SISTEM TELEKONTROL SCADA DENGAN FUNGSI DASAR MODBUS MENGGUNAKAN MIKROKONTROLER AT89S51 DAN KOMUNIKASI SERIAL RS485

Agus Tiyono[(1)], Sudjadi[(2)], Iwan Setiawan[(2)]
Laboratorium Teknik Kontrol Otomatik
Jurusan Teknik Elektro Fakultas Teknik Universitas Diponegoro

ABSTRAK

SCADA memiliki fungsi sebagai telemetery dan telecontrol. Dengan fungsi-fungsi tersebut, sistem SCADA memiliki kelebihan dapat melakukan pengawasan sekaligus pengendalian banyak plant yang letaknya berjauhan. Sistem SCADA terdiri dari 3 bagian utama yaitu Master (MTU, Master Terminal Unit), Slave (RTU, Remote Terminal Unit), dan media komunikasi. Master memiliki fungsi sebagai pengendali komunikasi, sedangkan Slave berfungsi menjalankan perintah dari Master. Komunikasi antara Master dan Slave menggunakan protokol Modbus. Media komunikasi pada sistem SCADA dapat menggunakan ethernet, wireless, atau serial.

Pada penelitian ini dibuat sebuah model sistem SCADA dengan menerapkan salah satu fungsinya, yaitu sebagai pengendali jarak jauh (telecontrolling). Dalam sistem telekontrol SCADA ini digunakan mikrokontroler AT89S51 sebagai pembentuk komponen Master dan Slave. Komunikasi antara Master dan Slave menggunakan fungsi dasar protokol Modbus dan komunikasi serial RS485.

Hasil pengujian menunjukkan bahwa sistem telekontrol SCADA Modbus yang dibuat mendukung fungsi protokol Modbus 05, 06, 15 dan 16 sebagai fungsi telecontrolling, serta fungsi 01, 02, 03 dan 04 sebagai monitoring. Fungsi telekontrol 05, 06, 15, dan 16 dapat berjalan dengan baik untuk mode pengalamatan broadcast dan unicast. Pengiriman query dengan Slave ID yang tidak didukung sistem SCADA Modbus akan menghasilkan time-out error, sedangkan untuk kode fungsi yang tidak didukung sistem akan menghasilkan exception response. Komunikasi antara Master dan Slave dapat berjalan dengan baik menggunakan panjang kabel 6 m dan 100 m, serta dengan delay waktu pengiriman antar karakter pesan tidak lebih besar dari 1656,90 µs.

Kata Kunci : *SCADA, Telekontrol, Modbus, RS485*

I. PENDAHULUAN

1.1 Latar Belakang Masalah

Dalam suatu industri yang memiliki banyak peralatan (*plant*) dengan letak geografis yang tersebar pada jarak yang cukup jauh, diperlukan sebuah sistem yang dapat melakukan pengawasan dan pengendalian jarak jauh. Sistem tersebut harus mampu melakukan pengambilan data (*data acquisition*) dengan baik dan menganalisa data tersebut, sehingga dapat melakukan pengawasan (*supervisory*) dan pengendalian (*controlling*) terhadap semua proses yang sedang berjalan.

Untuk mengatasi permasalahan tersebut, banyak industri yang menerapkan SCADA (*Supervisory Control and Data Acquisition*) sebagai sistem pengawasan dan pengendalian. Sistem SCADA terdiri dari 3 bagian utama yaitu Master, Slave, dan media komunikasi (*link*). Master berfungsi sebagai pengendali komunikasi dengan mengirimkan perintah (*query*), sedangkan Slave berfungsi memberikan respon atas perintah Master tentang kondisi *plant*. Master dan Slave biasanya berupa PLC atau mikroprosesor. Komunikasi antara Master dan Slave menggunakan aturan komunikasi protokol Modbus. Media komunikasi yang digunakan pada sistem SCADA dapat berupa ethernet, *wireless*, atau serial.

Pada penelitian ini dibuat sebuah sistem SCADA dengan menerapkan salah satu fungsinya, yaitu sebagai pengendali jarak jauh atau telekontrol (*telecontrolling*). Dalam sistem telekontrol SCADA ini digunakan mikrokontroler AT89S51 sebagai pembentuk komponen Master dan Slave. Komunikasi antara Master dan Slave menggunakan komunikasi serial RS485, dan fungsi dasar protokol Modbus.

1.2 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk membuat model sistem SCADA sebagai sistem telekontrol yang dibentuk dari fungsi-fungsi kontrol dasar protokol Modbus menggunakan komunikasi serial RS485.

1.3 Pembatasan Masalah

Dalam pembuatan tugas akhir ini penulis membatasi permasalahan sebagai berikut:

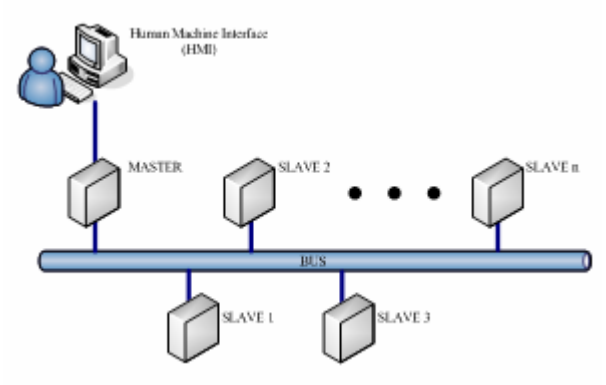
1. Sistem SCADA dibuat sebagai telekontrol, yaitu disusun dengan fungsi-fungsi kontrol dasar protokol Modbus.
2. Komponen Master dan Slave dibentuk dengan menggunakan mikrokontroler AT89S51.
3. Komunikasi antara Master dan Slave menggunakan aturan protokol Modbus dasar.
4. Komunikasi antara Master dan Slave menggunakan komunikasi serial RS485 secara *half-duplex*.
5. Antarmuka komunikasi serial RS485 menggunakan IC SN75176.

1) Mahasiswa Jurusan Teknik Elektro UNDIP
2) Staf Pengajar Jurusan Teknik Elektro UNDIP

II. DASAR TEORI

2.1 Sistem SCADA [6]

SCADA (*Supervisory Control And Data Acquisition*) adalah sebuah sistem pengawasan dan pengendalian, dengan cara melakukan pengumpulan dan analisa data secara *real time*. Sistem SCADA terdiri dari 3 bagian utama yaitu Master, Slave, dan media komunikasi. Arsitektur SCADA diperlihatkan pada Gambar 2.1.



Gambar 2.1 Arsitektur SCADA.

Sistem SCADA memiliki fungsi yaitu untuk pengukuran jarak jauh (*telemetering*) dan pengendalian jarak jauh (*telecontrolling*). Dengan fungsi tersebut, sistem SCADA mampu melakukan pengawasan dan pengendalian *plant* jarak jauh.

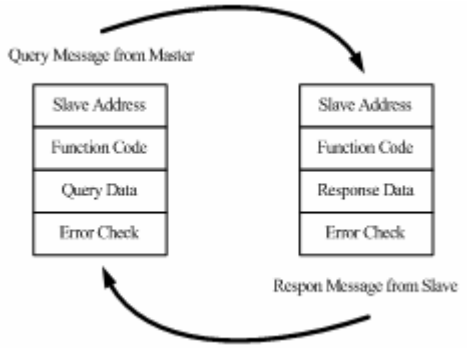
2.2 Protokol Modbus [12][13][14]

Protokol Modbus merupakan aturan-aturan komunikasi data dengan teknik Master-Slave. Dalam konunikasi tersebut hanya terdapat satu Master dan satu atau beberapa Slave yang membentuk sebuah jaringan. Komunikasi Modbus selalu diawali dengan *query* dari Master, dan Slave memberikan respon dengan mengirimkan data atau melakukan aksi sesuai perintah dari Master. Master hanya melakukan satu komunikasi dalam satu waktu. Slave hanya akan melakukan komunikasi jika ada perintah (*query*) dari Master dan tidak bisa melakukan komunikasi dengan Slave yang lain.

