Intergrasi Arduino -OPC *Server*-Modem GSM pada Sistem Pengontrolan Lampu dan *Air Conditioner* Melalui Fasilitas HMI dan SMS

Herdiawan, Parsaulian I. Siregar dan Agus Samsi

Program Studi Teknik Fisika – Institut Teknologi Bandung

Abstrak

Tujuan penelitian Ini adalah merancang dan membuat sebuah demoset yang terintegrasi antara Arduino-OPC server-Modem GSM yang digunakan untuk pengontrolan lampu dan (*Air Conditioner*) AC melalui fasilitas HMI dan SMS. Arsitektur sistem demoset ini terbagi menjadi tiga level. Level pertama merupakan field layer yang terdiri dari lampu dan AC. Level kedua merupakan layer pengontrol yang terdiri dari modem GSM, Arduino gateway, dan Arduino slave. Level ketiga merupakan layer supervisi yang terdiri dari HMI dan *celluler phone*.

Alur kerja secara sederhana dari sistem ini bermula dari Arduino slave yang dapat mengakses data dari lampu dan AC, data tersebut akan diteruskan ke Arduino gateway yang akan terhubung dengan OPC server dan modem GSM. Arduino Gateway akan mengatur kemana data tersebut akan diberikan, apakah OPC server atau modem GSM. OPC server dan modem GSM akan terhubung ke masing-masing client, yakni HMI ataupun *celluler phone*. Arah komunikasi dari setiap sistem yang terhubung bersifat dua arah, artinya OPC server dan HMI dapat mengakses data status lampu dan AC, sekaligus mengubah status tersebut melalui HMI dan *celluler phone*.

Dari hasil pegujian, rata-rata waktu pemrosesan data dari mulai dikirimnya perintah dari Arduino Gateway ke Arduino slave sampai diterimanya respons dari Arduino slave adalah 31,8 milidetik. Sementara itu untuk waktu pemrosesan data dari mulai diterimanya SMS oleh modem GSM sampai dikirimnya balasan SMS oleh modem GSM adalah 2,070 detik. Masing-Masing waktu pemrosesan data dihitung dengan mengambil data sebanyak 10 kali.

Kata Kunci : Modbus, Master, Slave, OPC server, HMI, Arduino gateway.

1 Pendahuluan

Pada penelitian ini akan dikembangkan sebuah demoset yang digunakan untuk melakukan monitoring dan pengontrolan pada penggunaan lampu dan AC. Kedua perangkat elektronik tersebut dipilih dikarenakan keduanya sangat banyak digunakan di sebuah bangunan. Selain itu AC juga merupakan salah satu perangkat yang mengonsumsi cukup banyak energi pad sebuah bangunan. *Monitoring* dilakukan dengan melihat status penggunaan kedua lampu dan AC tersebut, apakah berada pada kondisi menyala atau mati. Sementara itu, pengontrolan yang dilakukan hanya terbatas pada pengontrolan on-off, yakni untuk menyalakan dan mematikan lampu dan kompresor pada AC. Kompresor dipilih sebagai elemen yang dikontrol pada AC dikarenakan kompresor merupakan elemen utama pada sebuah AC yang harus diatur penggunaannya agar terhindar dari kerusakan. Selain itu kompresor juga merupakan elemen yang paling banyak mengkonsumsi energi pada AC, sehingga dari segi energi listrik yang dikonsumsi, mengontrol penggunaan kompresor tidak berbeda jauh dengan mengontrol penggunaan AC secara keseluruhan.

Proses *monitoring* dan pengontrolan akan dilakukan melalui HMI dan SMS. HMI digunakan untuk monitoring dan pengontrolan jarak dekat, sedangkan SMS dimanfaatkan untuk pengontrolan dan monitoring jarak jauh. Untuk memudahkan proses *monitoring* dan pengontrolan maka akan dilakukan sebuah sistem penjadwalan, yakni sebuah perintah yang muncul secara otomatis pada jam-jam tertentu, seperti mengirim status lampu dan AC. Penjadwalan terbagi menjadi dua bagian, yang pertama adalah penjadwalan yang

dilakukan untuk menyalakan dan mematikan kompresor AC secara bergantian dalam rentang waktu tertentu. Sementara penjadwalan kedua dilakukan untuk menyalakan lampu dan AC pada pagi hari dan mematikannya pada sore hari.

