RANCANG BANGUN SISTEM SCADA

BERBASIS RASPBERRY PI

LAPORAN TUGAS AKKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Elektro

Disusun Oleh :

Putra Arisandy

11524082

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2018

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM SCADA

BERBASIS RASPBERRY PI

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat untuk Memperoleh
Gelar Sarjana Teknik
Pada Program Studi Teknik Elektro
Fakultas Teknlogi Industri
Universitas Islam Indonesia

Disusun oleh :

Putra Arisandy
11524082

Yogyakarta 18–04-2018

Menyetujui,

Pembimbing 1	Pembimbing 2
Sisdarmanto Adinandra, S.T., M.Sc., Ph.D	**Medilla Kusriyanto, S.T., M.Eng**
025240101	**015240101**

LEMBAR PENGESAHAN PENGUJI

RANCANG BANGUN SISTEM SCADA BERBASIS RASPBERRY PI

TUGAS AKHIR
Disusun oleh:

Putra Arisandy
11524082

Telah dipertahankan didepan sidang penguji
sebagai Salah Satu Syarat untuk Memperoleh
Gelar Sarjana Teknik
pada Program Studi Teknik Elektro
Fakultas Teknologi Industri
Universitas Islam Indonesia

Yogyakarta, 11 Mei 2018

Tim Penguji,

Ketua
Sisdarmanto Adinandra, S.T., M.Sc., Ph.D. .....

Anggota I
Alvin Sahroni, ST, M.Eng, Ph.D. .....

Anggota II
Dwi Ana Ratna Wati, ST, M.Eng. .....

Mengetahui,

Ketua Jurusan Teknik Elektro
Universitas Islam Indonesia

Dr. Eng. Hendra Setiawan, S.T., M.T.

LEMBAR PERNYATAAN KEASLIAN

Saya yang bertanda tangan dibawah ini:

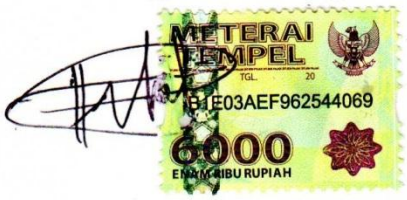
Nama : Putra Arisandy

No. Mahasiswa : 11524082

Dengan ini menyatakan bahwa:

1. Skripsi ini tidak mengandung karya yang diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan Saya juga tidak mengandung karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.
2. Informasi dan materi Skripsi yang terkait hak milik, hak intelektual, dan paten merupakan milik bersama antara tiga pihak yaitu penulis, dosen pembimbing, dan Universitas Islam Indonesia. Dalam hal penggunaan informasi dan materi Skripsi terkait paten maka akan diskusikan lebih lanjut untuk mendapatkan persetujuan dari ketiga pihak tersebut diatas.

Yogyakarta, 18 April 2018



Putra Arisandy

KATA PENGANTAR

Assalamualaikum wr.wb.,

Segala puji bagi Allah SWT, Tuhan semesta alam yang telah melimpahkan rahmat dan hidayah kepada hamba-Nya, sehingga tugas akhir ini dapat diselesaikan. Selawat dan salam semoga tercurah kepada Rasulullah Muhammad SAW beserta para keluarganya, sahabat dan para pengikutnya hingga akhir zaman. Tugas akhir yang berjudul "Rancang Bangun Sistem SCADA Berbasis Raspberry Pi" ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Program Studi Teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Pada kesempatan ini, ungkapan rasa terima kasih yang sebesar-besarnya diucapkan kepada berbagai pihak yang telah memberikan doa, bantuan, bimbingan, dukungan, kerja sama, fasilitas dan kemudahan lainnya. Untuk itu, dengan ketulusan hati saya mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Kedua orang tua saya yang telah membesarkan dan senantiasa mendukung saya tanpa henti – hentinya..
2. Bapak Hendra Setiawan, S.T., M.T., Ph.D. selaku Ketua Program Studi Teknik Elektro Universitas Islam Indonesia.
3. Bapak Sisdarmanto Adinandra, S.T., M.Sc., Ph.D dan Bapak Medilla Kusriyanto, S.T., M.Eng selaku Dosen Pembimbing Tugas Akhir yang telah meluangkan waktu dan bantuannya sampai terselesaikannya Tugas Akhir ini.
4. Segenap keluarga Teknik Elektro UII atas pengalaman, senyuman, ilmu bermanfaat, persaudaraan, dukungan, dan masih banyak lagi yang tak bisa disampaikan dengan perkataan.
5. Seluruh dosen dan staff Jurusan Teknik Elektro UII yang memberikan ilmu dan pengalaman untuk mempersiapkan diri saya di dunia kerja sebagai seorang *electrical engineer*.
6. Dan banyak pihak lain yang tidak dapat kami sebutkan seluruhnya yang telah membantu dalam penyelesaian Tugas Akhir ini.