Pada saat mengirimkan *query* ke Slave, Master menggunakan 2 mode pengalamatan, yaitu:

- *Unicast mode*.
  Master mengirimkan *query* kepada satu Slave. Setelah menerima dan memproses *query*, Slave akan memberikan jawaban berupa respon kepada Master.
- *Broadcast mode*.
  Master mengirimkan perintah (*query*) kepada semua Slave. Pada mode pengalamatan ini Slave tidak mengirimkan respon kepada Master.

Protokol Modbus membentuk sebuah format pesan untuk *query* Master dan respon Slave. Adapun siklus pengiriman *query*-respon ditunjukkan pada Gambar 2.2.



Gambar 2.2 Siklus pengiriman *query*-respon.

2.2.1 Mode Transmisi Serial

Dalam jaringan Modbus terdapat 2 mode transmisi serial, yaitu mode RTU dan mode ASCII. Setiap peralatan dalam sebuah jaringan Modbus harus mempunyai mode dan parameter serial yang sama. Pengaturan *default* Modbus adalah RTU, sedangkan mode ASCII adalah pilihan.

2.2.1.1 Mode RTU (*Remote Terminal Unit*)

Format masing-masing *byte* (11 bit) dalam mode RTU adalah:

Coding system: 8 bit biner, heksadesimal 0-9,A-F.
Bits per byte: 1 start bit.
8 data bits, *Least Significant Bit* (LSB) dikirim pertama.
1 bit untuk even/odd parity, no bit untuk no parity.
1 stop bit jika menggunakan parity, 2 bits untuk no parity.
Error check field: *Cyclical Redundancy Check* (CRC).

2.2.1.2 Mode ASCII (*American Standard Code for Information Interchange*)

Format masing-masing *byte* (10 bit) dalam mode ASCII adalah:

Coding system: Heksadesimal, karakter ASCII 0-9, A-F.
Bits per byte: 1 start bit.
7 data bits, *Least Significant Bit* (LSB) dikirim pertama.
1 bit untuk even/odd parity, no bit untuk no parity.
1 stop bit jika menggunakan parity, 2 bits untuk no parity.
Error check field: *Longitudinal Redundancy Check* (LRC).

2.2.2 Modbus Message Framing

2.2.2.1 ASCII *Framing*

Frame pesan pada mode transmisi ASCII ditunjukkan pada Tabel 2.1.

Tabel 2.1 ASCII *Framing*.

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHAR	2 CHAR	n CHAR	2 CHAR	2 CHAR CRLF

Pada mode ASCII, pesan dimulai dengan sebuah karakter "*colon*" (:) dalam ASCII 3A hex, dan diakhiri dengan sebuah pasangan "*carriage return – line feed*" (CRLF) dalam ASCII 0D dan 0A hex.

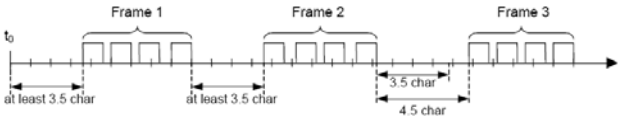
2.2.2.2 RTU *Framing*

Frame pesan pada mode transmisi RTU ditunjukkan pada Tabel 2.2.

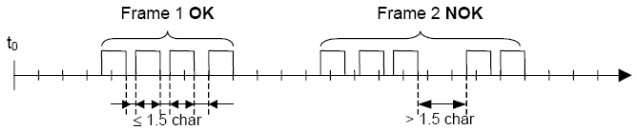
Tabel 2.2 RTU *Framing*.

START	ADDRESS	FUNCTION	DATA	CRC CHECK	END
$T_{3,5}$ ≥ 3,5 CHAR	8 BIT	8 BIT	nx8 BIT	16 BIT	$T_{3,5}$ ≥ 3,5 CHAR

Pada mode RTU, *frame* pesan dipisahkan oleh *silent interval* paling sedikit waktu 3,5 karakter. Interval waktu ini disebut $T_{3,5}$. Seluruh karakter dalam *frame* pesan harus ditransmisikan secara bersambung. Interval antar karakter dalam *frame* pesan tidak boleh lebih besar dari waktu 1,5 karakter ($T_{1,5}$). Jika interval antar karakter lebih besar dari $T_{1,5}$, maka *frame* pesan tersebut dinyatakan tidak lengkap dan akan diabaikan.



Gambar 2.3 Interval $T_{3,5}$ antar *frame*.



Gambar 2.4 Interval $T_{1,5}$ antar karakter dalam *frame*.

2.2.3 *Address Field*

Masing-masing Slave harus mempunyai alamat yang berbeda dalam range 1 – 247 untuk pengalamatan individual. Alamat 0 digunakan untuk pengalamatan *broadcast*.

2.2.4 *Function Field*

Function field pada *frame* pesan berisi nomer kode fungsi (*function code*). Kode fungsi yang valid mempunyai range 1 – 255, dimana kode 1 – 127 untuk fungsi normal, sedangkan 128 – 255 untuk fungsi *exception response*.

Function code berfungsi untuk memberitahu Slave tentang perintah yang harus dikerjakan dan sebagai indikasi respon normal atau jenis *error* yang terjadi (*exception response*).

Pada sistem komunikasi Modbus, jumlah *function code* yang didukung bervariasi tergantung kontroler dan peralatan Slave yang digunakan. Beberapa kode fungsi berikut keterangannya ditunjukkan pada Tabel 2.3.

Tabel 2.3 Kode fungsi Modbus.

Kode Fungsi	Fungsi	Aksi yang dikerjakan
1 = 01H	*Read Coil Status*	Membaca status ON/OFF coil (output digital).
2 = 02H	*Read Input Status*	Membaca status ON/OFF input digital.
3 = 03H	*Read Holding Registers*	Membaca nilai output analog.
4 = 04H	*Read Input Registers*	Membaca nilai input analog.
5 = 05H	*Force Single Coil*	Mengset status satu coil pada keadaan ON/OFF.
6 = 06H	*Preset Single Register*	Mengset nilai satu output analog.
15 = 0FH	*Force Multiple Coils*	Mengset status beberapa coil pada keadaan ON/OFF.
16 = 10H	*Preset Multiple Registers*	Mengset nilai beberapa output analog.

2.2.5 *Data Field*

Data field pada *query* berisi kode sebagai informasi tambahan pada *function code* tentang aksi yang harus dikerjakan Slave. Informasi tersebut bisa berupa alamat input-output, jumlah input-output, jumlah *byte* data, atau nilai data pengesetan. Jika tidak terjadi *error*, *data field* pada respon berisi data yang diminta. Sedangkan pada *exception response*, *data field* berisi *exception code*.

2.2.6 *Error Checking Field*

2.2.6.1 LRC (*Longitudinal Redundancy Check*)

Pada mode ASCII, *error checking field* berisi 2 karakter ASCII yang didasarkan metode LRC. Prosedur perhitungan nilai LRC adalah :

1. Tambahkan semua *byte* pesan tanpa mengikutkan karakter *start* yaitu colon dan karakter *end* yaitu CRLF, dan tanpa melibatkan *carry*.
2. Kurangkan nilai FF hex dengan nilai hasil penjumlahan semua *byte* pesan, untuk menghasilkan komplemen 1.
3. Tambahkan hasilnya dengan 1 untuk menghasilkan komplemen dua. Hasilnya merupakan nilai LRC.