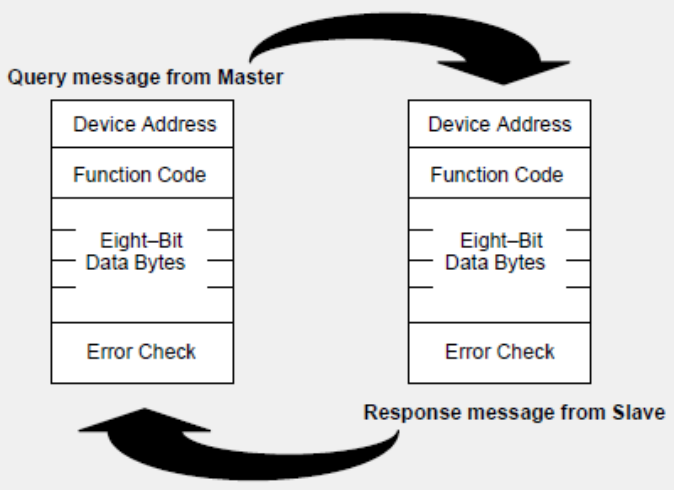
2 Teori Dasar

2.1 Modbus

Cara kerja dari komunikasi data berbasis protokol Modbus menggunakan prinsip *master-slave/server-client*. Master akan menginisisasi proses transfer data dengan memberikan sebuah perintah atau "*Query*". Sementara itu *slave* akan memberikan jawaban atas *Query* tersebut, dalam istilah Modbus jawaban tersebut disebut "response". Protokol Modbus memiliki standar dalam proses pengiriman data, data-data yang dikirim akan dikumpulkan dalam sebuah paket data yang disebut "pesan". Pesan yang dikirim menggunakan protokol Modbus memiliki struktur yang khas. Struktur pesan pada protokol Modbus mengharuskan adanya alamat pengirim dan alamat yang dituju. Selain itu pada protokol modbus juga terdapat metode pendeteksian *error* pada data. Modbus yang ditransmisikan secara serial memiliki 2 mode [6], yakni mode RTU dan mode ASCII. Pada penelitian ini mode Modbus yang digunakan adalah mode ASCII.

2.1.1 Query-Response Cycle

Protokol Modbus bekerja berdasarkan prinsip Master-*slave*, Master akan memberikan *Query* dan *slave* yang akan memberikan respon. Sementara itu, *Query* dan respon merupakan sebuah siklus yang berkaitan, *slave* tidak akan memberikan sebuah respon ketika tidak ada *Query* dari master. Dalam siklus *Query*-respon, pesan yang dikirim memiliki urutan tertentu.



Gambar 1 Query-Response Cycle [6]

Pada gambar 1 terlihat setiap pesan yang dikirim baik pada *Query message* ataupun *response message* diawali terlebih dahulu dengan alamat *device*. Kemudian dilanjutkan dengan *function code* yang berisi jenis perintah apa yang harus dilakukan oleh *slave*, setiap *function code* memiliki interpretasi perintah yang berbeda-beda. Sementara itu *data bytes*

berisi informasi tambahan yang dibutuhkan *slave* untuk melakukan perintah tertentu. Kemudian untuk memvalidasi pesan yang diolah maka dilakukanlah sebuah pengecekan *error* untuk memastikan susunan data pada pesan yang diterima oleh *slave* sesuai dengan data yang dikirim oleh *master*. Setelah *Query message* dari *master* selesai dikirim maka *slave* akan merespon pesan tersebut. Jika *slave* memberikan respon yang normal, maka struktur pesan yang dikirim oleh *slave* akan berisi alamat device dan *function code* yang sama dengan struktur pesan yang dikirim oleh *master* [6].

2.1.2 Struktur Pesan *Modbus*

Protokol Modbus memiliki sebuah standard untuk struktur pesan yang dikirim dalam proses transmisi data. Setiap pesan yang dikirim terdiri dari beberapa frame yang memiliki fungsi masing-masing. Tabel 1 menjelaskan frame yang terbentuk saat sebuah pesan dikirim dengan menggunakan protokol Modbus