Penulis menyadari bahwa dalam laporan tugas akhir ini masih jauh dari sempurna. Oleh karena itu penulis mengharapkan kritik dan saran membangun dari semua pihak demi kemajuan penulis

di masa mendatang. Harapan penulis laporan tugas akhir ini dapat membantu mengembangkan ilmu pengetahuan penulis pada khususnya dan pembaca pada umumnya.

Wassalamu'alaikum wr.wb.

Yogyakarta, 18 April 2018

ABSTRAK

Sistem SCADA berbasis Raspberry Pi untuk menggantikan fungsi komputer cukup efektif untuk meminimalkan biaya pengadaan dan biaya operasional. Selain itu perancangan sistem SCADA berbasis *open source* juga dapat menggantikan *software* SCADA yang memiliki lisensi berbayar. Pada penelitian ini sistem SCADA pada Raspberry Pi digunakan untuk mengakusisi data dari tiga buah *slave* mikrokontroler dengan komunikasi RS485. Satu buah *slave* juga dihubungkan dengan PLC LG Master K200S dengan komunikasi RS232C untuk mengakusisi data proses produksi pada *plant* konveyor. Kedua komunikasi RS485 dan RS232C dilakukan dengan menggunakan metode *request* dan *response*. Dimana master melakukan *request* terhadap *slave* yang dituju, kemudian *slave* membalas sesuai dengan *request* yang diterima. Untuk melakukan komunikasi dengan seluruh *slave* menggunakan komunikasi RS485, protokol komunikasi dibangun dengan format *header*, alamat, data, dan *tail* agar dapat melakukan proses *reading/writing* menuju alamat *slave* yang tepat. Komunikasi dengan PLC menggunakan komunikasi RS232C menggunakan format protokol yang telah ditentukan pada *datasheet* PLC LG Master K *series*. Segala sistem yang menyangkut *hardware* dan *software* pada sistem ini telah bekerja sesuai dengan tujuan penelitian. Metode *request* dan *response* menggunakan protokol RS485 dan RS232C dapat dieksekusi sesuai dengan alamat *slave* yang dituju. Untuk melakukan komunikasi dengan seluruh *slave* menggunakan komunikasi RS485, waktu sampling yang digunakan sebesar 333 *ms* per *slave* dengan tingkat *error* data saat melakukan komunikasi sebesar 22,66 % yang disebabkan oleh *noise*.

Kata Kunci : Raspberry Pi SCADA, Master Slave RS485, LG Master K Arduino.

DAFTAR ISI

LEMBAR PENGESAHAN i
LEMBAR PENGESAHAN PENGUJI ii
LEMBAR PERNYATAAN KEASLIAN iii
KATA PENGANTAR iv
ABSTRAK vi
DAFTAR ISI vii
DAFTAR GAMBAR ix
DAFTAR TABEL x
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang Masalah 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Penelitian 2
1.5 Manfaat Penelitian 2
BAB 2 TINJAUAN PUSTAKA 3
2.1 Studi Literatur 3
2.2 Komunikasi Serial/UART Raspberry Pi 3
2.3 Komunikasi RS485 4
2.4 Komunikasi RS232C Master K200S 4
BAB 3 PERANCANGAN SISTEM 8
3.1 Pembuatan *Hardware* 9
3.1.1 *Level Shifter* dan RS485 *Master* 9
3.1.2 RS485 *Slave* 1 dan *Slave* 2 9
3.1.3 RS485 dan RS232C *Slave* 3 10
3.2 Pembuatan Program 11