2.2.6.2 CRC (*Cyclical Redundancy Check*)

Pada mode RTU, *error checking field* berisi sebuah nilai 16 bit (2 *byte*) yang didasarkan pada metode CRC. Prosedur perhitungan CRC adalah :

1. Inisialisasi nilai register 16 bit CRC dengan FFFF hex.
2. Eksklusif OR 8 bit data pesan pertama dengan *low order byte* register CRC, letakkan hasilnya di register CRC.
3. Geser kanan register CRC 1 bit ke arah LSB, dan MSB diisi dengan 0. Nilai LSB register CRC yang tergeser diperiksa.

4. Jika LSB tergeser adalah 0, ulangi langkah 3 (pergeseran yang lain). Jika LSB tergeser 1, eksklusif-OR register CRC dengan nilai A001 hex (1010 0000 0000 0001).
5. Ulangi langkah 3 dan 4 sampai delapan pergeseran. Setelah delapan pergeseran, proses 8 bit data pesan pertama selesai.
6. Ulangi langkah 2 - 5 untuk 8 bit data pesan berikutnya sampai semua data diproses.
7. Nilai akhir register CRC adalah nilai CRC.
8. Pada saat CRC ditempatkan di pesan, nilai CRC *low order byte* dikirimkan terlebih dahulu dikuti *high order byte*.

2.2.7 *Exception Response*

Terdapat 4 proses komunikasi yang mungkin terjadi antara Master dan Slave, yaitu:

- Jika Slave menerima pesan *query* tanpa adanya kesalahan komunikasi, dan Slave dapat menangani *query* tersebut, Slave akan memberikan sebuah respon normal.
- Jika Slave tidak menerima *query* dikarenakan adanya kesalahan komunikasi, maka tidak ada respon yang dikirimkan. Master akan memberikan kondisi *time-out* untuk pengiriman *query* tersebut.
- Jika Slave menerima pesan *query*, tetapi terdeteksi kesalahan komunikasi (*parity*, LRC, atau CRC), maka tidak ada respon yang dikirimkan. Master akan memberikan kondisi *time-out*.
- Jika Slave menerima *query* tanpa adanya kesalahan komunikasi, tetapi Slave tidak dapat menangani perintah tersebut (contoh, perintah untuk membaca *coil* atau register yang tidak ada), Slave akan mengirimkan sebuah respon pengecualian (*exception response*) untuk memberikan informasi kepada Master letak kesalahan yang terjadi.

Pada sebuah *exception response*, Slave mengembalikan kode fungsi dengan MSB (*Most Significant Bit*) diset 1 dan *data field* diisi dengan kode pengecualian (*exception code*). Hal ini dimaksudkan agar Master mengetahui *exception* yang terjadi. Beberapa *exception code* berikut keterangannya ditunjukkan pada Tabel 2.4.

Tabel 2.4 *Exception code* dalam *exception response*.

Kode	Nama	Arti
01	*ILLEGAL FUNCTION*	Kode fungsi yang terdapat dalam *query* merupakan perintah (*action*) yang tidak diizinkan untuk Slave.
02	*ILLEGAL DATA ADDRESS*	Alamat data dalam *query* merupakan alamat yang tidak diizinkan untuk Slave.
03	*ILLEGAL DATA VALUE*	Nilai dalam *data field query* merupakan nilai yang tidak diizinkan untuk Slave.

2.3 Komunikasi Serial

Komunikasi serial ada 2 macam, yaitu komunikasi sinkron dan asinkron. Komunikasi sinkron dilakukan dengan menambahkan sinyal sinkronisasi. Komunikasi asinkron dilakukan dengan menetapkan kecepatan bit (*baud rate*) dan menyisipkan beberapa bit protokol, yaitu bit start, *parity* dan bit stop.

2.3.1 Port Serial AT89S51

Mikrokontroler AT89S51 memiliki fasilitas *port* serial yaitu P3.0 untuk RxD dan P3.1 untuk TxD. *Port* serial AT89S51 ini bersifat *full duplex*, artinya dapat mengirim dan menerima data secara simultan. Selain itu memiliki *buffer* penerima, sehingga *port* serial dapat bersiap menerima data kedua sebelum data pertama dibaca dari register penerima. Register penerima dan pengirim *port* serial diakses melalui SBUF (*Serial Buffer*).

Port serial AT89S51 dapat bekerja dalam 4 mode, dengan pemilihan seperti dalam Tabel 2.5.

Tabel 2.5 Pemilihan mode port serial.

SM0	SM1	Mode	Keterangan	*Baud Rate*
0	0	0	Shift Register	1/12 Frekuensi *Oscillator*
0	1	1	8 bit UART	Diset oleh Timer
1	0	2	9 bit UART	1/64 atau 1/32 Frek. *Oscillator*
1	1	3	9 bit UART	Diset oleh Timer

Baud rate untuk Mode 1 dan Mode 3 ditentukan oleh *overflow rate* Timer1 dan nilai SMOD dengan persamaan sebagai berikut.

$$Baud\ Rate\ Mode\ 1,3 = \frac{2^{SMOD}}{32} \times Timer\ 1\ Overflow\ Rate$$

$$Baud\ Rate\ Mode\ 1,3 = \frac{2^{SMOD}}{32} \times \frac{Frekuensi\ Osilator}{12 \times [256 - (TH1)]}$$

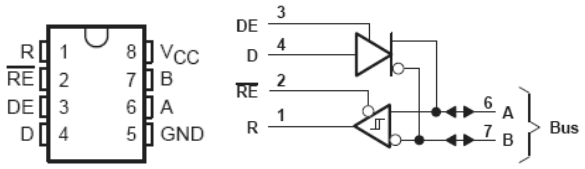
2.3.2 Komunikasi Serial RS232

Standar RS232 merupakan aturan mengenai level tegangan, konektor dan aturan komunikasi. Standar RS232 memiliki level tegangan antara -3 sampai -15 Volt untuk logika *high*, dan antara +3 sampai +15 Volt untuk logika *low*. Level tegangan antara -3 sampai +3 Volt tidak didefinisikan, sebab di daerah ini kemungkinan adalah *noise*.

2.3.3 Komunikasi Serial RS485

Komunikasi serial RS485 menggunakan sepasang kabel untuk mengirimkan satu sinyal. Tegangan antara kedua kabel saluran selalu berlawanan. Logika ditentukan dari beda tegangan antara kedua kabel tersebut.

SN75176 merupakan IC multipoint RS485 *transceiver*. Di dalam SN75176 terdapat sebuah *driver* dan *receiver* seperti pada Gambar 2.5.



Gambar 2.5 Bagan IC SN75176.

SN75176 dapat mendukung 32 unit paralel dalam satu jalur. Sensitivitas tegangan input *receiver* 0,2 V dan jarak maksimum 4000 feet.

Pada mode pengiriman (*transmitting*), kaki *enable* kirim DE diberi logika 1. Keluaran A dan B ditentukan oleh masukan *driver* D, dimana keluaran A akan sesuai dengan logika *driver* D, sedangkan B berkebalikan. Jika input D berlogika 1, maka output A akan bertegangan 5 Volt dan output B 0 Volt. Sebaliknya jika input D berlogika 0 maka output A bertegangan 0 Volt dan output B 5 Volt.

