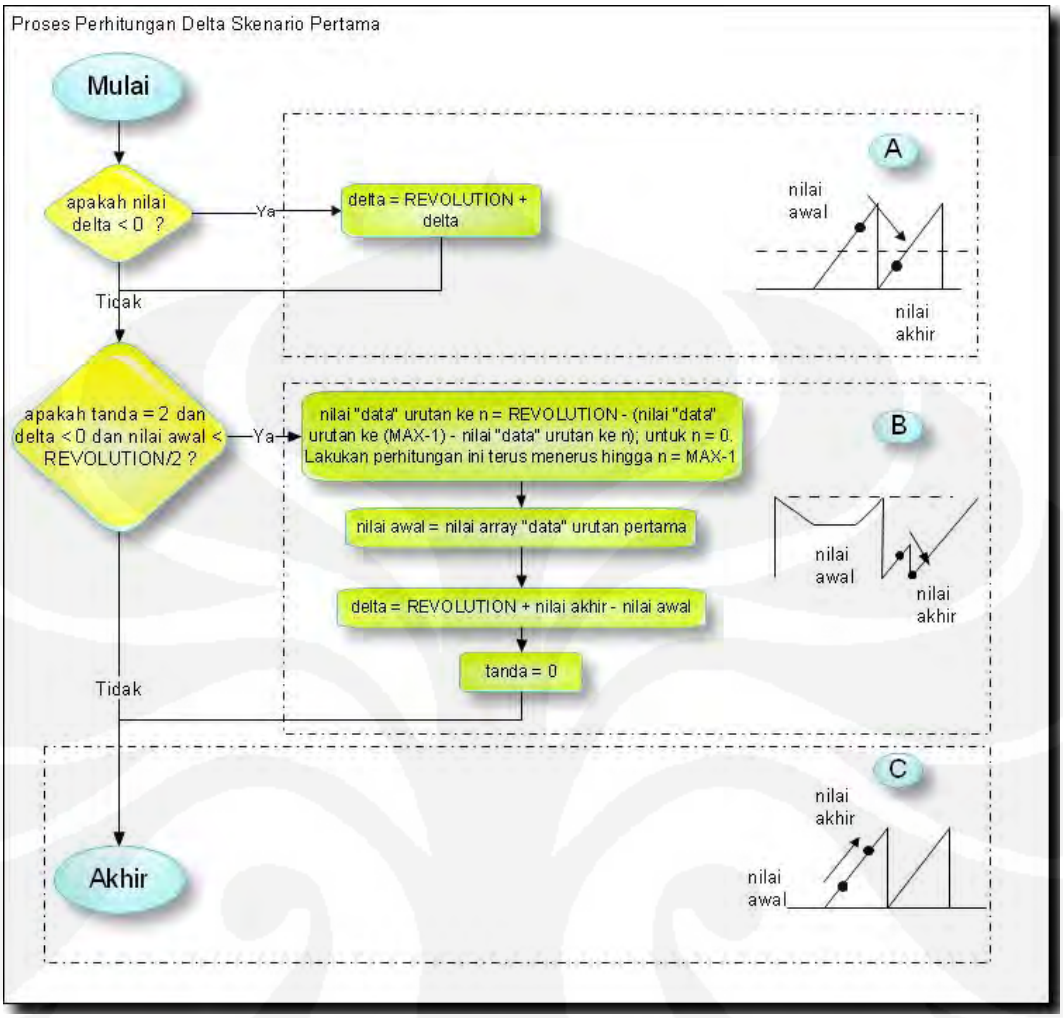
Gambar 3.11 Diagram Alir Algoritma Proses Pengecekan Metode Perhitungan Counter

Algoritma Proses Perhitungan Variabel "nilai awal" dan Variabel "nilai akhir" bertujuan untuk menangani masalah perhitungan variabel "nilai awal" dan "nilai akhir" untuk kondisi berakhirnya masa transisi. Berakhirnya masa transisi ditandai dengan nilai posisi sudut terakhir bernilai nol. Pada gambar 3.12 ditampilkan diagram alir proses perhitungan variabel "nilai awal" dan variabel "nilai akhir". Pada bagian kotak A di gambar 3.12 menandakan algoritma penyelesaian untuk kondisi counter yang bertransisi dari kondisi diam ke kondisi *count down*. Pada bagian kotak B di gambar 3.12 menandakan algoritma penyelesaian untuk kondisi counter yang bertransisi dari kondisi diam ke kondisi *count up*. Bila variabel "dummy" tidak bernilai nol berarti masa transisi telah habis sehingga algoritma proses ini tidak perlu lagi dilakukan.



Gambar 3.12 Diagram Alir Algoritma Proses Perhitungan Variabel “nilai awal” dan Variabel “nilai akhir”

Algoritma “Proses Perhitungan Delta Skenario Pertama” adalah algoritma yang bertujuan menghitung nilai perubahan posisi sudut saat counter sedang *count up*. Gambar diagram alir algoritma “Proses Perhitungan Delta Skenario Pertama” pada “counter thread” diilustrasikan dalam gambar 3.13.



Gambar 3.13 Diagram Alir Algoritma Proses Perhitungan Delta Skenario Pertama

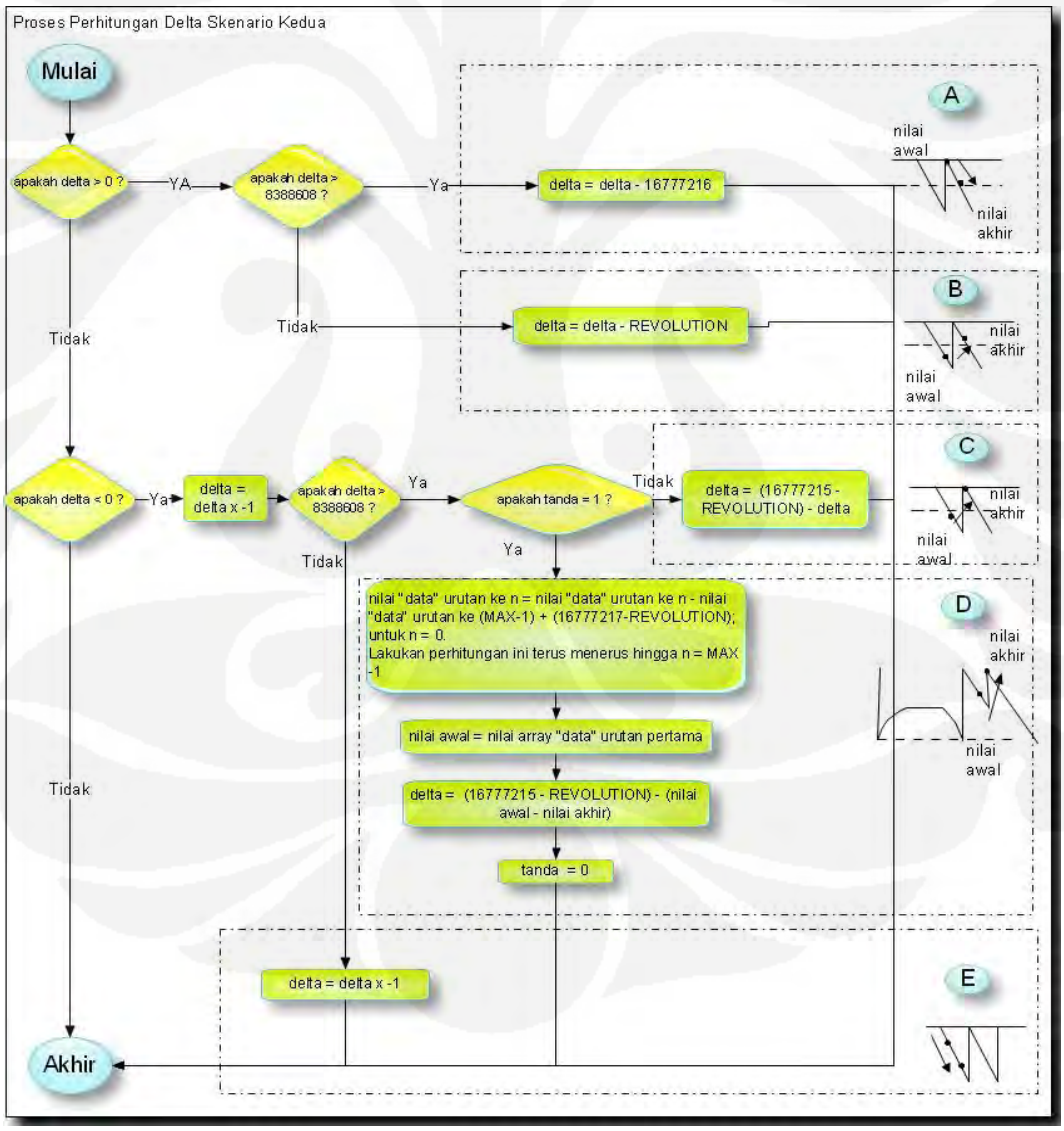
Diagram alir pada gambar 3.13 menunjukkan bahwa bila nilai perubahan posisi sudut yang didefinisikan dengan variabel “delta” lebih kecil dari nol sedangkan counter melakukan metode penambahan, maka terdapat kesalahan dalam perhitungan nilai perubahan posisi sudut yang dilakukan sebelumnya. Oleh karena itu, nilai perubahan posisi sudut yang sebenarnya diperoleh melalui persamaan:

$$\text{delta} = \text{REVOLUTION} + \text{delta} \qquad (3.6)$$

di mana variabel “REVOLUTION” menandakan banyaknya pulsa yang dihasilkan oleh encoder dalam satu revolusi. Hal ini dapat dilihat pada bagian kotak A dalam gambar 3.13. Pada bagian kotak B dalam gambar 3.13 merupakan algoritma yang digunakan untuk menghadapi kondisi masa transisi yang melebihi nilai nol. Kasus ini terjadi karena pergerakan nilai posisi sudut sudah melewati nilai nol tetapi

masih belum melewati masa transisi. Kondisi yang dapat memicu algoritma ini adalah variabel tanda yang masih bernilai 2 dan nilai variabel “nilai awal” lebih kecil dari 50 (diperoleh dari nilai variabel “REVOLUTION” dibagi 2). Alasan dipilihnya nilai 50 karena dianggap nilai batasan tersebut tidak dapat dilewati sebab encoder yang digunakan beresolusi 100. Pada bagian kotak C dalam gambar 3.13 menunjukkan bahwa perhitungan delta sebelumnya telah benar dan menunjukkan kondisi yang sebenarnya.

Algoritma “Proses Perhitungan Delta Skenario Kedua” adalah algoritma yang bertujuan menghitung nilai perubahan posisi sudut saat counter sedang *count down*. Gambar diagram alir algoritma “Proses Perhitungan Delta Skenario Kedua” pada “counter thread” diilustrasikan dalam gambar 3.14.



Gambar 3.14 Diagram Alir Algoritma Proses Perhitungan Delta Skenario Kedua

Penjelasan mengenai diagram alir algoritma “Proses Perhitungan Delta Skenario Kedua” pada “counter thread” sebagai berikut:

1. **Inisialisasi.**

   Pada algoritma ini terdapat dua kondisi yaitu kondisi saat nilai perubahan posisi sudut (didefinisikan dengan variabel “delta”) lebih besar dari nol dan kondisi saat nilai perubahan posisi sudut lebih kecil dari nol. Dalam algoritma ini akan dikenal variabel yang bernama REVOLUTION. Variabel “REVOLUTION” menandakan banyaknya pulsa yang dihasilkan oleh encoder dalam satu revolusi. Untuk kondisi counter *count down*, nilai nol pada isi counter berarti nilai 16777216. Hal ini karena resolusi counter pada NI PCI 6024E sebesar 24 bit.

2. **Untuk Nilai Delta Positif.**

   Bila nilai delta lebih besar dari 8388608 maka perhitungan delta yang sebelumnya perlu diperbaiki. Alasan diambilnya nilai 8388608 adalah diasumsikan bahwa saat counter dalam kondisi *count down*, perhitungan yang dilakukannya tidak dapat melewati nilai batasan tersebut karena encoder yang digunakan beresolusi 100. Kesalahan perhitungan delta ini terjadi karena nilai posisi sudut saat ini bernilai lebih besar dari 8388608 sedangkan nilai posisi sudut untuk 90*ms* yang lalu bernilai nol sebagaimana ditunjukkan dalam bagian kotak A pada gambar 3.14. Untuk memperoleh nilai perubahan posisi sudut yang sebenarnya digunakan persamaan sebagai berikut ini:

   $$delta = delta - 16777216 \qquad (3.7)$$

   Bila nilai delta bukan lebih besar dari 8388608, perhitungan delta yang sebelumnya juga perlu diperbaiki. Kesalahan ini terjadi ketika nilai posisi sudut saat ini bernilai lebih besar dari nilai posisi sudut untuk 90*ms* yang lalu sebagaimana ditunjukkan dalam bagian kotak B pada gambar 3.14. Oleh karena itu, nilai perubahan posisi sudut yang sebenarnya diperoleh melalui persamaan:

   $$delta = delta - REVOLUTION \qquad (3.8)$$

3. Untuk Nilai Delta Negatif.

Bila nilai delta yang dihasilkan bernilai negatif maka hasil perhitungan delta yang bernilai negatif tersebut dikalikan dengan nilai negatif satu (-1) untuk merubahnya ke bentuk positif. Setelah diperoleh nilai delta dalam bentuk positif maka bila delta yang dihasilkan bernilai lebih besar dari 8388608, maka terjadi kesalahan dalam perhitungan delta. Bila variabel "tanda" tidak bernilai satu, maka kondisi yang dialami oleh counter yaitu nilai posisi sudut saat ini bernilai nol dan nilai posisi sudut untuk 90*ms* yang lalu bernilai lebih besar dari 8388608 sebagaimana ditunjukkan dalam bagian kotak C pada gambar 3.14. Oleh karena itu, nilai perubahan posisi sudut yang sebenarnya diperoleh melalui persamaan:

delta = (16777215 - REVOLUTION) - delta (3.9)

Bila variabel "tanda" bernilai satu maka hal ini berarti bahwa masa transisi perhitungan counter belum dilewati. Hal ini dapat dilihat pada bagian kotak D gambar 3.14. Penyebab terjadinya counter melakukan perhitungan melebihi nilai batasan nol karena encoder belum mencapai satu putaran. Oleh karena itu, nilai delta perlu diperbaiki. Bila nilai delta yang dihasilkan lebih kecil dari 8388608, hal ini menandakan bahwa nilai posisi sudut saat ini lebih kecil dari nilai posisi sudut untuk 90*ms* yang lalu sebagaimana ditunjukkan dalam bagian kotak E pada gambar 3.14. Kedua nilai tersebut tidak mengandung nilai nol. Kemudian nilai perubahan posisi sudut dapat diperoleh melalui persamaan (3.10) yaitu:

delta = delta x -1 (3.10)

3.3.2.4 Algoritma Program VSGP (Very Simple Graph Plotter)

Pembahasan algoritma pada program VSGP (*Very Simple Graph Plotter*) hanya meliput algoritma penghitung kecepatan putar. Bagian algoritma "Penghitung Kecepatan Putar" pada program VSGP ini di dalamnya terdapat proses penghitungan kecepatan putar, posisi sudut, penggambaran grafik dan penulisan hasil bacaan ke *file*. Gambar 3.15 menampilkan diagram alir algoritma "Penghitung Kecepatan Putar" pada program VSGP. Program VSGP ini ditulis

dalam bahasa pemrograman C dan menggunakan beberapa fungsi GTK+ di dalam program tersebut.

Gambar 3.15 Diagram Alir Algoritma Penghitung Kecepatan Putar pada Program VSGP

Penjelasan mengenai diagram alir algoritma "Penghitung Kecepatan Putar" pada program VSGP sebagai berikut:

1. **Timer.**

   Pewaktuan VSGP ini diatur oleh sebuah *timer* sehingga proses penghitungan kecepatan putar, posisi sudut, penggambaran grafik dan penulisan hasil bacaan ke *file* dapat dijalankan dalam suatu interval waktu yang telah ditentukan. Dalam pengukuran kecepatan putar berbasis RTLinux ini, interval waktu ditentukan sebesar 10*ms* karena program RTPR melakukan pembacaan isi counter berlangsung untuk setiap 10*ms*. Oleh karena itu, interval waktu pada program VSGP harus lebih besar atau sama dengan 10*ms* supaya tidak ada kemungkinan untuk program VSGP

mengambil nilai posisi sudut yang sama untuk kedua kalinya dari *shared memory*.