3.2.1 Diagram *Ladder* PLC 11
3.2.2 Pembuatan GUI 12
3.2.3 Protokol Komunikasi RS485 13
3.2.4 Protokol Komunikasi RS232C 15
BAB 4 HASIL DAN PEMBAHASAN 19
4.1 Pengujian dan Analisa Komunikasi Raspberry Pi dengan Seluruh Slave 19
4.2 Pengujian dan Analisa Konverter RS485 – RS232C (Slave 3) 21
BAB 5 KESIMPULAN DAN SARAN 24
5.1 Kesimpulan 24
5.2 Saran 24
DAFTAR PUSTAKA 25
LAMPIRAN 26

DAFTAR GAMBAR

Gambar 3.1 Diagram blok sistem 8

Gambar 3. 2 Diagram blok *master* 9

Gambar 3.3 Diagram blok *slave* 1 dan 2 10

Gambar 3.4 Diagram blok *slave* 3 10

Gambar 3.5 Perbedaan konfigurasi kabel RS232C pada *external device* 11

Gambar 3.6 Tampilan GUI 12

Gambar 3.7 *Frame* balasan dari *slave* 13

Gambar 3.8 Nilai - nilai pada I/O dan *register* dipisahkan dengan *separator* [spasi] 14

Gambar 3. 9 *Frame* penulisan menuju *slave* 1 dan 2 15

Gambar 3.10 *Frame* penulisan menuju *slave* 3 15

Gambar 3.11 *Frame* untuk membaca 6 *register* PLC 16

Gambar 3.12 Pembagian frame pada bagian data 16

Gambar 3.13 Perintah untuk memberikan nilai ke *register* M000 17

Gambar 3.14 Perintah untuk memberikan nilai ke *register* D000 17

Gambar 4.1 Komunikasi RS485 berjalan dengan semestinya 19

Gambar 4.2 Proses *writing* port B0 slave 1 19

Gambar 4.3 Pengiriman data dari *master* tidak mendapat tanggapan dari *slave* 3 20

Gambar 4.4 Data yang diterima dari *slave* 2 mengalami kerusakan akibat *noise* 20

Gambar 4.5 *Frame* protokol balasan PLC 22

DAFTAR TABEL

Tabel 2.1 Daftar instruksi *frame instruction* dan *instruction type* 6

Tabel 2.2 Daftar *register* yang dapat dieksekusi melalui komunikasi RS232C..... 7

Tabel 3.1 Pengalamatan *register plant* konveyor..... 12

Tabel 3.2 *Frame* protokol *reading* data dari *master* menuju *slave* 13

Tabel 4.1 Kerusakan data pada komunikasi RS485..... 21

BAB 1
PENDAHULUAN

1.1 Latar Belakang Masalah

Kemampuan perangkat – perangkat elektronik seperti komputer dan *smartphone* yang dapat dieksplorasi secara *open source* memungkinkan *user* untuk ikut serta dalam proses pengembangan. Pada dunia industri maupun *home automation* perangkat seperti *smartphone* dan komputer banyak digunakan dalam sistem kontrol dan monitoring seperti :

1. Sistem kontrol dan monitoring proses produksi
2. *Data logger*
3. Sistem kontrol instalasi penerangan
4. Sistem monitoring penggunaan daya beban, dan lain – lain.

Dalam dunia automasi, sistem kontrol dan monitoring berbasis *Graphical User Interface* (GUI) biasa disebut dengan *Supervisory Control And Data Acquisition* (SCADA). SCADA dapat diartikan sebagai pengumpul data dan sistem pengendali yang ditampilkan dalam bentuk antarmuka untuk memudahkan proses monitoring. Perancangan dan pengoperasian SCADA tidak hanya terbatas pada *personal computer* (PC) dan *smartphone*. SCADA juga dapat dijalankan pada sebuah perangkat *mini* PC Raspberry Pi.