Pada mode penerimaan (*receiving*), kaki *enable* terima RE diberi logika 0. Output *receiver* R ditentukan oleh tegangan diferensial antara input A dan B (V_A - V_B). Jika tegangan diferensial V_A - V_B lebih besar dari +0,2 Volt, maka *receiver* R akan berlogika 1, sedangkan jika V_A - V_B lebih kecil dari -0,2 Volt maka *receiver* R akan berlogika 0. Untuk tegangan V_A - V_B antara -0,2 Volt sampai +0,2 Volt, maka level logika keluaran tidak terdefinisi.

Mode pengiriman dan penerimaan data SN75176 ditunjukkan pada Tabel 2.6 dan 2.7.

Tabel 2.6 Pengiriman data (*transmitting*).

INPUT D	ENABLE DE	OUTPUT	
		A	B
H	H	H	L
L	H	L	H
x	L	Z	Z

Tabel 2.7 Penerimaan data (*receiving*).

DIFFERENTIAL INPUTS (A – B)	ENABLE $\overline{RE}$	OUTPUT R
$V_{ID} \geq 0{,}2V$	L	H
$-0{,}2V < V_{ID} < 0{,}2V$	L	?
$V_{ID} \leq -0{,}2V$	L	L
x	H	Z
OPEN	L	?

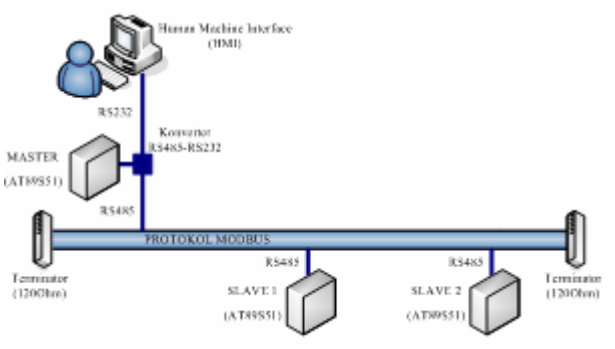
H = *High Level*, L = *Low Level*, x = *Irrelevant*, Z = *high impedance (off)*, ? = *Indeterminate*

Jika terdapat gangguan listrik yang menimpa saluran transmisi, maka induksi tegangan gangguan akan diterima kedua kabel saluran sama besar. Karena *Receiver* membandingkan selisih tegangan antara dua kabel saluran, maka induksi tegangan tidak akan berpengaruh pada output. Dengan kemampuan menangkal gangguan yang sangat baik ini, RS485 bisa dipakai untuk membangun saluran transmisi jarak jauh sampai 4000 feet dengan kecepatan tinggi.

III. PERANCANGAN SISTEM

3.1 Konfigurasi Sistem Telekontrol SCADA

Sistem telekontrol SCADA Modbus yang dibangun tersusun atas 1 Master dan 2 Slave yang membentuk sebuah jaringan bus, serta HMI (*Human Machine Interface*) sebagai antarmuka dalam proses *monitoring* dan pengendalian. Dalam jaringan bus ini digunakan kabel telepon sebagai media komunikasi, dimana di kedua ujungnya terdapat terminator berupa resistor 120 Ohm. Konfigurasi sistem telekontrol SCADA Modbus diperlihatkan pada Gambar 3.1.



Gambar 3.1 Konfigurasi sistem Scada Modbus.

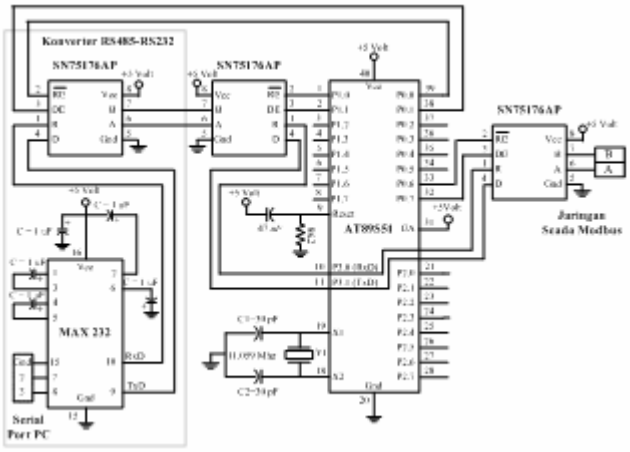
Sistem telekontrol SCADA Modbus dirancang dalam 3 bagian yaitu perancangan perangkat keras, perancangan protokol Modbus dan perancangan perangkat lunak.

3.2 Perancangan Perangkat Keras

Perangkat keras yang membangun sistem telekontrol SCADA Modbus terdiri atas komputer (PC, *Personal Computer*) sebagai HMI dan sistem minimum mikrokontroler AT89S51 yang dilengkapi komunikasi serial RS485 *transceiver*, SN75176 yang membentuk Master dan Slave.

3.2.1 Rangkaian Master Scada

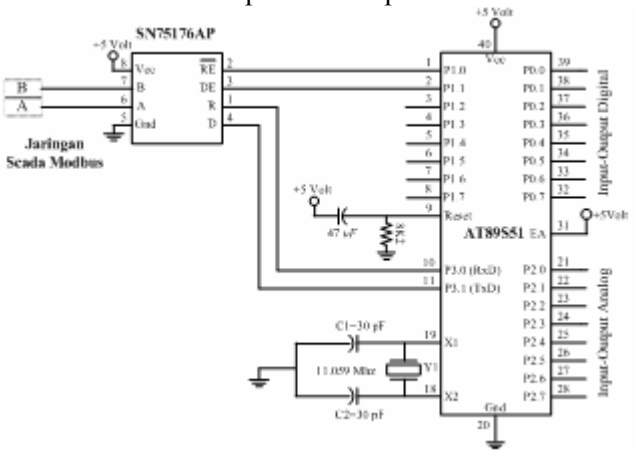
Konfigurasi rangkaian Master sistem telekontrol SCADA diperlihatkan pada Gambar 3.2.



Gambar 3.2 Rangkaian Master Scada.

3.2.2 Rangkaian Slave Scada

Rangkaian Slave sistem telekontrol SCADA Modbus diperlihatkan pada Gambar 3.3.



Gambar 3.3 Rangkaian Slave Scada.

Slave pada sistem telekontrol SCADA Modbus memiliki 4 jenis perangkat input-output sebagai sarana untuk berhubungan dengan peralatan luar dengan pembagian sebagai berikut:

- *Coil* (Output digital) yaitu *Port* 0 yang terdiri dari 8 bit, sehingga terdapat 8 *coil*.
- *Input status* (Input digital) yaitu 8 pin *Port* 0, sehingga terdapat 8 *Input status*.
- *Holding register* (Output analog) yaitu *Port* 2 sebagai output analog 8 bit, sehingga mempunyai range nilai 00 sampai FF (hex).
- *Input register* (Input analog) yaitu *Port* 2 sebagai input analog 8 bit, sehingga mempunyai range nilai 00 sampai FF (hex).

3.3 Perancangan Protokol Modbus

Protokol Modbus merupakan aturan komunikasi dalam transmisi data antara Master dan Slave. Fungsi dasar Modbus yang akan dibuat dalam sistem telekontrol SCADA Modbus terdiri dari fungsi 05, 06, 15, dan 16 sebagai fungsi pengesetan (*telecontrolling*) dan ditambahkan fungsi 01, 02, 03, dan 04 sebagai monitoring proses pengendalian.