2. **Proses Penghitungan Kecepatan Putar.**

   Program VSGP akan membaca isi *shared memory* yang berisi informasi dikirim oleh program RTPR. Informasi tersebut berisi nilai perubahan posisi sudut selama waktu deteksi dan nilai waktu pembacaan posisi sudut. Nilai waktu tersebut akan disimpan ke dalam suatu variabel sementara. Selanjutnya, program VSGP akan menghitung nilai kecepatan putar dalam bentuk rpm (*revolutions per minute*). Persamaan (3.4) digunakan untuk mencari nilai kecepatan putar ditulis sebagai berikut:

$$\omega = \frac{60m}{PT_c} \text{ (rpm)}$$

   dengan ω adalah kecepatan putar (rpm), m merupakan banyaknya pulsa yang dihasilkan selama T_c , P merupakan banyaknya pulsa yang dihasilkan oleh encoder dalam satu revolusi, dan T_c merupakan waktu deteksi (s). Nilai P telah ditentukan sesuai dengan spesifikasi encoder yang digunakan yaitu beresolusi 100. Bila kecepatan yang dihasilkan bernilai positif maka counter *count up*, sedangkan bila kecepatan yang dihasilkan bernilai negatif maka counter *count down*. Setelah nilai kecepatan putar dihitung, maka program VSGP akan menggambarkan grafik kecepatan putar terhadap waktu.

3. **Proses Penghitungan Posisi Sudut.**

   Program VSGP akan membaca isi *shared memory* yang berisi informasi mengenai jumlah pulsa yang dibaca dari isi counter untuk setiap 10*ms* oleh program RTPR. Dari variabel jumlah pulsa tersebut diperoleh posisi sudut dari motor DC yang berputar saat ini. Rumus yang digunakan dalam mencari posisi sudut sebagai berikut:

$$\theta = \frac{\phi * 360^o}{P} \text{ (derajat)} \qquad (3.11)$$

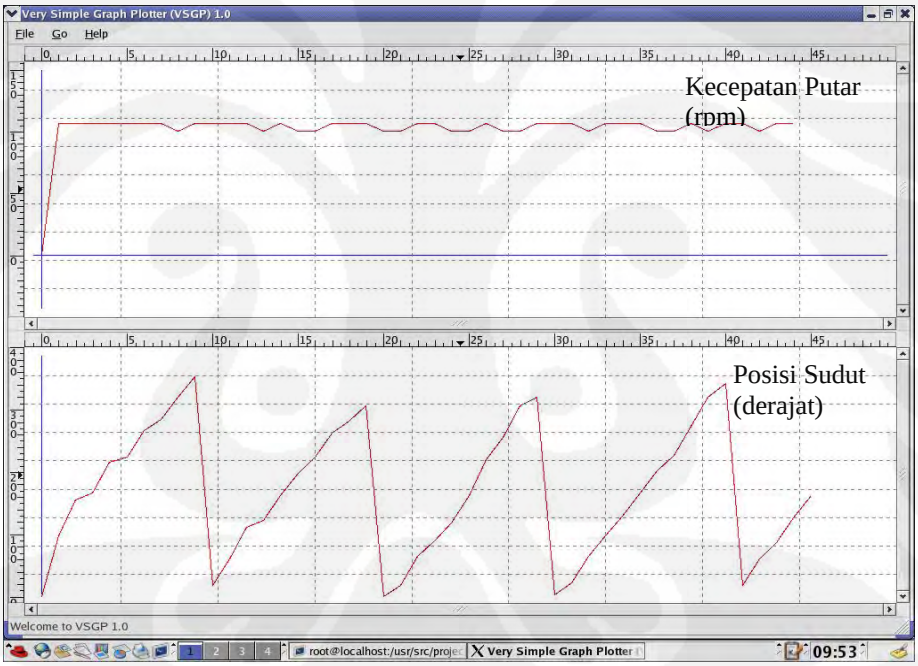
   dengan θ adalah posisi sudut dalam bentuk derajat, ϕ adalah jumlah pulsa yang dibaca oleh program VSGP saat ini, dan P merupakan banyaknya pulsa yang dihasilkan oleh encoder dalam satu revolusi. Nilai P telah

ditentukan sesuai dengan spesifikasi encoder yang digunakan yaitu bernilai 100. Setelah nilai posisi sudut dihitung, maka program VSGP akan menggambarkan grafik posisi sudut terhadap waktu.

4. **Proses Penulisan Data Hasil Perhitungan ke File.**

   Bila program VSGP menerima *trigger* untuk menulis data ke *file* maka program VSGP akan menyimpan nilai hasil perhitungan kecepatan putar dan posisi sudut dalam fungsi waktu ke dalam suatu *file*. Kemudian, program VSGP akan melaksanakan kembali proses penghitungan kecepatan putar setelah menunggu untuk waktu selama 10*ms*.

Berikut ini adalah gambar menu tampilan pada program VSGP yang akan dijalankan dalam proses pengukuran kecepatan putar berbasis *real time* Linux:



Gambar 3.16 Menu Tampilan pada Program VSGP

3.3.3 Cara Pengoperasian Perangkat Lunak Pengukuran Kecepatan Putar Berbasis Real Time Linux

Dalam proses pengoperasian perangkat lunak pengukuran kecepatan putar berbasis RTLinux, program harus dijalankan secara berurutan. Program yang pertama kali dijalankan adalah program SET kemudian disusul oleh program RTPR dan VSGP. Program SET harus dijalankan pertama kali karena program

SET berfungsi mengaktifkan counter dan mengkonfigurasi counter dalam hal metode perhitungannya. Setelah counter berhasil dikonfigurasi dan diaktifkan maka program RTPR baru dapat dijalankan. Program RTPR hanya melakukan proses membaca isi counter dan melakukan perhitungan nilai perubahan posisi sudut selama waktu deteksi. Oleh karena itu, program RTPR tidak dapat berjalan dengan baik sebelum counter diaktifkan dan dikonfigurasi. Setelah program RTPR dijalankan, baru dilanjut dengan pengaktifan program VSGP. Program VSGP berfungsi melakukan perhitungan kecepatan putar, dan menampilkan hasil perhitungan kecepatan putar dan posisi sudut dalam bentuk grafik. Program VSGP tidak dapat diaktifkan sebelum program RTPR. Hal ini karena program VSGP membutuhkan *shared memory* yang hanya dapat dibuat oleh program RTPR. Di dalam shared memory tersebut, program VSGP dapat mengakses nilai posisi sudut, waktu pembacaan nilai posisi sudut dan nilai perubahan posisi sudut selama waktu deteksi.

Dalam proses pengnon-aktifkan pengukuran kecepatan putar berbasis RTLinux, program juga harus dinon-aktifkan secara berurutan supaya tidak terjadi konflik. Program yang pertama kali dinon-aktifkan adalah program VSGP, kemudian disusul oleh program RTPR dan setelah itu program SET.

BAB IV

PENGUJIAN DAN ANALISA

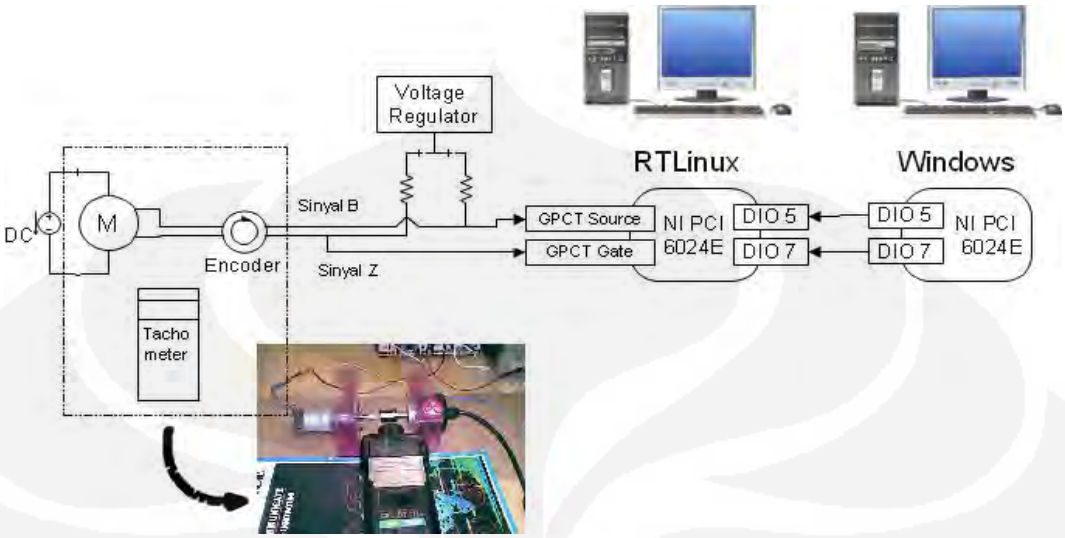
4.1 PENGUJIAN PENGUKURAN KECEPATAN PUTAR BERBASIS REAL TIME LINUX

Dalam membuktikan kelayakan dan kehandalan pengukuran kecepatan putar berbasis RTLinux ini, dilakukan pengujian dalam tiga skenario percobaan.

