Perancangan Antarmuka Instrumentasi dan Pengendalian Motor Servo Berbasis Octave

Adam Kurnia[#1], Oetomo[#2], Herry Sitepu[#3]

[#]*Departemen Teknologi Informasi, Institut Teknologi Harapan Bangsa*
Jl. Dipatiukur No. 80-84, Bandung, Indonesia

[1]adam.oktanofa@gmail.com

[2]oektomo@ithb.ac.id

[3]herry@ithb.ac.id

***Abstrak*— Aplikasi motor pada mesin menghasilkan kecepatan, ketepatan, dan kemudahan dalam bidang mekanika dan perindustrian. Motor yang dapat mengendalikan posisi sudut dan menghasilkan kecepatan secara otomatis adalah motor servo. Pengaturan posisi sudut motor servo dapat dilakukan melalui sebuah mikrokontroler yang diprogram dan dihubungkan ke motor servo. Pemrograman ini berfungsi untuk mengatur sudut, posisi, dan kecepatan yang sudah tetap dan tidak bisa diubah secara langsung. Pengubahan program secara tidak langsung menjadi kendala ketika sudut dan posisi motor servo akan diprogram secara berulang. Antarmuka dirancang dan akan digunakan sebagai pengendali motor servo secara langsung agar motor servo tidak diprogram secara berulang. Antarmuka dibuat menggunakan perangkat lunak Octave. Octave mengirim perintah ke motor servo melalui kabel serial yang menghubungkan PC dan mikrokontroller. Setelah itu, Octave akan mengakusisi data dari encoder untuk membandingkan antara masukan dan hasil keluaran pada motor servo. Data perbandingan akan digunakan sebagai *feedback* dari posisi motor servo jika sudah tepat sesuai masukan. Hasil akhir pada Tugas Akhir ini adalah perangkat lunak berupa antarmuka instrumentasi dan pengendalian motor servo berbasis Octave.**

Kata Kunci— Antarmuka, motor servo, posisi sudut, kecepatan, Octave.

Abstract*— Motor applications on a machine generates speed, accuracy, and ease in the field of mechanics and industrial. Motors that can control the angle position and automatically generate speed is servo motor. Setting the angle position of a servo motor can be done via a microcontroller that is programmed and connected to a servo motor. Programming serves to adjust the angle, position, and velocity that have been fixed and can not be modified directly. However, changing the program indirectly becomes a problem when the angle and position of the servo motor will be programmed repeatedly. The interface is designed and to be used as a directly servo motors controller so that the servo motor is not programmed repeatedly. The interface is made using the Octave software. Octave send commands to the servo motor via a serial cable which is connecting the PC and microcontroller. Then, Octave will acquire data from the encoder to compare between the input and output on the servo motor. Comparison data will be used as a feedback of servo motor position whether it is correct according to the inputs. The end result in this thesis is a software in the form of instrumentation interface and servo motor controller which is Octave-based.

***Keywords*— Interface, servo motors, position, angle, velocity, Octave.**

I. PENDAHULUAN

Motor servo adalah sebuah motor lisrtik yang dilengkapi rangkaian kendali dengan sistem *closed feedback* yang terintegrasi dalam motor tersebut. Motor servo mampu bekerja dua arah (*clock wise* dan *counter clock wise*). Arah dan sudut pergerakan rotor dikendalikan dengan memberikan pengaturan *duty cycle* sinyal *Pulse With Modulation* (PWM) pada bagian pin kontrolnya. Posisi sudut dari motor akan diumpanbalikkan ke rangkaian kontrol yang ada di dalam motor servo.

Posisi sudut motor servo dapat diatur melalui sebuah mikrokontroler yang diprogram dan dihubungkan ke motor servo. Pemrograman ini memprogram sudut, posisi, dan kecepatan yang tidak bisa diubah secara langsung. Pengubahan program secara tidak langsung menjadi kendala ketika sudut dan posisi motor servo akan diprogram secara berulang.