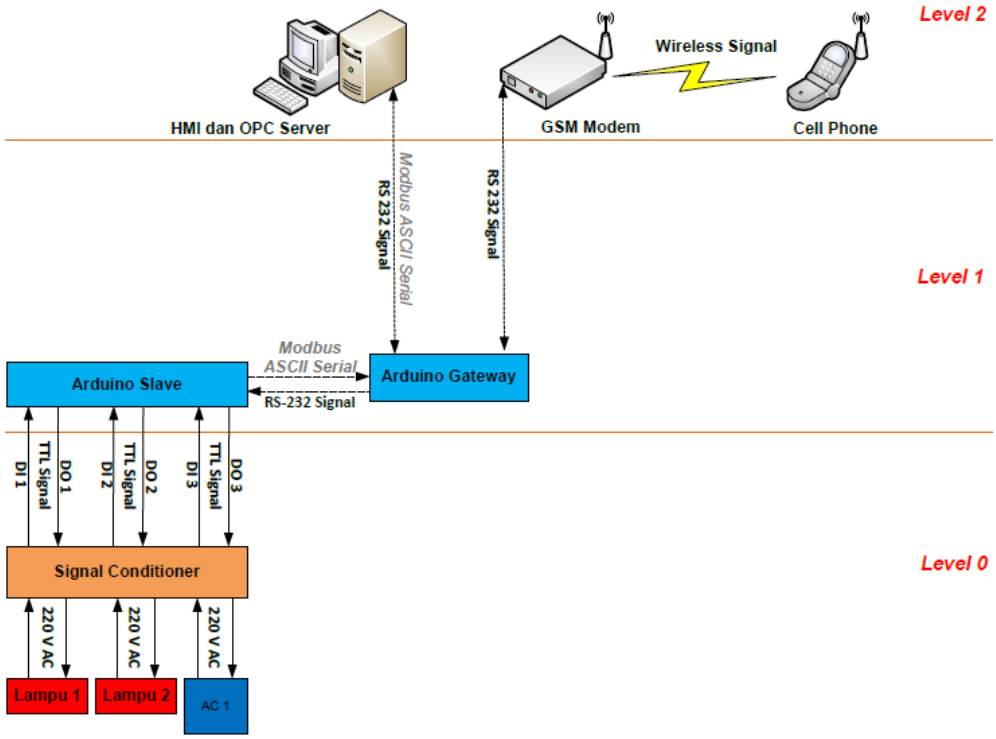
Tabel 1. Struktur Pesan Protokol Modbus

Start frame	Device adress	Function code	Data	Error check	Stop frame

Pesan pada Modbus dikirim dalam bentuk bilangan *hexadecimal*. Pemilihan bilangan *hexadecimal* dilakukan untuk mempermudah proses *troubleshooting* sebuah masalah. Ketika data yang dikirim adalah data biner 0 dan 1, maka susunan data akan menjadi sangat panjang sehingga sulit untuk membaca data yang asli. Pada tabel 1, urutan frame yang terbentuk terdiri dari *Start frame*, *device adress*, *function code*, data, *error check* dan *stop frame*. *Start frame* merupakan sebuah karakter awal yang menunjukkan adanya pesan Modbus yang diterima, pada Modbus ASCII *start frame* dan *stop frame* merupakan karakter yang tetap. *Start frame* pada struktur pesan Modbus ASCII adalah bilangan 0x3A. Sementara *stop framenya* adalah karakter 0x0D dan 0x0A. Pada frame *device adress* dan *function code*, datanya menggunakan bilangan *hexadecimal*, namun pada Modbus ASCII bilangan yang digunakan tidak berupa bilangan *hexadecimal* murni, akan tetapi bilangansebagai bilangan ASCII yang diinterpretasikan sebagai bilangan *hexadecima*l [4]. *Error check* pada Modbus ASCII dihitunhmenggunakan Metode *longitudinal redundancy check*.