Dalam perancangan perangkat keras, Slave sistem telekontrol SCADA Modbus memiliki 8 input-output digital yaitu *port* 0 dan 1 input-output analog 8 bit yaitu *port* 2. Alamat dari input-output tersebut adalah:

1. Input-Output Digital, yaitu *Port* 0.
   - → P0.0, *Address* 0000.
   - → P0.1, *Address* 0001.
   - → P0.2, *Address* 0002.
   - → P0.3, *Address* 0003.
   - → P0.4, *Address* 0004.
   - → P0.5, *Address* 0005.
   - → P0.6, *Address* 0006.
   - → P0.7, *Address* 0007.
2. Input-Output Analog, yaitu *Port* 2 dengan *Address* 0000.

Dalam sistem telekontrol SCADA Modbus ini digunakan protokol Modbus mode RTU dan cek *error* CRC (*Cyclical Redundency Check*). Sistem ini tersusun dari 2 Slave sehingga *Address Slave* memiliki *range* 01 – 02 untuk pengalamatan *unicast* dan 0 untuk *broadcast*.

1. Fungsi 01 dan 02

Pesan *query* terdiri dari 8 *byte* seperti ditunjukkan pada Tabel 3.1.

Tabel 3.1 Pesan *query* fungsi 01/02 sistem telekontrol SCADA Modbus.

Address	Function	Starting Address		Number Register		Error Check	
		Hi	Lo	Hi	Lo	Lo	Hi
01	01/02	00	00	00	01	-	-

Fungsi ini tidak mendukung mode *broadcast*. Setelah Slave melakukan proses pembacaan input-output digital, selanjutnya hasil pembacaan tersebut dikirimkan ke Master dalam bentuk pesan respon seperti pada Tabel 3.2.

Tabel 3.2 Pesan respon fungsi 01/02 sistem telekontrol SCADA Modbus.

Address	Function	Byte Count	Data	Error Check	
				Lo	Hi
01	01	01	00	-	-

2. Fungsi 03 dan 04

Fungsi ini merupakan fungsi untuk membaca I/O analog, dimana dalam sistem telekontrol SCADA Modbus ini hanya terdapat 1 I/O analog 8 bit, yaitu *port* 2 AT89S51.

Tabel 3.3 Pesan *query* fungsi 03/04 sistem telekontrol SCADA Modbus.

Address	Function	Starting Address		Number Register		Error Check	
		Hi	Lo	Hi	Lo	Lo	Hi
01	03	00	00	00	01	-	-

Tabel 3.4 Pesan respon fungsi 03/04 sistem telekontrol SCADA Modbus.

Address	Function	Byte Count	Data Register		Error Check	
			Hi	Lo	Lo	Hi
01	03	01	00	00	-	-

3. Fungsi 05 (*Force Single Coil*)

Fungsi 05 Modbus merupakan fungsi untuk mengatur kondisi sebuah *coil* (output digital) pada keadaan ON atau OFF. Respon dan *query* memiliki format yang sama, dimana Slave mengirimkan respon setelah Slave mengubah keadaan *coil* sesuai data ON/OFF yang diberikan. Fungsi ini mendukung mode *broadcast*. Data ON/OFF diisi dengan FF00 untuk ON, dan 0000 untuk OFF.

Tabel 3.5 Pesan *query* dan respon fungsi 05 sistem telekontrol SCADA Modbus.

Address	Function	Data Coil		Data ON/OFF		Error Check	
		Hi	Lo	Hi	Lo	Lo	Hi
01	05	00	00	00	01	-	-

4. Fungsi 06 (*Preset Single Register*)

Fungsi 06 Modbus berfungsi mengatur atau memberikan sebuah nilai pada output analog 8 bit. Respon dan *query* memiliki format yang sama, dimana respon dikirim setelah Slave mengubah nilai output analog. Fungsi ini mendukung mode *broadcast*. *Data value* mempunyai *range* antara 0000 sampai 00FF.

Tabel 3.6 Pesan *query* dan respon fungsi 06 sistem telekontrol SCADA Modbus.

Address	Function	Data Register		Data Value		Error Check	
		Hi	Lo	Hi	Lo	Lo	Hi
01	06	00	00	00	01	-	-

5. Fungsi 15 (*Force Multiple Coils*)

Fungsi 15 Modbus hampir sama dengan fungsi 05, hanya saja fungsi ini digunakan untuk mengatur *coil* lebih dari satu. Fungsi ini mendukung mode *broadcast*.

Tabel 3.7 Pesan *query* fungsi 15 sistem telekontrol SCADA Modbus.

Addr.	Func.	Starting Address		Quantity		Byte Count	Data Coil	Error Check	
		Hi	Lo	Hi	Lo			L	H
01	0F	00	00	00	01			-	-

Tabel 3.8 Pesan respon fungsi 15 sistem telekontrol SCADA Modbus.

Address	Function	Starting Address		Quantity		Error Check	
		Hi	Lo	Hi	Lo	Lo	Hi
01	0F	00	00	00	01	-	-

6. Fungsi 16 (*Preset Multiple Register*)

Fungsi 16 Modbus ini hampir sama seperti fungsi 06, hanya saja fungsi ini dapat mengatur atau mengeset nilai output analog lebih dari satu. Fungsi ini mendukung mode *broadcast*. *Data* selalu 1 *byte* dengan *range* antara 0000 sampai 00FF.

Tabel 3.9 Pesan *query* fungsi 16 sistem telekontrol SCADA Modbus.

Addr.	Func.	Starting Address		Quantity		Byte Count	Data		Error Check	
		H	L	H	L		H	L	L	H
01	10	00	00	00	01				-	-

Tabel 3.10 Pesan respon fungsi 16 sistem telekontrol SCADA Modbus.

Address	Function	Starting Address		Quantity		Error Check	
		Hi	Lo	Hi	Lo	Lo	Hi
01	10	00	00	00	01	-	-

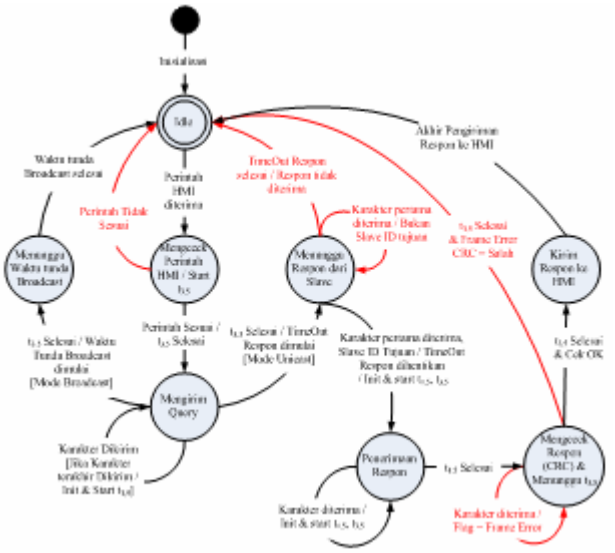
3.4 Perancangan Perangkat Lunak

Program dalam sistem telekontrol SCADA Modbus dibagi menjadi 2 yaitu program Master dan Slave pada mikrokontroler AT89S51 dan program HMI (*Human Machine Interface*) pada komputer. Program pada mikrokontroler dibuat menggunakan *assembler*, sedangkan program di dalam komputer menggunakan Visual C++.

3.4.1 Program pada Mikrokontroler

3.4.1.1 Program Master

Perancangan pembuatan program Master dilakukan dengan menggunakan diagram keadaan (*state diagram*) seperti terlihat pada Gambar 3.4.



Gambar 3.4 Diagram keadaan program Master.