Masalah ini biasanya terjadi pada sebuah pabrik ketika motor servo diaplikasikan pada mesin perakitan atau pemotongan. Mesin perlu diprogram agar keluarannya menghasilkan posisi sudut dan kecepatan yang tepat untuk setiap komponen yang akan dipasang atau dipotong. Mesin akan menjadi lebih efisien dan mudah diatur ulang secara langsung oleh teknisi untuk memasang komponen jenis lainnya tanpa diprogram ulang.

Solusi dari masalah yang telah disebutkan adalah perancangan dan pengimplemetasian antarmuka yang akan digunakan sebagai pengendali motor servo secara langsung. Bagian pengaturan sudut dan kecepatan dibuat dalam bentuk variabel sehingga mikrokontroler tidak perlu diprogram ulang. Posisi sudut dan kecepatan yang belum tepat dapat diatur melalui antarmuka.

Antarmuka dirancang dan dibuat menggunakan perangkat lunak Octave. Pemrograman Octave dilakukan pada *interactive command line interface*. Secara umum, perangkat

lunak Octave dibuat sebagai tiruan dari MatLab. Antarmuka akan mengirim dan mengakuisisi data dari posisi, sudut, dan kecepatan motor servo.

Hasil dari penelitian ini adalah perangkat lunak antarmuka untuk mengirim dan mengakusisi data yang menghubungkan PC dengan motor servo. Aplikasi antarmuka yang dibuat diharapkan dapat mempermudah pekerjaan dalam pengujian posisi motor servo yang akurat secara langsung. Aplikasi ini berguna bagi mahasiswa dan dosen untuk menguji motor servo.

Pada makalah ini, kajian pustaka tentang antarmuka, komunkasi serial, dan motor servo akan dibahas pada Bab 2. analisa kebutuhan aplikasi, perancangan desain antarmuka, dan diagram sistem aplikasi yang dibuat terdapat pada Bab 3. Bab 4 menjelaskan tentang implementasi dan hasil pengukuran perangkat lunak dari pengujian yang dilakukan. Kesimpulan dari makalah dan saran-saran untuk bahan pengembangan penelitian ke depan terdapat pada Bab 5.

II. KAJIAN PUSTAKA

A. *Antarmuka*

Antarmuka (*interface*) adalah salah satu media yang disediakan sistem sebagai sarana interaksi secara langsung antara pengguna dan sistem. Antarmuka pemakai (*User Interface*) merupakan mekanisme komunikasi antara pengguna (*user*) dengan sistem. Antarmuka dapat menerima dan memberikan informasi kepada pengguna.

Antarmuka berfungsi untuk memasukkan pengetahuan baru ke dalam *knowledge base* pada sebuah sistem, menampilkan penjelasan sistem, dan memberikan panduan pemakaian sistem secara *step by step* sehingga pengguna dapat memahami sebuah sistem dengan mudah. Hal-hal terpenting dalam membangun antarmuka adalah kemudahan dalam menggunakan sistem, komunikatif, dan bersifat interaktif. Antarmuka memiliki 2 bentuk tampilan, yaitu *Command Line Interface* (CLI) dan *Graphical User Interface* (GUI)[1].

B. *Mikrokontroler ATmega16*

Mikrokontroler ATmega16 terdiri atas unit-unit fungsional secara internal seperti *Arithmetic and Logical Unit* (ALU), register dan dekoder instruksi, dan *timer* beserta komponen kendali lainnya.

Mikrokontroler ATmega16 dapat memisahkan memori program dari memori data (bus alamat dan bus data), sehingga pengaksesan data dan program dapat dilakukan secara bersamaan (*concurrent*). Secara garis besar mikrokontroler ATMega16 terdiri dari [2]:

a. Arsitektur RISC dengan *throughput* mencapai 16 MIPS pada frekuensi 16Mhz.
b. Memiliki kapasitas Flash memori 16Kbyte, EEPROM 512 Byte, dan SRAM 1Kbyte
c. Saluran I/O 32 buah, yaitu *port* A, *port* B, *port* C, dan *port* D.
d. CPU yang terdiri dari 32 buah register.
e. *User* interupsi internal dan eksternal
f. *Port* antarmuka SPI dan *Port* USART sebagai komunikasi serial
g. *Non-volatile program memory*
h. Fitur *Peripheral*:
   - Tiga buah *timer/counter* dengan kemampuan perbandingan (*compare*). *Timer* terdiri dari dua buah 8-bit *timer/counter* (*prescaler* terpisah dan mode *compare*) dan sebuah 16-bit *timer/counter* (*prescaler* terpisah, mode *compare*, dan mode *capture*)
   - 8 kanal ADC yang terdiri dari delapan *single-ended channel* (register ADCH dan ADCL) dan dua *differential channel* dengan *programmable gain*.
   - *Real time counter* dengan osilator tersendiri.
   - Empat kanal PWM dan Antarmuka komparator analog.
   - *Byte-oriented two-wire serial interface*.
   - *Watchdog timer* dengan osilator internal.
   - *Watchdog timer* dengan osilator internal.

ATMega16 dengan model 40 pin *dual in-line package* (DIP) memiliki 8 pin untuk masing-masing *port* (*port* A, *port* B, *port* C, dan *port* D). Pin-pin selain dari *port* A, B, C,dan D merupakan pin pendukung *pheriperal* ATmega16 seperti pin untuk catu daya, *grounding*, dan lain-lain.

C. *Protokol Kontrol Data Link*

Sebuah protokol data diperlukan karena adanya kemungkinan *error* pada proses pengiriman data [6]. Data yang dikirimkan diubah menjadi bit-bit tertentu agar mempermudah dalam pengirimannya. Kemungkinan *error* pada bagian penerima data juga bisa terjadi karena adanya penyusunan data dan sinkronisasi agar data yang diterima utuh dan sesuai dari sumber pengirimnya. Adanya kemungkinan *error* pada bagian pengirim dan penerima data dapat diatasi dengan menambahkan *layer* (lapisan) kontrol data yang berfungsi sebagai *flow control*, *error detection*, dan *error control* [6]. Lapisan kontrol ini disebut protokol kontrol data *link* (*data link control protocol*).

Dalam komunikasi data, *flow control* adalah proses pengelolaan laju transmisi data antara dua titik untuk mencegah pengirim terlalu cepat kehabisan data dan penerima lambat dalam menerima data. *Flow control* diperlukan jika pengirim mengirimkan informasi pada tingkat yang lebih cepat dari penerima untuk menerima dan memproses data. Tanpa *flow control*, buffer dari receiver akan penuh hingga *overflow* ketika sedang memproses data yang diterima sebelumnya. Ketika data diterima, sejumlah proses harus dilaksanakan sebelum *buffer* dapat dikosongkan dan siap menerima data lainnya.

D. *Komunikasi Serial*

Komunikasi serial merupakan suatu proses pengiriman data secara sekuensial atau satu persatu melalui sebuah kanal informasi [4]. Komunikasi ini memiliki kecepatan komunikasi yang rendah tetapi sangat mendukung untuk komunikasi jarak jauh. Komunikasi serial memiliki beberapa parameter yang harus ditentukan yaitu: *baud rate* (kecepatan transmisi data), *start bit*, *data bit*, *parity bit* (terdiri dari *odd* dan *even parity* yang digunakan untuk *error cheking*), dan *stop bit*. Format data yang digunakan pada komunikasi serial adalah 1 *bit start*

(*low*), 8 *bit* data, dan 1 *bit* stop (*high*). Format data ditunjukkan pada Gambar 1. Kondisi *low* setelah *stop bit* pada adalah *start bit* yang menandakan data berikutnya akan dikirimkan. Jika tidak ada lagi data yang ingin dikirim, maka jalur transmisi ini akan tetap dalam keadaan high. Keadaan *low* yang jeda waktunya cukup lama untuk mengirimkan 8-bit data disebut '*Break Signal'*. Jika pengirim menyebabkan jalur komunikasi dalam keadaan seperti ini, penerima akan menganggap ini adalah '*break signal'* atau sinyal rusak.