3 Perancangan Sistem

3.1 Arsitektur Sistem



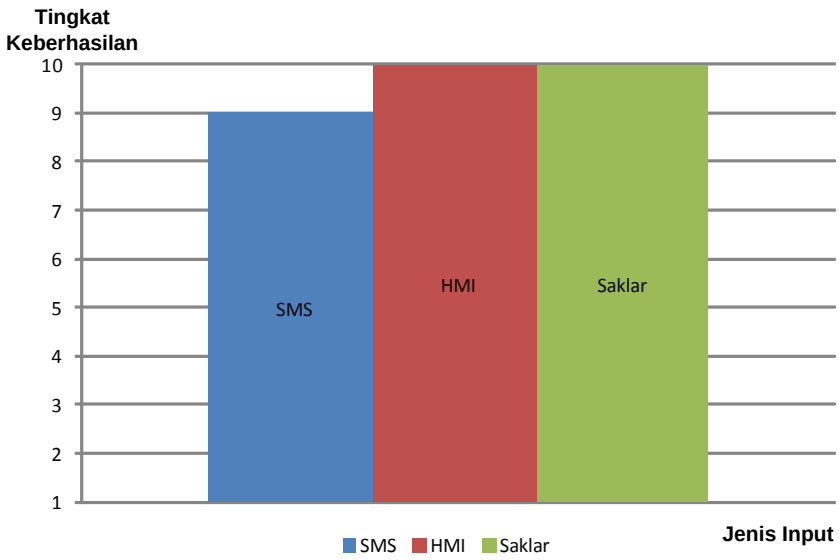
Gambar 2. Arsitektur Sistem

Pada Gambar 2 arsitektur sistem dari demoset dibatasi hanya sampai level 2 saja. Untuk level 3 keatas tidak dimasukkan karena sistem belum mewadahi untuk mencapai level tersebut. Alur kerja secara sederhana dari sistem ini bermula dari *Arduino slave* yang dapat mengakses data dari lampu dan AC, data tersebut kemudian akan diteruskan ke *Arduino gateway* yang akan terhubung dengan OPC *server* dan Modem GSM. Untuk jalur OPC *server*, *Arduino gateway* beserta *Arduino slave* akan menjadi *slave* dan OPC *server* akan menjadi *master*. Sementara itu untuk jalur modem GSM, *Arduino gateway* akan menjadi *master* dan *Arduino slave* akan menjadi *slave*. OPC *server* dapat mengakses data dari modem GSM, komunikasi yang terjadi antara OPC *server* dengan *Arduino gateway* juga menggunakan protokol Modbus ASCII serial. Data yang diakses OPC *server* akan digunakan oleh *client* untuk keperluan *monitoring* atau pengontrolan. *Client* dari OPC *server* pada penelitian ini adalah HMI. Untuk jalur modem GSM, komunikasi yang terjadi antara *Arduino gateway* dan *Arduino slave* juga menggunakan protokol Modbus ASCII serial. *client* yang akan menggunakan data tersebut adalah *celluler phone*.

4 Pengujian dan Analisis

4.1 Pengujian Demoset

Demoset yang telah dibuat akan melalui sebuah tahap pengujian. Pengujian dilakukan dengan memberikan perintah sebanyak 30 kali. Jenis perintah yang dilakukan adalah mengubah status lampu satu. Perintah yang diberikan divariasikan berasal dari 3 input yang berbeda, yakni HMI, SMS dan Saklar. Masing-masing dari ketiga input tersebut memberikan perintah sebanyak 10 kali dengan urutan yang mengacak dan jeda waktu yang tidak tetap. Rentang waktu pengujian dari pengiriman perintah sebanyak 30 kali berlangsung dari pukul 16 : 42 : 27 sampai dengan pukul 16 : 51 : 31, yakni selama 9 menit 58 detik. Berikut ini adalah grafik dari pengujian tersebut



Gambar 3 Grafik Tingkat Keberhasilan Pengujian Demoset

Dari grafik pada Gambar 3, dari 10 input yang diberikan, masing-masing dari HMI dan saklar, perangkat lampu dan AC selalu mengalami perubahan terhadap input yang diberikan. Namun untuk input dari SMS, dari 10 kali input yang diberikan terdapat 1 kali kondisi dimana demoset tidak merespon.

4.2 Waktu Pemrosesan Data

Pada bagian ini, dideteksi waktu dari pemrosesan data dari demoset yang dibuat. Waktu pemrosesan data terbagi menjadi 2 bagian yaitu:

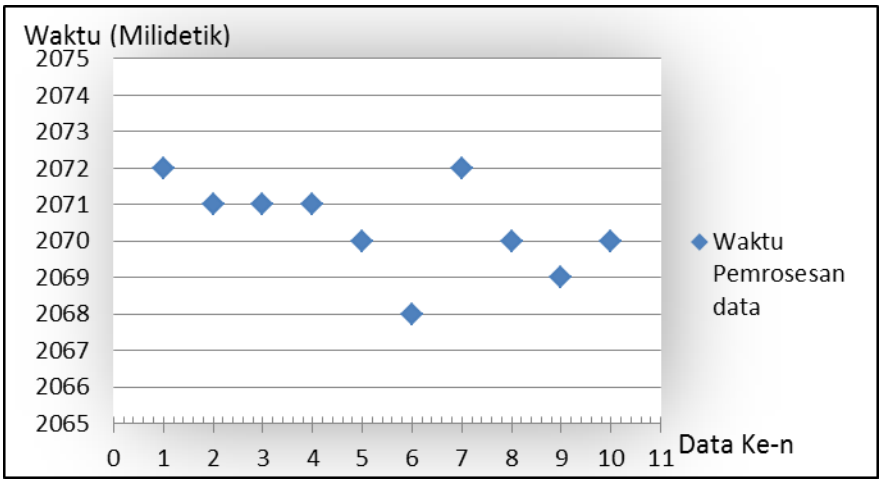
1. Pada bagian a, waktu yang dihitung adalah waktu yang dibutuhkan demoset untuk mengolah data dari mulai diterimanya sebuah SMS oleh modem GSM sampai dikirimnya kembali balasan SMS oleh modem GSM.
2. Pada bagian b, Waktu yang dibutuhkan untuk mengolah data mulai dari dikirimnya *query* message pada Arduino *gateway*, sampai diterimanya kembali *response* dari Arduino *gateway* oleh Arduino *slave*.

4.2.1 Waktu Pemrosesan data bagian a

Penghitungan waktu pemrosesan data bagian a dilakukan dengan memberikan input SMS berisi karakter T kepada GSM modem, kemudian GSM modem akan membalas SMS tersebut dengan mengirim SMS yang berisi status lampu 1, lampu 2 dan AC 1. Proses penghitungan dilakukan dengan menggunakan dua variabel. Kemudian selisih dari kedua variabel tersebut akan menghasilkan waktu kecepatan pemrosesan data pada demoset untuk input dari SMS

Waktu Pemrosesan Data = Variabel 2 – Variabel 1 (1)

Data variabel 1 dan variabel 2 diambil sebanyak 10 kali dengan rentang waktu yang tidak tetap. Ketika ada SMS berisi karakter T yang diterima oleh modem GSM maka variabel 2 akan menandai waktu tersebut, Ketika data tersebut sudah menjadi sebuah pesan yang siap dikirim, variabel 1 akan menandai waktu pada saat itu. berikut ini adalah plot grafik dari waktu pemrosesan data bagian a

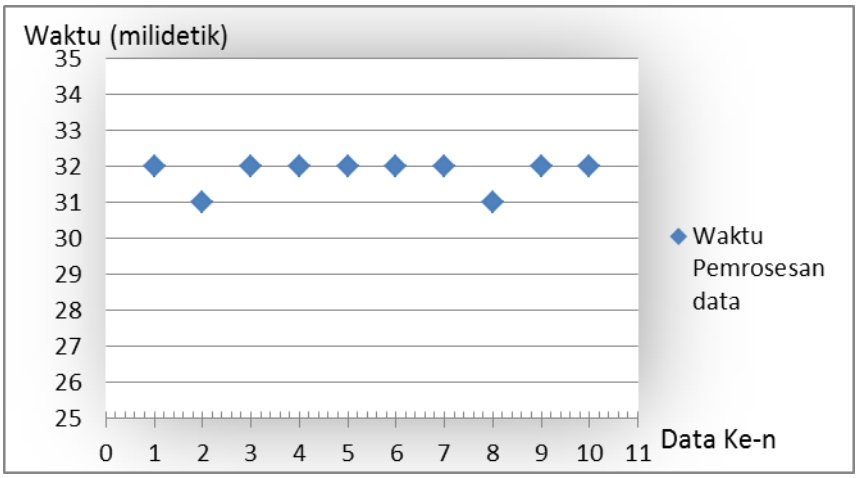


Gambar 4 Grafik Pemrosesan Data (a)

Data yang diambil sebanyak 10 kali. Dari gambar 4 dapat rata-rata waktu pemrosesan data adalah 2070 milidetik atau 2,070 detik. Grafik yang diperoleh diatas merupakan waktu yang dibutuhkan mulai dari diterimanya SMS pada GSM modem sampai dikirimnya SMS berisi status lampu dan AC.