Program Master diawali dengan inisialisasi mode serial, mode Timer, dan *baud rate*. Mode serial yang digunakan yaitu komunikasi multiprosesor Mode 3 (9 bit UART) dengan *baudrate* 9600 bps.

"*Idle*" *state* merupakan keadaan Master menunggu perintah dari HMI. Pada state ini, RS485 Master ke HMI diset pada mode terima dan RS485 konverter RS485-RS232 diset pada mode kirim, sedangkan RS485 Master ke Slave diambangkan. Pengaturan ini dimaksudkan agar data dari HMI hanya diterima oleh Master tanpa mengganggu komunikasi jaringan Scada.

Setelah perintah dari HMI diterima, Master beralih menuju *state* "mengecek perintah HMI dan start $T_{3,5}$". Perintah HMI yang dapat ditangani Master memiliki 4 format yaitu

- Header 255 diikuti dengan alamat Slave dan fungsi untuk perintah polling (fungsi 01, 02, 03 dan 04).
- Header 254 diikuti alamat Slave, kode fungsi dan beberapa data untuk perintah kontrol (fungsi 05, 06, 15 dan 16).

- Header 253 diikuti dengan alamat Slave, fungsi dan beberapa data untuk perintah manual, dan
- Header 252 diikuti dengan nilai TH0 dan TL0 untuk perintah set *delay* karakter.

Pada *state* "mengirim *query*" setelah karakter terakhir dikirim, Master akan menginisialisasi dan mulai menghitung $T_{3,5}$. Untuk mode *broadcast*, setelah $T_{3,5}$ selesai Master akan mulai menghitung waktu tunda *broadcast* sampai waktu tersebut selesai. Pada mode *unicast*, setelah $T_{3,5}$ selesai Master akan mulai menghitung *time-out response*. Besarnya waktu tunda *broadcast* adalah 100 ms (2 x 50 ms), sedangkan *time-out response* adalah 1 s (20 x 50 ms).

State "menunggu respon dari Slave" merupakan keadaan menunggu pesan jawaban dari Slave. Jika Master tidak menerima respon sampai waktu *time-out response* selesai, maka akan terjadi *time-out error*. Apabila Master menerima respon, maka Master akan memeriksa karakter pertama dari respon tersebut. Jika karakter tersebut bukan Slave ID tujuan, maka Master akan menunggu kembali sampai didapatkan Slave ID tujuan atau terjadi *time-out error*.

Setelah Master mendapatkan Slave ID tujuan, Master akan menghentikan *time-out response* serta menginisialisasi dan mulai menghitung $T_{1,5}$ dan $T_{3,5}$, kemudian menuju ke *state* "penerimaan respon". Setiap terjadi penerimaan karakter, Master akan menginisialisasi dan menghitung ulang $T_{1,5}$ dan $T_{3,5}$. Setelah $T_{1,5}$ selesai Master akan mengecek respon (CRC) dan menunggu waktu $T_{3,5}$ selesai. Jika sebelum waktu $T_{3,5}$ selesai terdapat karakter yang diterima, maka Master akan memberi tanda (*flag*) *frame error*, ditandai dengan *flag* RI = 1.

Apabila $T_{3,5}$ selesai dan terdeteksi *frame error* atau nilai CRC salah, maka Master tidak akan mengirimkan pesan ke HMI. Namun jika $T_{3,5}$ selesai dan cek CRC benar, maka Master akan mengirimkan respon yang diterima dari Slave menuju HMI. Setelah proses ini, selanjutnya Master kembali menunggu perintah dari HMI.

Besarnya waktu $T_{1,5}$ dan $T_{3,5}$ untuk *baud rate* 9600 bps dihitung sebagai berikut:

$$T_{3,5} = 3{,}5 \times \frac{Jumlah\ bit\ per\ karakter}{Baud\ Rate}$$

$$= 3{,}5 \times \frac{11\ bit}{9600\ bit/s} = 0{,}004010\ s = 4010\ \mu s$$

$$T_{1,5} = 1{,}5 \times \frac{Jumlah\ bit\ per\ karakter}{Baud\ Rate}$$

$$= 1{,}5 \times \frac{11\ bit}{9600\ bit/s} = 0{,}001719\ s = 1719\ \mu s$$

Besarnya siklus mesin (*machine cycles*) mikrokontroler AT89S51 adalah 12 periode osilator, sehingga untuk frekuensi kristal 11,0592 Mhz adalah:

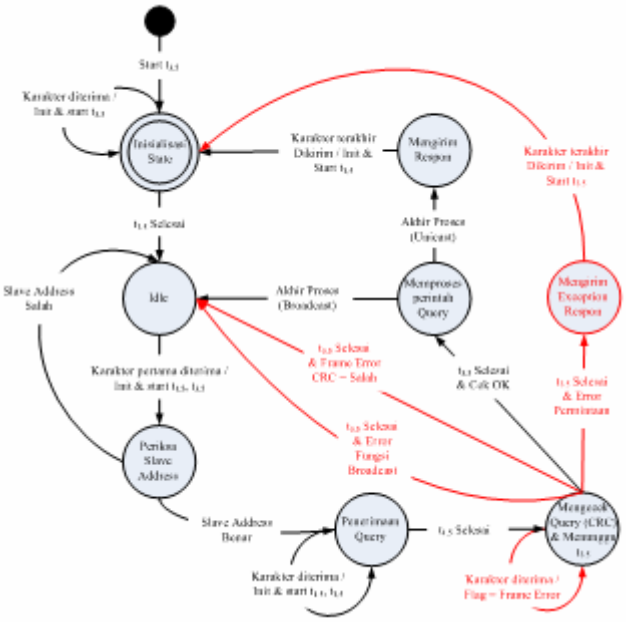
$$Siklus\ Mesin = \frac{12}{11{,}0592 Mhz} = 1{,}085 \mu s$$

Untuk menghasilkan waktu $T_{1,5}$ = 1719µs dan $T_{3,5}$ = 4010µs, maka Timer harus mencacah sebanyak :

$$\text{Cacah } T_{1,5} = \frac{1719\ \mu s}{1{,}085 \mu s} = 1584 \text{ , isi Timer = -1584.}$$

$$\text{Cacah } T_{3,5} = \frac{4010\ \mu s}{1{,}085 \mu s} = 3696 \text{ , isi Timer = -3696.}$$

3.4.1.2 Program Slave



Gambar 3.5 Diagram keadaan program Slave.

Program Slave diawali dengan inisialisasi awal, yang meliputi inisialisasi mode serial dan *baud rate*, mode Timer, dan mode terima RS485. Mode komunikasi serial yang digunakan sama seperti Master yaitu Mode 3 (9 bit UART), dengan SM2 = 0, TB8 = 0 dan *baud rate* 9600 bps. Pada inisialisasi awal, RS485 diset pada mode terima.

Kemudian program Slave melakukan inisialisasi dan mulai menghitung waktu 3,5 karakter ($T_{3,5}$). Jika sebelum $T_{3,5}$ selesai terdapat data serial yang masuk, maka akan dilakukan inisialisasi dan penghitungan ulang $T_{3,5}$. Setelah $T_{3,5}$ selesai, selanjutnya Slave menunggu perintah dari Master dan siap melakukan komunikasi.