Data yang dikirimkan dengan konsep seperti pada Gambar 1 disebut data yang terbingkai (*to be framed*) oleh *start* dan *stop bit*. Jika *stop bit* dalam keadaan *low*, berarti telah terjadi *framing error*. Hal ini terjadi karena perbedaan kecepatan komunikasi antara pengirim dengan penerima.

Komunikasi serial merupakan fitur penting dalam perancangan sistem, karena dapat menghubungkan mikrokontroler dengan perangkat lainya. Komunikasi serial dapat dilakukan dengan menggunakan sebuah media trasmisi berupa kabel serial dan USB *to serial*.

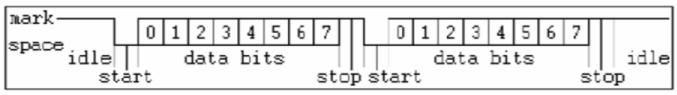
Penggunaan mikrokontroler atau *single chip microprocessor* juga sudah populer, beberapa di antaranya sudah dilengkapi dengan *Serial Communication Interface*, misalnya prosesor Intel 8051 yang dilengkapi dengan *built-in* USART. Prosesor dapat berkomunikasi dengan prosesor lain melalui 2 *port*, yaitu TxD dan RxD.

Transmisi data komunikasi serial dibedakan menjadi 2 macam, yaitu komunikasi data sinkron dan komunikasi data asinkron. Perbedaannya terletak pada *clock* pendorong data. Dalam komunikasi data seri sinkron, *clock* untuk *shift register* dikirimkan bersama data seri. Pada komunikasi data asinkron, *clock* untuk *shift register* tidak ikut dikirim. Rangkaian penerima data harus dilengkapi dengan rangkaian yang mampu membangkitkan *clock* yang diperlukan. Upaya agar penerima data dapat membangkitkan *clock* yang dapat digunakan untuk mendorong *shift register* penerima adalah bagian terpenting dari komunikasi seri asinkron [3].

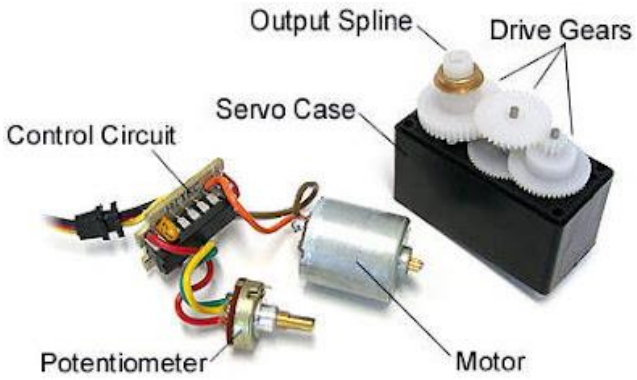
E. *Motor Servo*

Motor servo adalah sebuah motor listrik searah yang dilengkapi rangkaian kendali dengan sistem *closed feedback* yang terintegrasi dalam motor tersebut [5]. Motor servo disusun dari sebuah motor DC, *gearbox*, variabel resistor (VR) atau potensiometer dan rangkaian kontrol. Susunan dan komponen motor servo ditunjukkan pada Gambar 2. Potensiometer berfungsi untuk menentukan batas maksimum putaran sumbu (*axis*) motor servo. Sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang pada pin kontrol motor servo.

Pada bagian luar motor servo terdapat beberapa komponen seperti housing motor servo, konektor kabel, lubang sekrup, dan jangkar. Posisi putaran sumbu (*axis*) dari motor akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor servo mampu bekerja dengan arah putaran searah jarum jam dan berlawanan dengan jarum jam. Arah dan sudut pergerakan rotornya dapat dikendalikan dengan memberikan pengaturan *duty cycle* sinyal PWM pada bagian pin kontrolnya.



Gambar 1 Format data komunikasi serial



Gambar 2 Bagian dalam motor servo.

Motor servo terbagi menjadi dua jenis berdasarkan arusnya, yaitu motor servo AC dan DC. Motor servo berjenis AC dapat menangani arus yang tinggi atau beban yang berat, sehingga motor servo AC banyak diaplikasikan pada mesin-mesin industri, sedangkan motor servo DC digunakan pada aplikasi-aplikasi yang lebih kecil.