4.2.2 Waktu Pemrosesan data bagian b

Untuk waktu pemrosesan data bagian b, metode penghitungan juga sama dengan metode penghitungan waktu pemrosesan data pada bagian a, namun yang membedakan adalah peletakan variabel 1 dan variabel 2. Variabel 2 akan menandai waktu saat akan dikirimnya data dari *Arduino gateway* ke *Arduino slave,* sedangkan variabel 1 akan menandai waktu saat diterimanya *response* oleh *Arduino gateway*. Berikut ini hasil pengambilan data yang dilakukan. Berikut ini adalah grafik dari data yang diambil



Gambar 5 Grafik Pemrosesan Data(b)

Dari grafik pada gambar 5 terlihat bahwa waktu yang dibutuhkan untuk memproses data dari mulai dikirimnya data dari *Arduino gateway* ke *Arduino slave* sampai diterimanya kembali data oleh *Arduino slave* berkisar pada angka 31 dan 32. Jika dihitung rata-ratanya maka waktu pemrosesan data pada bagian ini adalah 31, 8 milidetik. Lamanya waktu pengolahan SMS pada *Arduino gateway* dapat diperoleh dengan mengurangi rata-rata waktu pemrosesan data pada bagian a dengan waktu pemrosesan data pada bagian b, yakni 2070 milidetik- 31,8 milidetik = 2038,2 milidetik. Jika dibandingkan dengan waktu pemrosesan data bagian a, Waktu pemrosesan data pada bagian b jauh lebih kecil. Dari hasil percobaan ini, didapatkan bahwa lamanya waktu pemrosesan data pada bagian a menjadi jauh lebih lama karena pengolahan data pada SMS membutuhkan waktu yang cukup lama. Pengolahan data pada SMS menjadi cukup lama karena adanya beberapa perintah AT command yang pasti digunakan pada setiap pengolahan data SMS, yakni AT command AT+CMGR yang digunakan untuk membaca SMS yang masuk dan AT command AT +CMGS yang digunakan untuk mengirim data SMS.

5 Kesimpulan

Dari penelitian yang telah dilakukan dapat disimpulkan bahwa :

1. Demoset Pengontrolan Lampu dan AC melalui HMI dan SMS sudah berhasil dibuat namun belum cukup handal, hal ini dikarenakan dari 30 kali pengujian menggunakan input HMI, SMS dan saklar, terdapat 1 input dari SMS yang tidak direspon oleh demoset.
2. Waktu Pemrosesan data bagian a, yakni mulai dari diterimanya data pada GSM modem sampai akan dikirimnya SMS oleh GSM modem berkisar sekitar 2070 milidetik. Sementara untuk waktu pemrosesan data bagian b, yakni dari mulai dikirimnya data oleh *Arduino gateway* ke *Arduino slave* rata-ratanya adalah 31,8 milidetik.
3. Perbedaan waktu yang cukup signifikan pada waktu pemrosesan data bagian a dan bagian b terjadi karena waktu yang dibutuhkan Arduino gateway untuk mengelola SMS cukup signifikan, yakni sekitar 2038,2 milidetik.

6 Daftar Pustaka

[1] Asni. Jatiningsih. Pembuatan Modbus *Master Simulator* Untuk Pengujian Komunikasi Data *Multi-Point* PLC. Teknik Fisika Institut Teknologi Bandung, Jawa Barat. 2012.

[1] Buddy. SM5100B-D *AT Command. Shanghai Sentrue Technologies Corporation*, China. 2008.

[2] Linggar Galih, Gea. Perancangan dan Pembuatan Sistem Simulator Terintegrasi MATLAB-PLC-HMI-Database Untuk *Miniplant* Sistem Tangki GandaDengan Fasilitas *State Observer*. Teknik Fisika Institut Teknologi Bandung, Jawa Barat.2013.

[3] Krisna. Resi. Protokol Modbus. Teknik Fisika Institut Teknologi Bandung, Jawa Barat. 2012.

[4] *Ministry Of Energy and Mineral Resources*. *Indonesia Energy Statistics. Centre For Data And Information On Energy And Mineral Resources*, Jakarta. 2010.

[5] Tim Modicon *Incorporation Industrial Automation Systems*. *Modicon* Modbus *Protocol Reference Guide*. North Andover, Massachusetts. 1996.

[6] Tim *Control* Laboratory. *Basics Of Data Communication*. Teknik Fisika Institut Teknologi Bandung, Jawa Barat. 2003.

[7] Tim Developers Home, "What Makes SMS Messaging So Successful Worldwide?,"http://www.developershome.com/sms/sms_tutorial.asp?page=smsIntro2 , Mei, 2013.