Dalam skenario percobaan pertama, dilakukan satu tahap pengukuran kecepatan putar terhadap motor DC merek Pittman Express model GM87 12-21. Pengukuran kecepatan putar pada motor DC dilakukan secara serentak oleh tachometer dan pengukuran kecepatan putar berbasis RTLinux ini. Referensi data pengukuran kecepatan putar ditentukan dengan mengambil sampel data pengamatan dari tachometer dan pada saat yang sama pengukuran kecepatan putar berbasis RTLinux didapat dengan mengambil nilai rata-rata dari data sebanyak 400 buah. Pengambilan data kecepatan putar dimulai dari tegangan catu motor DC sebesar 0V (*volt*). Selanjutnya, tegangan catu dinaikkan sebesar 1V dan metode pengambilan data pada proses pengukuran kecepatan putar kedua ini dilakukan sama dengan metode pengambilan data pada proses sebelumnya. Untuk proses pengukuran seterusnya, pengambilan data kecepatan putar dilakukan untuk setiap kenaikan tegangan catu motor DC sebesar 1V. Pengukuran kecepatan putar ini berlangsung hingga tegangan catu motor DC sebesar 18V.

Setelah diperoleh hasil pengukuran kecepatan putar yang diukur oleh pengukuran kecepatan putar berbasis RTLinux dan tachometer, maka perhitungan kecepatan rata-rata untuk hasil pengukuran kecepatan putar berbasis RTLinux dapat dimulai. Nilai refensi pengukuran kecepatan putar diambil dari hasil pengukuran oleh tachometer. Selanjutnya, dilakukan perhitungan untuk mencari persen kesalahan dalam pengukuran kecepatan putar berbasis RTLinux dan perhitungan untuk mencari hubungan antara pengukuran kecepatan putar berbasis RTLinux dengan pengukuran kecepatan putar oleh tachometer. Gambar 4.1

menampilkan gambar rangkaian pengukuran kecepatan putar untuk skenario percobaan pertama.

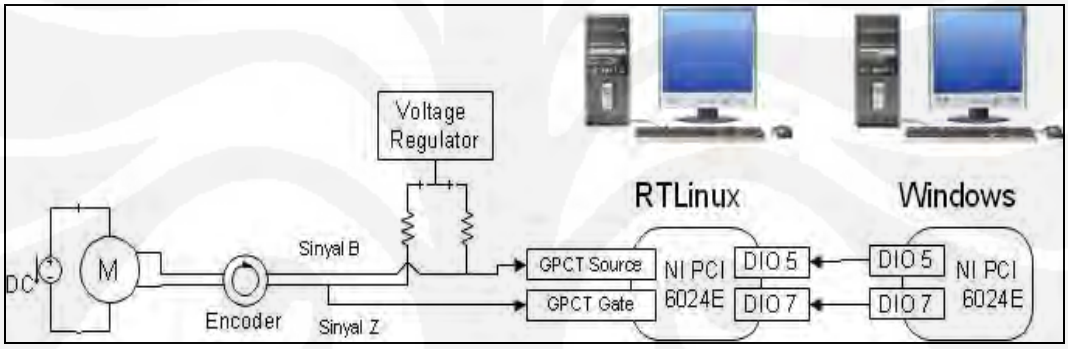


Gambar 4.1 Rangkaian Pengukuran Kecepatan Putar untuk Skenario Percobaan Pertama

Skenario percobaan kedua yang dilakukan sebagai berikut; Pengukuran kecepatan putar untuk motor DC dimulai dari tegangan catu motor DC sebesar 15V. Pada skenario tersebut, counter dikonfigurasi untuk melakukan perhitungan dalam metode *count up*. Setelah melewati waktu sekitar 10s, tegangan catu tersebut diturunkan sebesar 5V sehingga tegangan catu menjadi sebesar 10V. Penurunan tegangan catu sebesar 5V dalam setiap jangka waktu sekitar 10s ini terus berlangsung hingga tegangan yang diberikan ke motor DC sebesar 0V. Saat tegangan catu motor DC sebesar 0V maka arah perhitungan counter diubah menjadi *count down* yang mana sebelumnya adalah *count up*. Setelah itu, tegangan catu motor DC dinaikkan sebesar 5V untuk setiap jangka waktu sekitar 10s sehingga tegangan catu saat ini sebesar 10V. Kenaikkan ini terus berlangsung hingga tegangan catu motor DC sebesar 15V.

Skenario percobaan ketiga adalah kebalikan dari scenario percobaan kedua. Dalam percobaan tersebut, pengukuran kecepatan putar untuk motor DC dimulai dari tegangan catu motor DC sebesar 15V. Pada skenario tersebut, counter dikonfigurasi untuk melakukan perhitungan dalam metode *count down*. Setelah melewati waktu sekitar 10s, tegangan catu tersebut diturunkan sebesar 5V sehingga tegangan catu menjadi sebesar 10V. Penurunan tegangan catu sebesar 5V dalam setiap jangka waktu sekitar 10s ini terus berlangsung hingga tegangan

yang diberikan ke motor DC sebesar 0V. Saat tegangan catu motor DC sebesar 0V maka arah perhitungan counter diubah menjadi *count up* yang mana sebelumnya adalah *count down*. Setelah itu, tegangan catu motor DC dinaikkan sebesar 5V untuk setiap jangka waktu sekitar 10s sehingga tegangan catu saat ini sebesar 10V. Kenaikkan ini terus berlangsung hingga tegangan catu motor DC sebesar 15V. Tujuan dari dilakukannya skenario percobaan kedua dan skenario percobaan ketiga adalah untuk menguji kemampuan algoritma pengukuran kecepatan putar berbasis RTLinux dalam melakukan perhitungan kecepatan putar untuk kondisi counter sedang *count up*, diam atau *count down*. Gambar 4.2 menampilkan gambar rangkaian pengukuran kecepatan putar untuk skenario percobaan kedua dan ketiga.



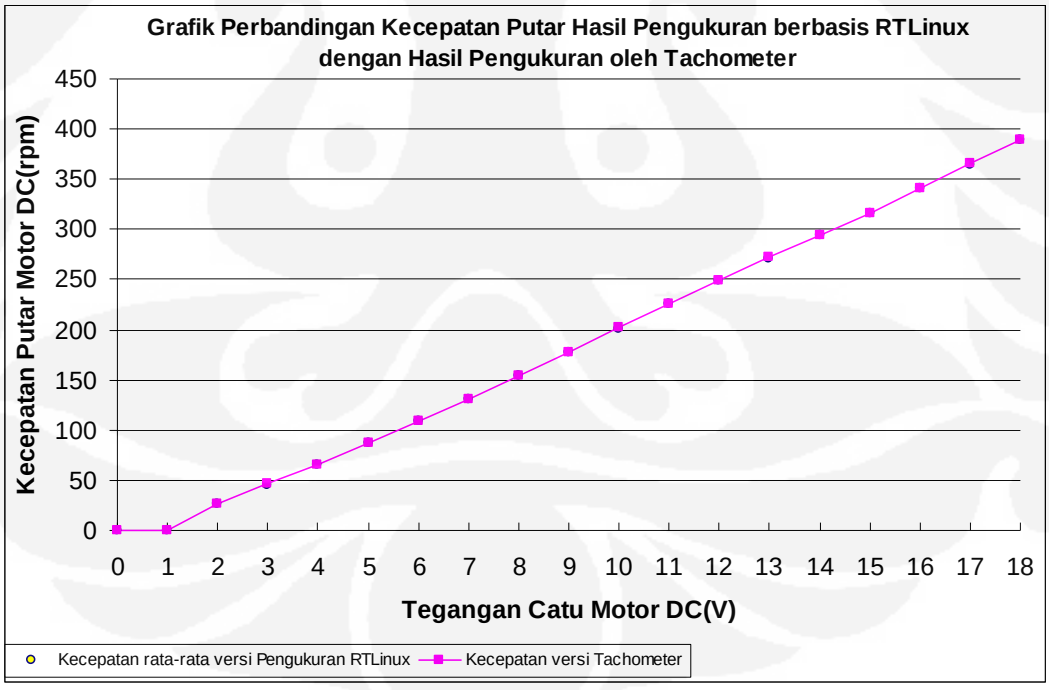
Gambar 4.2 Rangkaian Pengukuran Kecepatan Putar dan Skenario Ketiga untuk Skenario Percobaan Kedua dan Ketiga