III. ANALISIS DAN PERANCANGAN

A. *Analisis*

1) *Analisis Sistem Antarmuka Berbasis Octave*

Motor servo dapat dikendalikan dengan menghubungkan kabel sinyal pada salah satu *port* mikrokontroler dan diatur sebagai keluaran dari mikrokontoler. Kabel yang digunakan untuk transmisi sinyal kontrol hanya dapat menerima masukan sinyal kontrol dari mikrokontroler.

Sebuah antarmuka dirancang untuk memudahkan pengguna menjalankan sistem yang dibuat. Agar mikrokontroler dapat diprogram ulang dengan mudah, pada program perintah untuk mengatur motor servo perlu diberikan variabel data untuk menyimpan data masukannya. Variabel data diperlukan agar masukan dari pengguna dapat langsung masuk ke program sistem dan menggerakkan motor servo setelah program dieksekusi.

Data yang dikirimkan dan diakuisisi pada sistem yang dibuat membutuhkan sebuah protokol data. Sebuah protokol data diperlukan karena adanya kemungkinan *error* pada proses pengiriman dan penerimaan data. Data yang dikirimkan diubah menjadi bit-bit tertentu dan disusun sedemikian rupa agar mempermudah pengiriman dan penerimaan data.

Sistem antarmuka berbasis Octave membutuhkan proses pengiriman dan pengakuisisian data. Jika motor servo yang digunakan membutuhkan spesifikasi untuk mengirim sekaligus mengakuisisi data posisi sudut, dan kecepatan, maka motor servo sederhana tidak dapat memenuhi spesifikasi sistem perancangan. Oleh karena itu, motor servo akan dirancang dan dirangkai dari DC motor yang akan

dihubungkan dengan perangkat lainnya hingga dapat memenuhi sepsifikasi sistem.

2) Analisis Spesfikasi Sistem

Untuk membangun sistem antarmuka berbasis Octave dibutuhkan beberapa spesifikasi sistem. Spesifikasi sistem yang dibutuhkan berupa perangkat lunak dan perangkat keras. Spesifikasi yang dibutuhkan sistem adalah sebagai berikut:

a. Protokol data

Kemungkinan *error* yang akan terjadi pada penerima dan pengirim dapat diperkecil kemungkinannya dengan protokol data. Protokol data juga dapat mengatur urutan dan isi data yang sedang ditransmisikan.

b. Perangkat lunak Octave

Perangkat lunak Octave digunakan pada PC yang berfungsi sebagai pengirim dan pengakuisisi data. Octave akan dipasang pada sistem operasi Ubuntu. Antarmuka yang akan dirancang juga dibuat menggunakan perangkat lunak Octave.

c. Perangkat keras

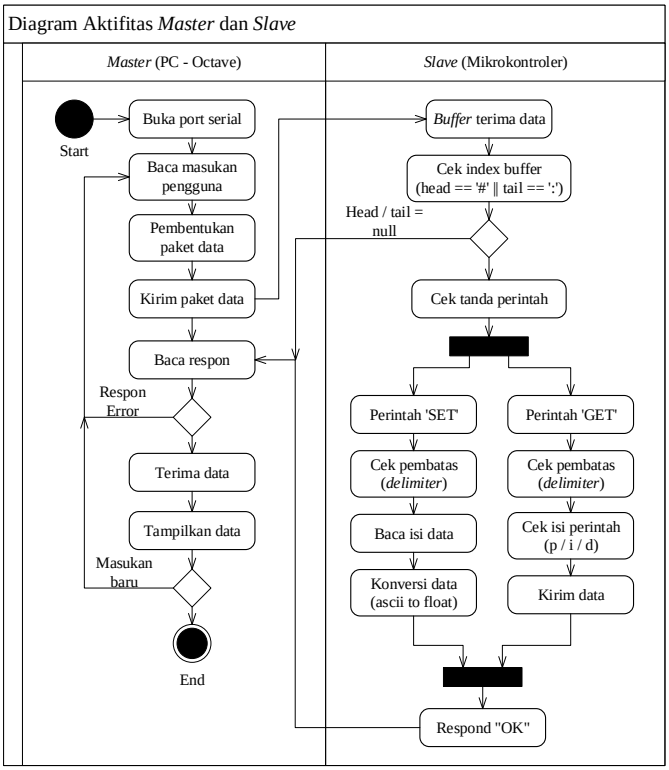
Perangkat keras yang dibutuhkan pada sistem antara lain adalah PC, mikrokontroler, *driver* motor H-*bridge*, motor DC, potensiometer, dan *serial to USB converter*.