Penggunaan Raspberry Pi sebagai pengganti PC dapat memangkas biaya pengadaan dan operasional yang harus dikeluarkan. Selain itu sistem SCADA berbasis *open source* juga dapat menggantikan fungsi *software* SCADA yang memiliki lisensi berbayar.

Pada penelitian ini, Raspberry Pi digunakan sebagai *master* proses kontrol instalasi penerangan, monitoring suhu, dan monitoring proses produksi pada plant konveyor. Raspberry Pi akan dihubungkan dengan 3 *slave* mikrokontroler menggunakan komunikasi RS485. Salah satu *slave* akan terhubung dengan *Programmable Logic Controller* (PLC) melalui komunikasi RS232C.

Proses komunikasi antar kontroler membutuhkan sebuah protokol komunikasi agar proses pertukaran data dapat diterima sesuai dengan alamat yang dituju. Protokol komunikasi RS485 dirancang sendiri oleh peneliti, Sedangkan komunikasi RS232C dengan PLC menggunakan protokol yang sudah ditetapkan oleh *vendor* pembuat sesuai dengan *datasheet* PLC tersebut.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini adalah bagaimana membangun aplikasi SCADA pada Raspberry Pi dan bagaimana melakukan komunikasi dengan protokol RS485 dan RS232C.

1.3 Batasan Masalah

1. PLC yang digunakan adalah LG Master K200S modular 32 *input* digital dan 16 *output* digital.
2. Struktur *frame* protokol PLC yang digunakan hanya berdasarkan alamat – alamat *register* yang digunakan untuk mengoperasikan *plant* konveyor.

1.4 Tujuan Penelitian

1. Membangun format protokol komunikasi RS485 antara Raspberry Pi dengan mikrokontroler.
2. Mengakses PLC melalui mikrokontroler menggunakan protokol RS232C.

1.5 Manfaat Penelitian

1. Memberikan sebuah sistem SCADA menggunakan Raspberry Pi sebagai pengganti PC.
2. Mengetahui cara membangun *frame* protokol dengan komunikasi RS485.
3. Mengetahui cara mengakses PLC melalui protokol RS232C.
4. Mampu membuat modul ekspansi untuk PLC dengan komunikasi RS232C.

BAB 2
TINJAUAN PUSTAKA

2.1 Studi Literatur

Penelitian sejenis pernah dilakukan oleh Papasideris dkk [1] dimana pada penelitian tersebut komunikasi RS485 digunakan untuk menghubungkan tiga perangkat kontroler untuk melakukan akusisi data suhu. Selain itu pada penelitian tersebut juga dijelaskan bagaimana membangun protokol untuk melakukan komunikasi antar kontroler.

Penelitian selanjutnya dilakukan oleh Xu dkk [2] dimana pada penelitian tersebut Raspberry Pi terhubung dengan berbagai *environmental sensors* yang digunakan untuk mengukur probabilitas curah hujan. Komunikasi yang digunakan oleh Raspberry Pi untuk terhubung dengan sensor – sensor tersebut adalah komunikasi RS485.

Penelitian lainnya dilakukan oleh Huh dkk [3] dimana pada penelitian tersebut juga membahas tentang komunikasi antar kontroler untuk melakukan monitoring terhadap *micro grid*. Dimana pada penelitian tersebut menggunakan dua buah kontroler yaitu Arduino UNO dan PLC berjenis NC-EPLC yang saling berkomunikasi secara *serial*.

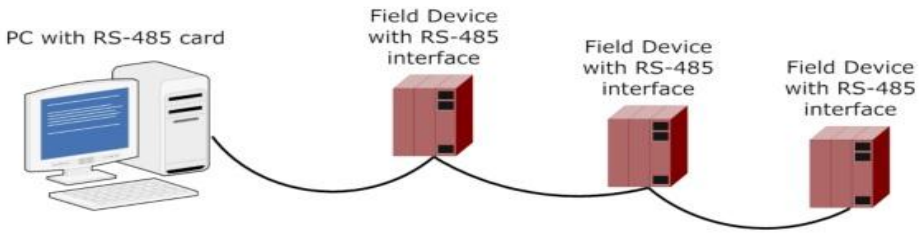
2.2 Komunikasi Serial/UART Raspberry Pi

Raspberry Pi 3 memiliki 2 buah *built- in Universal Asynchronous Receiver – Transmitter* (UART) yaitu PL011 dan *mini* UART. Keduanya memiliki cara kerja yang berbeda dimana PL011 digunakan sebagai modul Bluetooth dan Mini UART digunakan sebagai *Linux Console* yaitu sebuah sistem untuk mengakses kernel linux. Untuk melakukan komunikasi *serial* menggunakan pin *General Purpose Input Output* (GPIO) pada Raspberry Pi 3, perangkat yang digunakan adalah *mini* UART.