4.2 ANALISA PERCOBAAN

Pada skenario percobaan pertama dilakukan perhitungan nilai kecepatan rata-rata untuk setiap kenaikan tegangan sebesar 1V pada 400 buah sampel data hasil pengukuran kecepatan putar berbasis RTLinux. Di skenario percobaan pertama ini, data hasil pengukuran kecepatan putar yang diperoleh dari tachometer dianggap data yang ideal. Berikut ini disajikan tabel hasil pengukuran kecepatan putar untuk tegangan catu yang berbeda-beda beserta dengan besar nilai persen kesalahannya dan gambar grafik perbandingan nilai kecepatan rata-rata hasil pengukuran kecepatan putar berbasis RTLinux dengan hasil pengukuran kecepatan putar oleh tachometer:

Tabel 4.1 Tabel Hasil Pengukuran Kecepatan Putar untuk Skenario Percobaan Pertama

Tegangan ke Motor DC (V)	Kecepatan rata-rata versi pengukuran berbasis RTLinux (rpm)	Kecepatan versi Tachometer (rpm)	Error (%)
0	0	0	0
1	0	0	0
2	27	26,9	0,866171
3	45,91692	46,2	0,612727
4	66,13357	65,3	1,27652
5	87,35021	87,8	0,512286
6	108,9001	109	0,091654
7	130,4833	130,6	0,089364
8	154,7331	154,6	0,086108
9	176,9499	176,9	0,028208
10	200,75	201,8	0,520301
11	226,0502	226	0,022208
12	248,5168	248,4	0,047008
13	271,4999	271,9	0,147152
14	294,3332	294,4	0,022706
15	316,7832	316,8	0,005292
16	340,55	340,5	0,014689
17	365,1335	365,3	0,045586
18	389,1167	389,2	0,021396



Gambar 4.3 Grafik Perbandingan Kecepatan Putar Hasil Pengukuran berbasis RTLinux dengan Hasil Pengukuran Kecepatan Putar oleh Tachometer

Persamaan yang digunakan dalam mencari besar nilai persen kesalahan dalam pengukuran kecepatan putar berbasis RTLinux sebagai berikut:

$$\% \text{ Error} = \frac{|\overline{V}_{RTLinux} - V_{tachometer}|}{V_{tachometer}} x100\% \qquad (4.1)$$

dimana: $\overline{V}_{RTLinux}$ = kecepatan rata-rata hasil pengukuran berbasis real time Linux

$V_{tachometer}$ = kecepatan hasil pengukuran tachometer

Pada tabel pengambilan data untuk kecepatan putar untuk skenario pertama, diperoleh nilai besar persen kesalahan hasil pengukuran oleh pengukur kecepatan putar berbasis RTLinux sebesar 1,27652%. Berdasarkan tabel pengukuran kecepatan putar untuk skenario percobaan pertama (4.1), diperoleh bahwa untuk tegangan catu motor DC sebesar 1V maka kecepatan putar yang diukur baik melalui pengukuran kecepatan putar berbasis RTLinux ataupun melalui tachometer sebesar 0 rpm. Hal ini karena tegangan minimum untuk motor DC Pittman Express GM87 12-21 sebesar 1,26V. Setelah melalui perhitungan untuk mencari kecepatan putar pada tegangan catu sebesar 1,26V maka diperoleh hasil perhitungan sebagai berikut:

1. Untuk hasil pengukuran kecepatan melalui tachometer, kecepatan putar motor DC tersebut sebesar 7,3 rpm.
2. Untuk hasil pengukuran kecepatan melalui pengukuran berbasis RTLinux, kecepatan putar rata-rata motor DC (400 sampel data) tersebut sebesar 7,757 rpm.
3. Jadi besar persen kesalahan yang diperoleh untuk pengukuran kecepatan putar untuk tegangan catu 1,26V sebesar 6,26%.

Secara keseluruhan pengukuran kecepatan putar berbasis RTLinux memiliki besar nilai persen kesalahan sebesar 6,26%. Berdasarkan pengamatan dari grafik perbandingan kecepatan putar hasil pengukuran berbasis RTLinux dengan hasil pengukuran oleh tachometer, dapat disimpulkan bahwa hasil pengukuran kecepatan putar yang diperoleh baik melalui tachometer maupun melalui pengukuran berbasis RTLinux hampir sama.

Akan tetapi, perlu diketahui juga bahwa pengukuran kecepatan putar berbasis RTLinux ini akan terjadi kesalahan pengukuran untuk kecepatan di bawah 6,67 rpm. Hal ini dapat dibuktikan melalui penyelesaian melalui persamaan (3.4) sebagai berikut ini:

$$\omega = \frac{60 \text{ x } 1}{100 \text{ x } 90 \text{ x } 10^{-3}} = 6,67\, rpm$$

di mana waktu deteksi untuk setiap perubahan posisi sudut sebesar 90*ms* dan nilai pulsa posisi sudut yang terukur sebanyak 1 buah. Hal ini terjadi untuk kasus misalnya kecepatan sebenarnya dari suatu motor sebesar 3 rpm, dan encoder mengeluarkan jumlah pulsa sebanyak 1 buah sehingga kecepatan yang terukur oleh pengukuran RTLinux sebesar 6,67 rpm.

Berikut ini adalah tabel data yang digunakan dalam mencari hubungan antara pengukuran kecepatan putar berbasis RTLinux dengan pengukuran kecepatan putar oleh tachometer:

Tabel 4.2 Tabel Hasil Pengukuran Kecepatan Putar Berbasis RTLinux dan Tachometer

Tegangan ke Motor DC (V)	**Kecepatan rata-rata versi pengukuran berbasis RTLinux (rpm)**	**Kecepatan versi Tachometer (rpm)**
0	0	0
1,26	7,757	7,3
2	27	26,9
3	45,91692	46,2
4	66,13357	65,3
5	87,35021	87,8
6	108,9001	109
7	130,4833	130,6
8	154,7331	154,6
9	176,9499	176,9
10	200,75	201,8
11	226,0502	226
12	248,5168	248,4
13	271,4999	271,9
14	294,3332	294,4
15	316,7832	316,8
16	340,55	340,5
17	365,1335	365,3
18	389,1167	389,2

Selanjutnya dalam mencari hubungan antara pengukuran kecepatan putar berbasis RTLinux dengan pengukuran kecepatan putar oleh tachometer tersebut maka digunakan metode *least-squares regression*. Dalam metode *least-squares regression*, digunakan model linear regresi. Persamaan umum untuk model linear regresi tersebut adalah:

$$\hat{y} = a + bx \qquad (4.2)$$

dimana $\hat{y}$: nilai estimasi

a : perpotongan yang tidak diketahui (*unknown intercept*)

b : parameter kemiringan (*slope parameter*)

Diberikan sampel data dengan $\{(x_i,y_i); i = 1, 2,...,n\}$ maka nilai a dan nilai b dapat diperoleh melalui persamaan sebagai berikut ini:

$$b = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^{\,2} - \left(\sum_{i=1}^{n} x_i\right)^2} = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2} \qquad (4.3)$$

$$a = \frac{\sum_{i=1}^{n} y_i - b\sum_{i=1}^{n} x_i}{n} = \bar{y} - b\bar{x} \qquad (4.4)$$

dimana, $\bar{y}$: nilai rata-rata dari sampel data y_i

$\bar{x}$: nilai rata-rata dari sampel data x_i

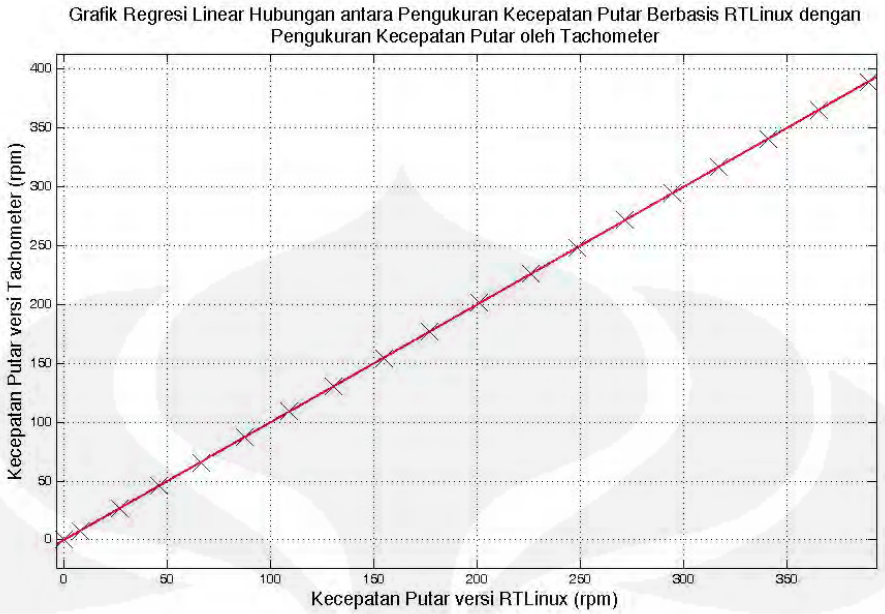
Melalui persamaan (4.2), (4.3) dan (4.4) maka diperoleh persamaan regresi linear untuk hubungan antara pengukuran kecepatan putar berbasis RTLinux dengan pengukuran kecepatan putar oleh tachometer sebagai berikut ini:

$$\hat{y} = 1,001x - 0,07607 \qquad (4.5)$$

dimana $\hat{y}$ = estimasi hasil pengukuran kecepatan putar oleh tachometer

x = hasil pengukuran kecepatan putar berbasis RTLinux

Berikut ini ditampilkan juga grafik regresi linear untuk hubungan antara pengukuran kecepatan putar berbasis RTLinux dengan pengukuran kecepatan putar oleh tachometer dalam gambar 4.4 sebagai berikut ini:



Gambar 4.4 Grafik Regresi Linear untuk Hubungan antara Pengukuran Kecepatan Putar berbasis RTLinux dengan Pengukuran Kecepatan Putar oleh Tachometer

Setelah itu dilakukan perhitungan untuk mencari nilai koefisien determinasi yang merupakan suatu pengukuran proporsi ketidak-pastian sampel data yang dapat dijelaskan oleh model estimasi. Persamaan untuk koefisien determinasi adalah sebagai berikut:

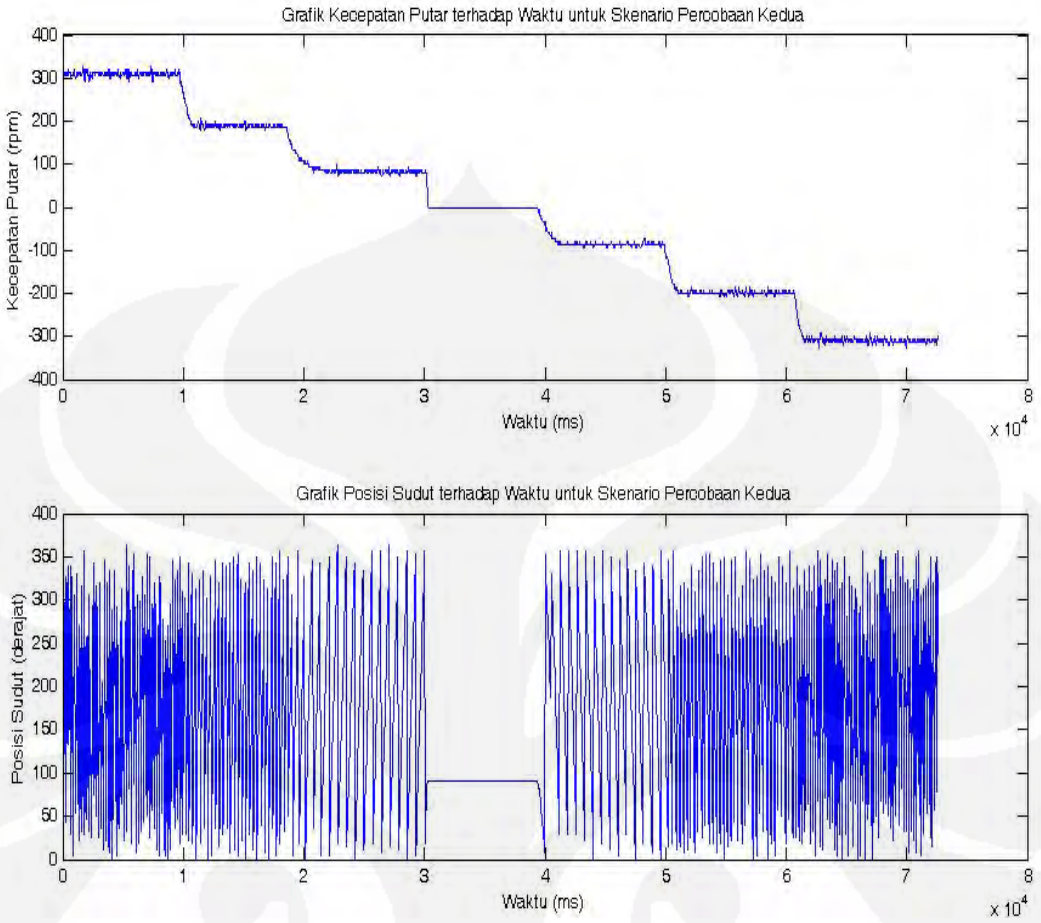
$$R^2 = 1 - \frac{SSE}{SST} \qquad (4.6)$$

dimana SSE = the error sum of squared ; SSE = $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

SST = total corrected sum of squares; SST = $\sum_{i=1}^{n}(y_i - \bar{y}_i)^2$

Untuk model regresi linear pada persamaan (4.5) diperoleh nilai koefisien determinasi sebesar 0,99999. Hal ini berarti 99,999% dari total ketidak-pastian sampel data dapat dijelaskan oleh model regresi linear tersebut.

Dalam skenario percobaan kedua, diperoleh nilai hasil pengukuran kecepatan putar dan posisi sudut dan ditampilkan dalam gambar 4.5 berikut ini:



Gambar 4.5 Grafik Kecepatan Putar terhadap Waktu dan Posisi Sudut terhadap Waktu untuk Skenario Percobaan Kedua

Dari gambar 4.5, untuk kondisi counter saat sedang *count up* maka nilai kecepatan putar yang ditampilkan bernilai positif, tetapi untuk kondisi counter saat sedang *count down* maka nilai kecepatan putar yang ditampilkan bernilai negatif. Pada gambar 4.5 untuk bagian grafik posisi sudut terhadap waktu, dapat dihitung kecepatan putar motor DC untuk setiap kenaikan 10s. Berikut ini ditampilkan perhitungan kecepatan putar motor DC dari waktu t = 0s hingga t = 30s:

1. Untuk t = 0s hingga t = 10s, diperoleh jumlah putaran sebanyak 52 putaran, sehingga frekuensi kerja motor DC saat ini sebesar 5,2 Hz. Kecepatan putar motor DC dalam satuan rpm sebesar 312 rpm.
2. Untuk t = 10s hingga t = 20s, diperoleh jumlah putaran sebanyak 31 putaran, sehingga frekuensi kerja motor DC saat ini sebesar 3,1 Hz. Kecepatan putar motor DC dalam satuan rpm sebesar 186 rpm.

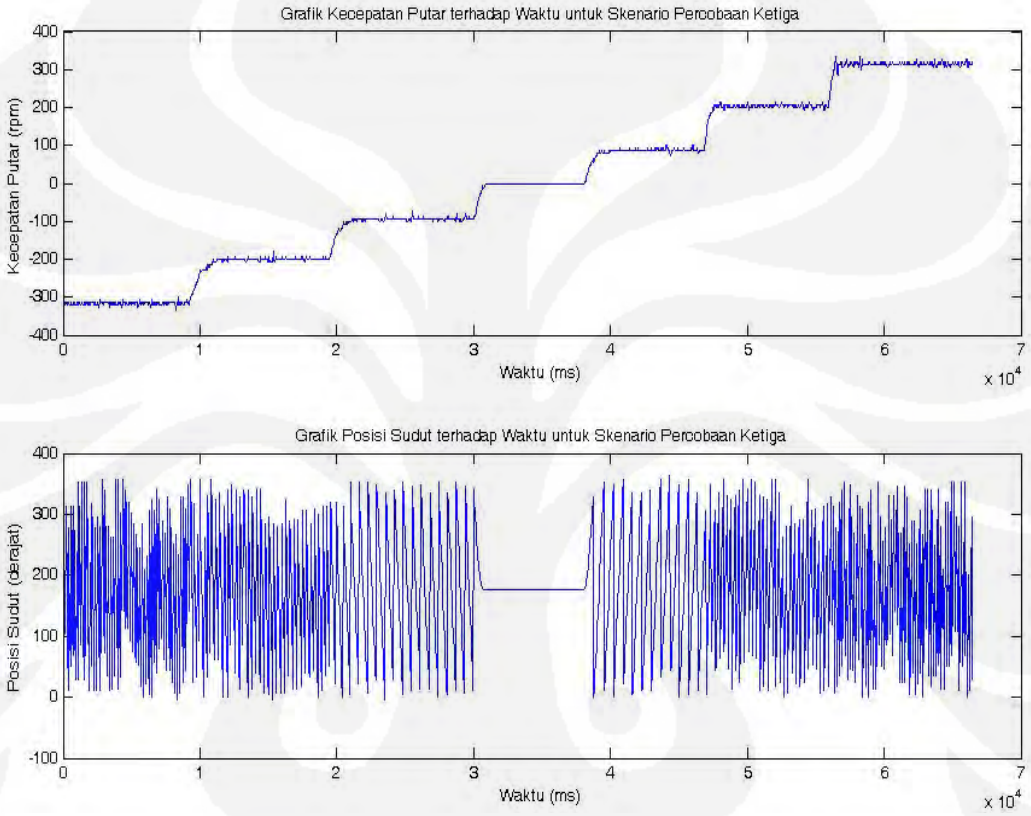
3. Untuk t = 20s hingga t = 30s, diperoleh jumlah putaran sebanyak 14 putaran, sehingga frekuensi kerja motor DC saat ini sebesar 1,4 Hz. Kecepatan putar motor DC dalam satuan rpm sebesar 84 rpm.