B. *Perancangan Sistem*

Berdasarkan analisis spesifikasi kebutuhan yang telah dibahas sebelumnya, maka dilakukan perancangan sistem antarmuka berbasis Octave. Diagram status antara *master* dan *slave* yang menggambarkan sistem seceara umum ditunjukkan pada Gambar 3. Diagram aktifitas digunakan untuk menggambarkan aktifitas yang terjadi pada PC dan mikrokntroler. Pengguna dapat memberi masukan dan mendapat respon dengan baik format paket data yang dikirimkan sesuai dengan format paket data yang telah ditetapkan.

Pada bagian *slave*, data diterima dari PC dan tersimpan oleh *buffer* data. *Buffer* data akan dicek per-indeksnya untuk mencari *head* dan *tail* dari paket data yang diterima. Jika terdapat *head* dan *tail* pada paket datanya, maka proses selanjutnya adalah mengecek perintah yang diberikan. Tanda perintah '*s*' berfungsi untuk mengisi variable data pada mikrokontroler dan tanda perintah '*g*' berfungsi untuk mengambil nilai dari variable. Setelah proses *parsing* selesai, mikrokontroler akan mengirimkan respon 'Ok' ke PC sebagai tanda untuk PC dapat melakukan pengiriman data kembali. Respon '*error'* dikirimkan ke PC apabila terjadi kesalahan pada proses *parsing.* Jika sudah selesai, pengguna dapat mengakhiri program.

Pengguna dapat memasukan perintah utuk dapat membuka dan mengatur *port* serial, mengirim perintah, dan mendapatkan umpan balik dari data masukan. Urutan pertama sebelum memasukkan perintah untuk mengirim dan menerima data adalah pengaturan untuk *port* serial. Jika port serial tidak terdeteksi maka pengguna mengatur ulang *port* serial hingga terdeteksi oleh PC. Setelah terdeteksi, pengguna dapat memasukan perintah untuk mengirim data atau mengambil data yang tersimpan pada mikrokontroler.



Gambar 3 Bagian dalam motor servo.

IV. IMPLEMENTASI DAN PENGUJIAN

A. *Implementasi Program*

Implementasi program merupakan proses penerjemahan hasil perancangan sistem yang telah dilakukan. Rancangan sistem yang telah dibuat akan dibentuk ke dalam bentuk kode pada Octave dan mikrokontroler. Implementasi sistem juga didukung oleh perangkat keras dan perangkat lunak yang sesuai agar sistem dapat digunakan secara efektif dan efisien. Perangkat keras dan perangkat lunak sistem yang diimplementasikan dalam penelitian ini adalah komputer (PC) sebagai terminal masukan dan keluaran, USB-*to-serial*-TTL sebagai penghubung antara PC dan mikrokontroler, dan mikrokontroler ATmega16.

Octave yang digunakan dalam implemetasi ini sudah terpasang dengan *package* "*instrument-control*". *Package* ini berfungsi untuk dapat mengirim dan menerima data dari mikrokontroler menggunakan *port* serial. PC sebagai *master* akan mengirimkan perintah ke mikrokontroler untuk mengirimkan data dan mengakuisisi data. Data yang diterima pada mikrokontroler akan ditampung pada *buffer*. Setelah data diterima, proses selanjutnya adalah *parsing* paket data. Pembentukan paket data dalam bentuk *array of string* dilakukan dengan mengubah tipe data *number* menjadi *string* terlebih dahulu pada setiap masukannya. Konversi dari *number* menjadi *string* menggunakan fungsi "num2str" pada Octave. Setelah data dikonversi menjadi *string*, paket data dibentuk dengan menggunakan fungsi "*strcat*" untuk

menyatukan data yang telah dikonversi dengan *head*, *tail*, dan *delimiter*. Data dikirimkan ke mikrokontroler dengan menggunakan fungsi "*srl_write*" setelah paket data terbentuk.