Jika Slave menerima data serial, maka Slave akan memeriksa data tersebut sebagai Slave *Address*. Jika Slave *Address* tidak sesuai dengan alamat Slave penerima, maka Slave akan kembali menunggu. Tetapi jika Slave *Address* sesuai, maka

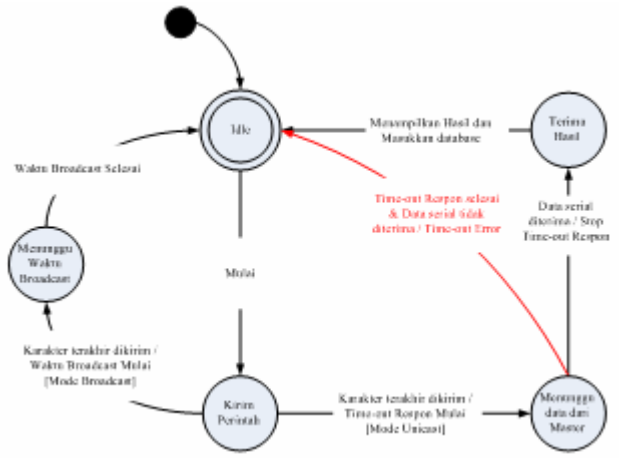
Slave akan menerima *query* tersebut. Setiap menerima karakter, Slave akan melakukan inisialisasi dan menghitung ulang $T_{1,5}$ dan $T_{3,5}$.

Selanjutnya Slave akan mengecek *query* (CRC) dan menunggu $T_{3,5}$. Jika terjadi kesalahan *frame*, CRC salah atau pada mode *broadcast* fungsi yang dipilih tidak mendukung mode *broadcast*, maka Slave tidak melakukan aksi atas perintah tersebut dan menunggu perintah berikutnya. Jika tidak terjadi kesalahan *frame* atau CRC benar, tetapi Slave tidak dapat menangani perintah tersebut, sebagai contoh perintah dengan nomer fungsi yang tidak didukung, maka Slave akan mengirim *exception response*. Apabila *frame* benar dan Slave bisa menangani perintah dari Master, maka Slave akan memproses *query* tersebut.

Pada mode *broadcast*, setelah selesai memproses perintah *query* Slave akan siap menerima perintah berikutnya tanpa mengirimkan respon. Sedangkan pada mode *unicast*, Slave akan mengirimkan respon sebagai jawaban atas *query* dari Master. Setelah proses ini, Slave akan melakukan inisialisasi dan mulai menghitung $T_{3,5}$, dan siap melakukan komunikasi berikutnya".

3.4.2 Program pada Komputer

Program pada komputer dibangun dengan menggunakan bahasa pemrograman Visual C++. Program tersebut membentuk sebuah HMI (*Human Machine Interface*). HMI berfungsi untuk memberikan perintah kepada Master untuk melakukan komunikasi dengan Slave.



Gambar 3.6 Diagram keadaan program HMI.

IV. PENGUJIAN DAN ANALISA

Pengujian dilakukan dengan menjalankan komunikasi fungsi-fungsi protokol Modbus antara Master dan Slave melalui HMI. Pengujian tersebut meliputi pengujian validitas fungsi, pengujian *delay query*, dan pengujian panjang kabel.

4.1 Pengujian Validitas Fungsi

Pengujian validitas fungsi dilakukan untuk mengetahui keberhasilan fungsi-fungsi telekontrol protokol Modbus dalam komunikasi antara Master dan Slave. Pengujian dilakukan pada semua fungsi yang didukung sistem telekontrol SCADA Modbus, untuk mode pengalamatan *unicast* dan *broadcast*. Pengujian juga dilakukan untuk Slave ID dan kode fungsi yang tidak didukung oleh sistem.

4.1.1 Mode Pengalamatan *Unicast*

Pada mode pengalamatan *unicast*, perintah (*query*) dari Master hanya ditujukan kepada satu Slave. Hasil pengujian ditunjukkan pada Tabel 4.1.

Tabel 4.1 Hasil pengujian validitas fungsi mode *unicast* untuk kode fungsi 05, 06, 15 dan 16.

Fung.	Slave	Alamt	Data	Coil	Output Analog	Respn
05	1	0000	FF00	Coil1 ON	-	OK
		0007	FF00	Coil8 ON	-	OK
		0007	0000	Coil8 OFF	-	OK
	2	0001	FF00	Coil2 ON	-	OK
		0006	FF00	Coil7 ON	-	OK
		0006	0000	Coil7 OFF	-	OK
06	1	0000	0001	-	1	OK
		0000	00FF	-	255	OK
	2	0000	0010	-	16	OK
		0000	0080	-	128	OK
15	1	0000	07	Coil1-3 ON	-	OK
		0000	FF	Semua Coil ON	-	OK
	2	0000	0F	Coil1-4 ON	-	OK
		0000	FF	Semua Coil ON	-	OK
16	1	0000	0004	-	4	OK
		0000	00FF	-	255	OK
	2	0000	0020	-	32	OK
		0000	0080	-	128	OK

Dari hasil pengujian mode pengalamatan *unicast* terlihat bahwa semua fungsi dapat berjalan dengan baik, baik pada Slave 1 maupun Slave 2. Perintah pengesetan status coil dan pengesetan nilai output analog yang dikirimkan Master, dapat diterma dan dikerjakan dengan baik oleh Slave.

4.1.2 Mode Pengalamatan *Broadcast*

Pada mode pengalamatan *broadcast*, perintah dari Master ditujukan pada semua Slave yang terdapat dalam sistem. Hasil pengujian ditunjukkan pada Tabel 4.2.

Tabel 4.2 Hasil pengujian validitas fungsi mode *broadcast* untuk kode fungsi 05, 06, 15 dan 16.

Fungsi	Alamat	Data	Monitoring fungsi 01,03		Pengamatan LED	
			Slave 1	Slave 2	Slave 1	Slave 2
05	0000	FF00	Coil1 ON	Coil1 ON	Berhasil	Berhasil
	0007	FF00	Coil8 ON	Coil8 ON	Berhasil	Berhasil
06	0000	0001	OA = 1	OA = 1	Berhasil	Berhasil
	0000	00FF	OA = 255	OA = 255	Berhasil	Berhasil
15	0000	01	Coil1 ON	Coil1 ON	Berhasil	Berhasil
	0000	0F	Coil1-4 ON	Coil1-4 ON	Berhasil	Berhasil
16	0000	0001	OA = 1	OA = 1	Berhasil	Berhasil
	0000	00FF	OA = 255	OA = 255	Berhasil	Berhasil

Dari hasil pengujian terlihat bahwa mode pengalamatan *broadcast* dapat berjalan dengan baik. Master dapat memberi perintah kepada Slave 1 dan Slave 2 dalam waktu bersamaan untuk melakukan pengesetan status coil atau pengesetan nilai output analog (OA).

4.1.3 Pengujian Fungsi Invalid

Hasil pengujian fungsi invalid dan kondisi *time-out error* ditunjukkan pada Tabel 4.3.

Tabel 4.3 Hasil pengujian fungsi invalid dan *time-out error*.

Slave ID	Fungsi	Respon	Keterangan
1	7	Kode fungsi = 87 H, Data = 01	Fungsi Invalid
1	8	Kode fungsi = 88 H, Data = 01	Fungsi Invalid
3	5	Respon tidak dikirimkan	Time-out Error
3	7	Respon tidak dikirimkan	Time-out Error

Dari hasil pengujian ini terlihat bahwa fungsi 7 dan 8 merupakan fungsi invalid. Hal ini dikarenakan sistem telekontrol SCADA Modbus tidak mendukung fungsi-fungsi tersebut. Keadaan fungsi invalid dapat dilihat pada respon yang dikirimkan Slave, dimana kode fungsi ditambahkan 80H dan data diisi 01, keadaan ini disebut *exception response*. Pengiriman *query* dengan Slave ID 03 memberikan keadaan *time-out error*. Hal ini karena sistem telekontrol SCADA Modbus hanya memiliki 2 Slave yang aktif, yaitu Slave 1 dan Slave 2.