Dari ketiga buah perhitungan kecepatan putar di atas, dapat disimpulkan bahwa semakin berkurang kecepatan putar motor DC maka semakin berkurang frekuensinya. Selain itu, skenario percobaan kedua ini membuktikan bahwa algoritma pengukuran kecepatan putar berbasis RTLinux dapat menghitung kecepatan putar dengan tepat untuk counter yang mengalami fase perubahan dari kondisi *count up* ke kondisi diam dan kemudian ke kondisi *count down*.

Pada skenario percobaan ketiga, berdasarkan gambar 4.6, terlihat bahwa kondisi counter mula-mula sedang *count down* yang nilai kecepatan yang ditampilkan bernilai negatif, tetapi saat kondisi counter sedang *count up* maka nilai kecepatan yang ditampilkan bernilai positif. Pada gambar 4.6 untuk bagian grafik posisi sudut terhadap waktu, dapat dihitung kecepatan putar motor DC untuk setiap kenaikan 10s. Berikut ini ditampilkan perhitungan kecepatan putar motor DC dari waktu t = 0s hingga t = 30s:

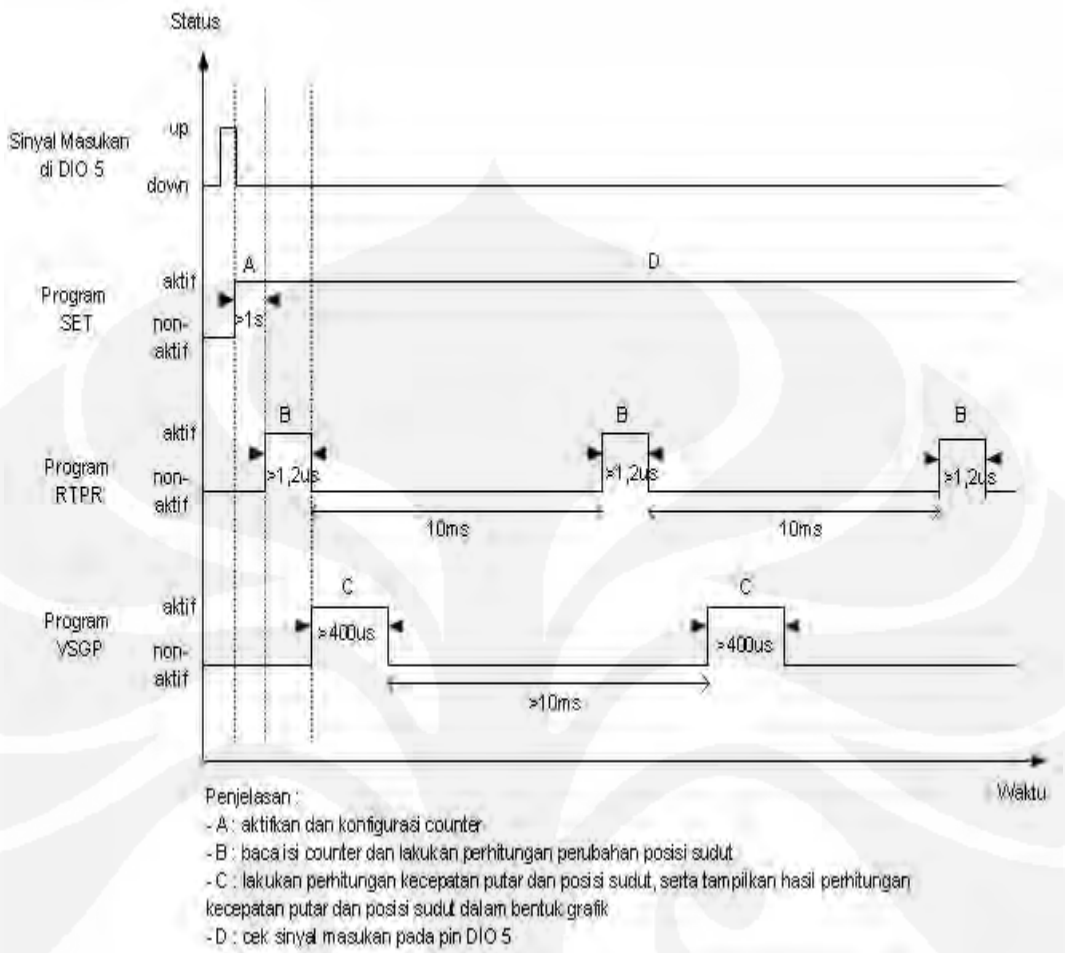
1. Untuk t = 0s hingga t = 10s, diperoleh jumlah putaran sebanyak 52 putaran, sehingga frekuensi kerja motor DC saat ini sebesar 5,2 Hz. Kecepatan putar motor DC dalam satuan rpm sebesar 312 rpm.
2. Untuk t = 10s hingga t = 20s, diperoleh jumlah putaran sebanyak 33 putaran, sehingga frekuensi kerja motor DC saat ini sebesar 3,3 Hz. Kecepatan putar motor DC dalam satuan rpm sebesar 198 rpm.
3. Untuk t = 20s hingga t = 30s, diperoleh jumlah putaran sebanyak 16 putaran, sehingga frekuensi kerja motor DC saat ini sebesar 1,6 Hz. Kecepatan putar motor DC dalam satuan rpm sebesar 96 rpm.

Dari hasil tampilan gambar grafik 4.6, maka skenario percobaan ketiga ini membuktikan bahwa algoritma pengukuran kecepatan putar berbasis RTLinux dapat menghitung kecepatan putar dengan tepat untuk counter yang mengalami fase perubahan dari kondisi *count down* ke kondisi diam dan kemudian ke kondisi *count up*.



Gambar 4.6 Grafik Kecepatan Putar terhadap Waktu dan Posisi Sudut terhadap Waktu untuk Skenario Percobaan Ketiga

Dari data hasil pengukuran kecepatan putar dari skenario percobaan dua dan skenario percobaan tiga, maka dapat digambarkan diagram pewaktuan program pengukuran kecepatan putar berbasis RTLinux dalam gambar 4.7 berikut ini:



Gambar 4.7 Diagram Pewaktuan Pengukuran Kecepatan Putar Berbasis Real Time Linux

Pada gambar 4.7 di atas, program SET dapat diaktifkan setelah pin DIO 5 menerima masukan sinyal *down*. Setelah program SET berhasil dijalankan dan berhasil mengaktifkan dan mengkonfigurasi counter, maka proses selanjutnya adalah menjalankan program RTPR. Waktu yang dibutuhkan untuk program RTPR membaca isi counter dan melakukan perhitungan nilai perubahan posisi sudut lebih besar dari 1,2 μs. Program VSGP dapat dijalankan setelah program RTPR selesai melakukan pekerjaannya. Waktu yang dibutuhkan untuk program VSGP menghitung kecepatan putar dan posisi sudut dan menampilkannya dalam bentuk grafik lebih besar dari 400 μs. Untuk proses selanjutnya, program RTPR secara periodik dijalankan terlebih dahulu sebelum program VSGP.