B. *Pengujian Fungsional Sistem Atarmuka*

Pengujian fungsional bertujuan untuk menjamin perangkat lunak yang dibangun memiliki kualitas yang handal. Pengujian juga dilakukan untuk mengidentifikasi cacat atau masalah dalam proses pembangunan sebuah aplikasi. Pengujian dapat dilakukan jika mikrokontroler sudah terhubung dengan PC menggunakan USB-*to*-serial-TTL. Mikrokontroler harus sudah memiliki catu daya agar komunikasi antara PC dan mikrokontroler dapat berjalan. Pin Tx pada USB serial dihubungkan dengan pin PD0 pada mikrokontroler. Pin Rx pada USB serial dihubungkan dengan pin PD1 pada mikrokontroler. Jika sudah dihubungkan dengan benar, maka pengujian aplikasi sistem antarmuka dapat dilakukan. Pengujian dilakukan dengan cara memberikan sejumlah masukan (valid dan tidak valid) pada sistem dan memperhatikan keluaran yang dihasilkan oleh sistem. Hasil keluaran dari sistem kemudian diproses sesuai dengan kebutuhan fungsionalnya.

1) Pengujian Fungsional Menggunakan Hyperterminal

Pengujian sistem pengiriman dan pengakuisisian data dilakukan dengan menggunakan *hyperterminal* RealTerm terlebih dahulu sebelum dilakukan pengujian pada antarmuka Octave. Pengujian awal dengan menggunakan *hyperterminal* berfungsi untuk meminimalisir adanya kesalahan *error* pada saat pengujian sistem antarmuka Octave. Hasil pengujian fungsional menggunaan *hyperterminal* ditunjukkan pada Tabel 1.

2) Pengujian Fungsional Menggunakan Octave

Pada pengujian fungsional dengan menggunakan antarmuka Octave, pengguna tidak perlu mengetikkan format paket data. Pengguna cukup memberi masukan perintah dan nilai yang akan dikirimkan melalui antarmuka Octave. Kesalahan dalam pengiriman dan pengakuisisian paket data saat pengujian dapat diminimalisir oleh sistem aplikasi antarmuka yang dirancang menggunakan Octave. Hasil pengujian fungsional menggunaan Octave ditunjukkan pada Tabel 2.

V. Analisis Hasil Pengujian Sistem Antarmuka Berbasis Octave

Berdasarkan hasil dari seluruh pengujian fungsional menggunakan *hyperterminal* dan perangkat lunak Octave pada Tabel I dan Tabel II, sistem antarmuka berbasis Octave berjalan dengan baik sesuai perintah dan masukan dari *user*. Selain itu, tidak terdapat kesalahan tampilan/ antarmuka untuk setiap perintah dari masukan pengguna.

Tabel I

Hasil Pengujian Fungsional Menggunakan *Hyperterminal*

Jenis Perintah	Langkah Pengujian	Hasil yang Diharapkan	Status
SET	Ketik dan kirim paket data bilangan desimal #s;1.234;3.456;6.789:	RealTerm menampilkan respon "k"	OK
SET, GET	Kirim paket data tanpa tanda *head* ('#)	RealTerm menampilkan respon "r"	OK
SET, GET	Kirim paket data tanpa tanda *tail* (:)	RealTerm menampilkan respon "r"	OK
Jenis Perintah	**Langkah Pengujian**	**Hasil yang Diharapkan**	**Status**
SET, GET	Kirim paket data tanpa tanda *head* dan *tail* (# dan :)	RealTerm menampilkan respon "r"	OK
SET, GET	Kirim paket data tanpa atribut paket data	RealTerm menampilkan respon "r"	OK
GET	Ketik dan kirim paket data #g;p;i;d:	RealTerm menampilkan respon paket data dengan isi data desimal dua angka dibelakang tanda koma.	OK
GET	Ketik dan kirim paket data #g;p;i;d:	RealTerm menampilkan respon paket data dengan nilai isi data adalah nol.	OK