Mini UART bekerja pada tegangan 3,3V sehingga butuh penanganan ekstra saat dihubungkan dengan RS485 *Converter*. Sebuah *adapter* penyetara tegangan harus digunakan untuk menjembatani kedua protokol apabila keduanya memiliki level tegangan yang berbeda. *Pin* pengirim dan penerima yang digunakan pada Raspberry Pi menggunakan pin 8 dan pin 10 pada GPIO *header* RaspberryPi 3 [4].

2.3 Komunikasi RS485

Komunikasi RS485 merupakan komunikasi yang menggunakan dua buah kabel untuk menghubungkan antar perangkat. RS485 banyak digunakan untuk menangani komunikasi jarak jauh mencapai 1220 meter tanpa menggunakan *repeater*. Kelebihan lain dari komunikasi RS485 adalah dapat menghubungkan lebih dari 32 perangkat komunikasi [5]. Karena kemampuan komunikasi jarak jauh dan kemampuan untuk melakukan komunikasi *multi-device* tersebut, RS485 menjadi solusi untuk menutupi kekurangan yang dimiliki oleh komunikasi RS232. Hubungan antar perangkat menggunakan komunikasi RS485 seperti ditunjukkan Gambar 2.1.



Gambar 2.1 Contoh komunikasi RS485

Komunikasi RS485 tidak mengatur standar protokol yang harus digunakan. Karena itu *use*r bebas dalam menentukan protokol yang akan digunakan untuk komunikasi antar perangkat. Kebebasan dalam penentuan protokol ini juga membuat RS485 menjadi dasar utama terbentuknya komunikasi yang sering digunakan pada dunia industri seperti Profibus, Interbus, dll. Dalam dunia automasi sering dijumpai perangkat – perangkat seperti RS232 *to* RS485 *Converter*, RS485 *Smart Switch*, RS485 *Repeater* yang banyak digunakan dalam dunia PLC, SCADA, dan *Remote Terminal Unit* (RTU). Semua perangkat tersebut dibangun menggunakan teknologi RS485.

2.4 Komunikasi RS232C Master K200S

Master K200S tipe A dan C (K3P-07AS dan K3P-07CS) memiliki fitur pendukung untuk melakukan komunikasi dengan *external device* tanpa harus menggunakan modul Cnet I/F [6]. Meskipun tidak memiliki seluruh fungsi yang dapat dilakukan oleh modul Cnet I/F, fitur tersebut sangat berguna bagi *user* yang ingin melakukan pengembangan sistem yang murah berbasis jaringan RS232C. Fungsi yang didukung oleh komunikasi RS232C meliputi :

1. Pembacaan dan penulisan individu
2. Pembacaan dan penulisan kontinyu
3. Monitoring status *Central Processing Unit* (CPU)
4. Monitoring status *register*

5. Monitoring proses eksekusi

Untuk melakukan komunikasi dengan *external device* menggunakan RS232C, terdapat standar protokol yang harus dipatuhi sesuai dengan *datasheet* PLC LG Master K200S. Kesalahan dalam mengaplikasikan protokol tersebut akan membuat proses penulisan dan pembacaan tidak akan dieksekusi oleh PLC. Secara umum standar protokol RS232C yang digunakan oleh Master K200S adalah seperti ditunjukan Gambar 2.2.