BAB V

KESIMPULAN

Setelah dilakukan perancangan, pemrograman dan penganalisaan hasil yang diperoleh dari pengukuran kecepatan putar berbasis RTLinux maka dapat disimpulkan sebagai berikut:

1. Ada dua alasan kuat yang menyatakan bahwa pengukuran kecepatan putar berbasis RTLinux dapat diandalkan dalam mengukur kecepatan putar suatu motor yaitu:
   a. Pengukuran kecepatan putar berbasis RTLinux memiliki persen kesalahan pembacaan sebesar 6,26%. Pengujian ini dilakukan dengan membandingkan hasil pengukuran kecepatan putar oleh pengukuran berbasis RTLinux dengan pengukuran oleh tachometer.
   b. Berdasarkan hasil pemodelan dengan menggunakan *least-square regression*, diperoleh hubungan antara pengukuran kecepatan putar berbasis RTLinux dengan pengukuran kecepatan putar oleh tachometer yang memiliki koefisien determinasi sebesar 99,999%. Hal ini membuktikan bahwa pengukuran kecepatan putar berbasis RTLinux memiliki hubungan yang linear dengan pengukuran kecepatan putar oleh tachometer.
2. Pada pengujian pengukuran kecepatan putar untuk kondisi saat counter melakukan metode perhitungan *count up* ataupun melakukan metode perhitungan *count down* terbukti bahwa algoritma pengukuran kecepatan putar berbasis RTLinux mampu melakukan pengukuran kecepatan putar dengan baik pada kedua kondisi tersebut.

DAFTAR ACUAN

[1] *6023E/6024E/6025E User Manual - Multifunction I/O Boards for PCI, PXI, and CompactPCI Bus Computers* (Texas: National Instruments Corporation, 1999).

[2] *DAQ-STC™ Technical Reference Manual System Timing Controller for Data Acquisition* (Texas: National Instruments Corporation, 1998).

[3] *Getting Started with RTLinux* (FSM Labs, Inc., 2001), hal 9.

[4] *Introduction to Linux for Real-Time Control* (National Institute of Standards and Technology: Intelligent Systems Division), hal. 32-33.

[5] Amol Lad, P. Raghavan, Sriram Neelakandan, *Embedded Linux System Design and Development* (New York: Auerbach Publications, 2006), hal. 29-33.

[6] Alan S. Morris, *Measurement and Instrumentation Principles 3rd Edition* (Oxford: Butterworth-Heinemann, 2001), hal. 392-394.

[7] Anjar Widodo. "Perbaikan Pengukuran Kecepatan Rotor Menggunakan Pengendali Fuzzy pada Sistem Pengendalian Kecepatan Motor dengan Encoder Resolusi Rendah." Skripsi, Program Sarjana Fakultas Teknik UI, Depok, 2005, hal. 15-17.

DAFTAR PUSTAKA

Andrew Krause, *Foundations of GTK+ Development* (New York: Apress, 2007)

Caleb Tennis (2004). "Data Acquisition with Comedi". Diakses 20 November 2007.
http://www.linuxjournal.com/article/7332

Christer Rosenquist. "Hard Realtime Rapid Prototyping Development Platform". Tesis, Department of Electrical Engineering, Linkőpings Universitet, Linkőping, 2003.

David Schleef, Frank Mori Hess, Herman Bruyninckx (2005). "The Control and Measurement Device Interface Handbook". Diakses 1 Agustus 2007.
http: //www.comedi.org

David Schleef. "Writing a Real-Time Compatible Device Driver for Linux Comedi".

Gregory K. McMillan, Douglas M. Considine, *Process/ Industrial Instruments and Controls Handbook 5th Edition* (New York: McGraw-Hill, 1999)

Herman Bruyninckx, *Real-Time and Embedded Guide* (Belgium, 2002)

Ismael Ripoll (2000), "Real-Time Linux (RT- Linux)". Diakses 9 September 2007.
http:\\www.nl.linuxfocus.org/English/May1998/article4.html

Kevin Dankwardt (2002). "Real Time and Linux". Diakses 20 November 2007.
http://www.linuxdevices.com/articles/AT5997007602.html
http://www.linuxdevices.com/articles/AT5503476267.html
http://www.linuxdevices.com/articles/AT6320079446.html

Matt Sherer (2002). "RTLinux Application Development Tutorial". Diakses 20 November 2007.
http://www.linuxjournal.com/article/5694

Michael Barabanov and Victor Yodaiken (1997). "Introducing Real-Time Linux". Diakses 20 November 2007.
http://www.linuxjournal.com/article/0232

Neil Matthew and Richard Stones, *Beginning Linux Programming 3rd Edition* (Indianapolis: Wiley, 2004), hal. 623-675.

Nicolas McGuire et al., *A Comparative Study on Real Time Enhanced Linux Variants* (OpenTech EDV Research GmbH, 2005)

PCI E Series Register-Level Programmer Manual - Multifunction I/O Boards for PCI Bus Computers (Texas: National Instruments Corporation, 1998).

Peter Wurmsdobler. “A Simple Control Application with Real Time Linux”. http://www.thinkingnerds.com/nerds/peterw/peterw.html

R.A. Stephan. “Real-time Linux in Control Application Area”. Tesis, Faculty of Electrical Engineering, University of Twente, Netherlands, 2002.

Ronald E. Walpole et al., *Probability & Statistics for Engineer & Scientists 7th Edition* (New Jersey: Prentice-Hall,Inc., 2002), hal. 356-367.

LAMPIRAN

Lampiran 1– Langkah-Langkah Pengaktifan Program Pengukuran Kecepatan Putar Berbasis RTLinux

Dalam mengaktifkan program pengukuran kecepatan putar berbasis RTLinux, diharuskan terlebih dahulu me-*load* modul RTLinux dan modul COMEDI yang diperlukan. Berikut ini adalah isi dari *script* yang berfungsi me-*load* modul RTLinux dan modul COMEDI:

```
#!/bin/sh

echo "====== RTLINUX START ============"
rtlinux start

echo "====== COMEDI START =============="
/sbin/modprobe comedi
/sbin/modprobe ni_pcimio
/sbin/modprobe kcomedilib
/sbin/modprobe comedi_rt_timer
/sbin/modprobe comedi_test
/sbin/modprobe comedi_bond
/sbin/modprobe comedi_fc

exit 0
```

Setelah modul tersebut berhasil di-*load* ke kernel Linux, maka langkah selanjutnya adalah mengkonfigurasi *driver* COMEDI untuk kartu NI PCI 6024E. Dalam mengkonfigurasi *driver* tersebut, hal yang harus dilakukan pertama kali adalah mengakses terlebih dahulu *folder* tempat "comedilib" di-*install*. Langkah selanjutnya mengakses file yang bernama "comedi_config" yang berada di dalam *folder* yang bernama "comedi_config". Perintah untuk mengkonfigurasi *driver* COMEDI tersebut adalah:

.comedi_config /dev/comedi0 ni_pcimio

Setelah driver COMEDI berhasil dikonfigurasi, maka program pengukuran kecepatan putar berbasis RTLinux dapat mulai diaktifkan.

Lampiran 2– Spesifikasi Kartu DAQ NI PCI 6024E

General	
Bus Type	PCI
OS Support	Windows, Linux, Real-Time, Mac OS
Product Family	E Series
Real-Time Compatible	Deterministic single-point control, Robust critical test
Triggering	Digital

Analog Input	
Number of Channels	16 SE/8 DI
Sample Rate	200 kS/s
Resolution	12 bits
Simultaneous Sampling	No
Maximum Voltage Range	-10..10 V
Number of Ranges	4
On-Board Memory	512 samples

Analog Output	
Number of Channels	2
Update Rate	10 kS/s
Resolution	12 bits
Maximum Voltage Range	-10..10 V

Digital I/O	
Number of Channels	8 DIO
Timing	Static
Logic Levels	TTL
Maximum Input Range	0..5 V
Maximum Output Range	0..5 V
Input Current Flow	Sinking, Sourcing
Current Drive (Channel/Total)	24 mA/192 mA
Watchdog Timer	No
Programmable Power-Up States	No
Handshaking I/O	No
Pattern I/O	No

Counter/Timers	
Number of Counter/Timers	2
Resolution	24 Bits
Maximum Source Frequency	20 MHz
Logic Levels	TTL
Maximum Range	0..5 V
Timebase Stability	100 ppm
GPS Synchronization	No
Pulse Generation	Yes
Buffered Operations	Yes
Debouncing/Glitch Removal	No
Number of DMA Channels	1

Physical Specifications	
Length	17.5 cm
Width	10.7 cm
I/O Connector	68-pin male SCSI-II type