Tabel II

Hasil Pengujian Fungsional Menggunakan Octave

Langkah Pengujian	Hasil yang Diharapkan	Status
Ketik "set" pada baris perintah Octave	Aplikasi menampilkan perintah untuk memasukkan nilai kP	OK
Ketik "get" pada baris perintah Octave	Aplikasi menampilkan nilai dari kP, kI, dan kD dengan masing-masing nilai 0.	OK
Ketik "ae" selain perintah "set" dan "get" pada baris perintah Octave	Aplikasi menampilkan pesan error dan pengguna keluar dari aplikasi	OK
Ketik "get" pada baris perintah Octave	Aplikasi menampilkan nilai dari kP, kI, dan kD.	OK

Langkah Pengujian	Hasil yang Diharapkan	Status
Memasukkan nilai desimal pada kP dan kD	Aplikasi menampilkan pesan "Respond: OK" dan paket data yang dikirim	OK
Ketik "get" pada baris perintah Octave	Aplikasi menampilkan nilai dari kP, kI, dan kD dengan masing-masing nilai dibulatkan dua angka dibelakang koma.	OK
Memasukkan tanda (simbol) selain dari angka	Aplikasi menampilkan pesan *error* dan pengguna keluar dari aplikasi	OK

VI. KESIMPULAN

Berdasarkan uraian yang telah dikemukakan pada analisis, implementasi, pengujian, dan pembahasan sebelumnya, maka dapat ditarik kesimpulan bahwa perancangan dan pengimplementasian antarmuka instrumentasi berhasil dilakukan dan berjalan dengan baik. Pengiriman dan pengakuisisian data dilakukan dengan menggunakan aplikasi antarmuka. Data yang dikirimkan dan diterima ditampilkan menggunakan tampilan CLI. Format paket data ditentukan dan dibentuk dalam aplikasi antarmuka untuk mengurangi kesalahan dalam proses komunikasi data.

DAFTAR REFERENSI

[1] Akinari, "Definisi *Interface* (Antarmuka) dan Contohnya", Internet: http://www.caralengkap.com/2012/10/definisi-interfaceantar-muka-dan.html [10 Oktober 2014]

[2] A. E. Putra, "Modul-1: ATMega16 dan Bascom AVR", 2010.

[3] H. Andrianto, *Komunikasi Serial USART pada Pemrograman Mikrokontroler AVR ATMega16 Menggunakan Bahasa C,* Informatika, Bandung, 2013, pp. 123-126.

[4] Kustanto, "Komunikasi Serial", 2011.

[5] "Motor Servo", Internet: http://elektronika-dasar.web.id/teori-elektronika/motor-servo/ [7 Oktober 2014]

[6] W. Stallings, "Data Link Control Protocols," in *Data and Computer Communications*, 8th ed., Prentice Hall, 2007, pp. 207-225.

Adam Kurnia Oktanofa, kelahiran Bandung, Jawa Barat tahun 1993, menyelesaikan S1 di Jurusan Teknik Elektro (*Mobile Technology*) ITHB pada Agustus 2015. Minat penelitian pada mikrokontroler dan aplikasi berbasis Android.

Oetomo, menempuh pendidikan S1 di Teknik Elektro Universitas Udayana, Bali. Gelar magister diperoleh di Jurusan Teknik Elektro Institut Teknologi Bandung. Minat penelitian pada bidang sistem kendali dan robotika.

Herry Imanta Sitepu, menempuh pendidikan S1 di Teknik Elektro ITB dan lulus tahun 1999, dan memperoleh gelar magister dan doktor di jurusan yang sama di ITB. Sejak tahun 2006 aktif sebagai pengajar di Prodi Sistem Komputer ITHB. Minat penelitian: *computer networking*, *programming* dan *distributed system*.