Gambar 2.2 Struktur *frame* protokol LG Master K200S

Keterangan :

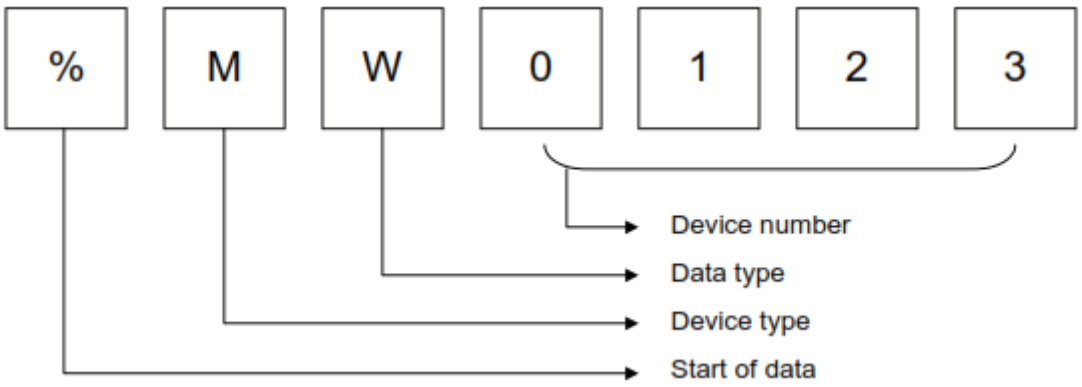
- Header (ENQ) : *Enquire* merupakan tanda awal sebuah data atau sering disebut dengan *Start of Request Frame*.
- Station No : Alamat *station* PLC ditetapkan melalui *software* KGL-Win pada tab *parameters*.
- Instruction : Jenis instruksi untuk dieksekusi seperti *read/write*.
- Instruction Type : Tipe data intruksi untuk dieksekusi seperti : *single* bit, *multiple* bit *(block)*, dll
- Data : Tipe *register* yang akan dieksekusi
- Tail (EOT) : *End of Text* sebagai penanda berakhirnya sebuah data.
- BCC : Dieksekusi CPU secara otomatis jika terdapat penggunaan huruf kecil

Instruksi yang digunakan dalam *frame instruction* dan *instruction type* dijelaskan dalam Tabel 2.1. Bagian ini memuat perintah yang harus dilakukan oleh PLC seperti membaca nilai *register*, menulis nilai ke sebuah *register*, membaca status PLC, dll.

Tabel 2.1 Daftar instruksi *frame instruction* dan *instruction type*

Item		Instruction				Deskripsi
		Main Instruction		Instruction Type		
		Simbol	ASCII Code (hex)	Simbol	ASCII Code (hex)	
Read	Single	r (R)	h72 (h52)	SS	h5353	Membaca single bit atau word
	Continous	r (R)	h72 (h52)	SB	h5342	Membaca multiple bit atau word
Write	Single	w (W)	h77 (h57)	SS	h5353	Menulis single bit atau word
	Continuos	w (W)	h77 (h57)	SB	h5342	Menulis multiple bit atau word
Monitoring number registration		x (X)	h78 (h58)	-	-	Device register yang sedang dimonitoring
Monitoring execution		y (Y)	h79 (h59)	-	-	Eksekusi fungsi monitoring
Read CPU status		r (R)	h72 (h52)	ST	h5354	Membaca status PLC

Instruksi yang digunakan dalam *frame* data dapat dilihat pada Gambar 2.3. Bagian ini memuat alamat *register* yang akan diakses.



Gambar 2.3 Struktur *frame* "Data"

Keterangan :

- *Start of data* : Simbol "%" sebagai penanda dimulainya *frame* data.
- *Device type* : Tipe *register* yang akan dieksekusi.
- *Data type* : Tipe data bit (X) atau word (W).

- *Device number* : Alamat *register* yang akan dieksekusi.