4.2 Pengujian *Delay Query*

Pengujian dilakukan dengan mengeset *delay* waktu pengiriman antar karakter pesan *query*, yang dikirimkan dari Master ke Slave. Pengujian ini bertujuan untuk mengetahui batas *delay* waktu yang masih dapat terjadi untuk proses pengiriman *query* yang valid. Dalam aturan standar Modbus besar *delay* tersebut adalah waktu 1,5 karakter ($T_{1,5}$), yaitu 0,00171875 s untuk *baud rate* 9600 bps. Untuk menghasilkan *delay* tersebut dengan Timer 0, maka Timer ini harus mencacah sebanyak 1584 kali, isi Timer -1584 yaitu TH0 = F9 hex = 249 dan TL0 = D0 hex = 208 untuk kristal 11,0592 Mhz. Hasil pengujian untuk variasi *delay query* ditunjukkan pada Tabel 4.4.

Tabel 4.4 Hasil pengujian *delay query*.

TH0	TL0	Delay (µs)	05		06		15		16	
			1	2	1	2	1	2	1	2
249	208	1718,75	Nok	Nok	Nok	Nok	Nok	Nok	Nok	Nok
250	0	1666,67	Nok	Nok	OK	Nok	OK	Nok	Nok	Nok
250	5	1661,24	Nok	OK	Nok	OK	OK	OK	Nok	OK
250	6	1660,15	Nok	Nok	OK	OK	Nok	OK	OK	OK
250	7	1659,07	OK	OK	OK	Nok	Nok	Nok	OK	OK
250	8	1657,98	OK	OK	OK	OK	OK	OK	Nok	OK
250	9	1656,90	OK	OK	OK	OK	OK	OK	OK	OK
250	10	1655,81	OK	OK	OK	OK	OK	OK	OK	OK

Hasil pengujian *delay query* menunjukkan bahwa komunikasi antara Master dan Slave akan berjalan dengan baik tanpa *error* dengan *delay* waktu pengiriman antar karakter pesan tidak lebih besar dari 1656,90 µs. Standar Modbus menyebutkan bahwa komunikasi dapat berjalan dengan baik untuk *delay* antar karakter maksimum $T_{1,5}$ atau 1718,75 µs. Terdapat sedikit perbedaan *delay* waktu sistem yang dibuat dengan standar Modbus, yaitu 61,85 µs. Hal ini disebabkan oleh perbedaan waktu eksekusi mnemonic program, dimana Master mengirimkan karakter dengan rutin biasa, sedangkan Slave menerima karakter dengan rutin interupsi.

4.3 Pengujian Panjang Kabel

Pengujian ini dilakukan dengan menggunakan panjang kabel 6 meter dan 100 meter. Hasil pengujian ditunjukkan pada Tabel 4.5.

Tabel 4.5 Hasil pengujian variasi panjang kabel.

Panjang Kabel (m)	05		06		15		16	
	1	2	1	2	1	2	1	2
100	OK	OK	OK	OK	OK	OK	OK	OK
6	OK	OK	OK	OK	OK	OK	OK	OK

Dari hasil pengujian variasi panjang kabel terlihat bahwa komunikasi antara Master dan Slave sistem telekontrol SCADA Modbus dapat berjalan dengan baik, baik untuk panjang kabel 6 meter maupun 100 meter. Semua fungsi protokol Modbus yang didukung, yaitu fungsi 05, 06, 15 dan 16 dapat berfungsi dengan baik. Hal ini karena sistem telekontrol SCADA Modbus yang dibuat menggunakan komunikasi serial RS485 yang memiliki jarak komunikasi maksimum sampai 4000 feet.

V. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisa yang dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

1. Sistem telekontrol SCADA Modbus yang dibuat mendukung fungsi protokol Modbus 05, 06, 15 dan 16 sebagai fungsi pengesetan (*telecontrolling*), serta fungsi 01, 02, 03 dan 04 sebagai monitoring.
2. Hasil pengujian menunjukkan fungsi 05, 06, 15 dan 16 dapat berjalan dengan baik dengan menggunakan mode pengalamatan *unicast* dan *broadcast*.
3. Pengiriman *query* dengan Slave ID yang tidak didukung akan menghasilkan *time-out error*.
4. Pengiriman *query* dengan kode fungsi yang tidak didukung akan menghasilkan *exception response*, dengan kode fungsi ditambah 80H dan *data field* berisi 01.
5. Master dan Slave dapat berkomunikasi dengan baik tanpa *error* dengan *delay* waktu pengiriman antar karakter pesan tidak lebih besar dari 1656,90 µs.
6. Berdasarkan hasil pengujian, komunikasi antara Master dan Slave dapat berjalan dengan baik menggunakan panjang kabel 6 meter dan 100 meter.

5.2 Saran

1. Perancangan Master dan Slave dapat dilakukan menggunakan AVR dengan bahasa C sebagai pembentuk protokol Modbus.
2. Waktu 3,5 karakter ($T_{3,5}$) dan waktu 1,5 karakter ($T_{1,5}$) dianjurkan menggunakan 2 Timer.
3. Untuk pengembangan sistem telekontrol SCADA Modbus perangkat input-output Slave dapat ditambah, baik untuk coil, input digital, output analog dan input analog.

DAFTAR PUSTAKA

[1] Agfianto Eko Putra, *Belajar Mikrokontroler AT89C51/52/55 (teori dan aplikasi)*, Gava Media, Yogyakarta, 2002.

[2] Axelson, Jan, *Networks for Monitoring and Control Using an RS-485 Interface*, Microcomputer Journal, 1995.

[3] Kadir, Abdul, *Panduan Pemrograman Visual C++*, Andi, Yogyakarta, 2004.

[4] Kadir, Abdul, *Pemrograman C++*, Andi, Yogyakarta, 1995.

[5] Nelson, Todd, *The Practical Limits of RS-485*, National Semiconductor, 1995.

[6] Pandjaitan Bonar, *Teknologi Sistem Pengendalian Tenaga Listrik Berbasis SCADA*, Prenhallindo, Jakarta, 1999.

[7] Sivasothy, Sivakumar, *Transceivers and Repeaters Meeting the EIA RS-485 Interface Standard*, National Semiconductor, 1998.

[8] Sudjadi, *Teori dan Aplikasi Mikrokontroler*, Graha Ilmu, Semarang, 2005.

[9] ……., http://www.delta-electronic.com

[10] ……..,http://www.modicon.com/techpubs/toc7.html

[11] ……., http://www.modbus.org

[12] ……., *Modbus Over Serial Line Specification and Implementation Guide V1.0.*

[13] ……., *Modbus Protocol Reference Guide*, M-System CO., LTD., Osaka, Japan.

[14] ……., *Modicon Modbus Protocol Reference Guide PI-MBUS-300*, Modicon Inc., North Andover, Massachusetts, 1996.

Agus Tiyono. Dilahirkan di Pemalang 19 Mei 1983. Menempuh pendidikan menengahnya di SMUN 1 Pemalang. Saat ini sedang menempuh pendidikan tinggi di Jurusan Teknik Elektro Universitas Diponegoro dengan konsentrasi Kontrol.

Semarang, Januari 2007

Mengetahui/Mengesahkan,

Pembimbing I	Pembimbing II
Ir. Sudjadi, MT.	Iwan Setiawan, ST. MT.
NIP. 131 558 567	NIP. 132 283 183