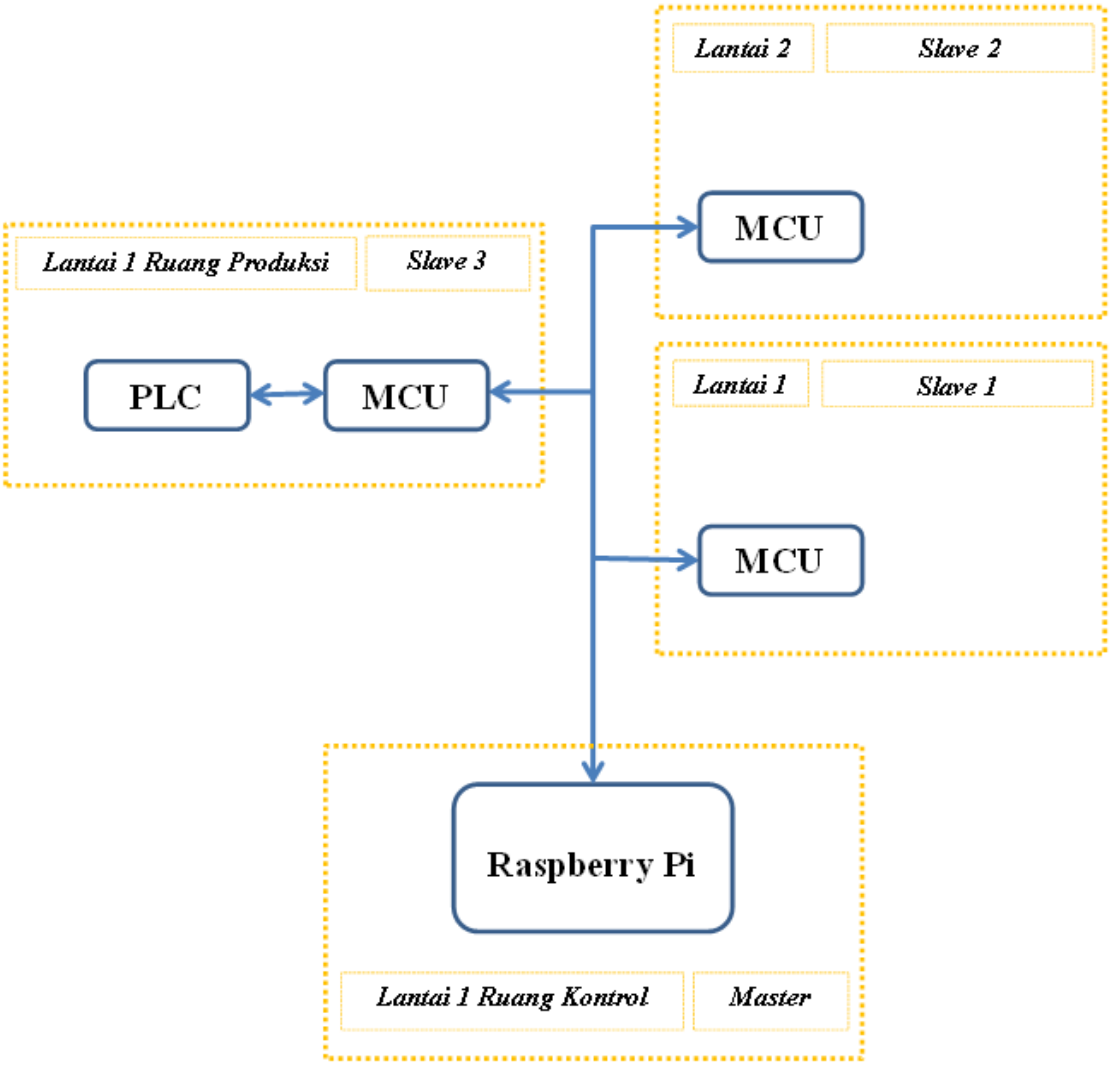
Jenis – jenis *register* yang dapat dieksekusi dengan protokol RS232C dapat dilihat pada Tabel 2.2.

Tabel 2.2 Daftar *register* yang dapat dieksekusi melalui komunikasi RS232C

Device Type	**Instruction**
P (I/O relay)	Read/Write
M (Auxiliary relay)	Read/Write
K (Keep relay)	Read/Write
L (Link relay)	Read/Write
F (Special relay)	Read
T (Timer contact relay)	Read/Write
T (Timer elapsed relay)	Read/Write
C (Counter contact relay)	Read/Write
C (Counter elapsed relay)	Read/Write
S (Step controller)	Read/Write
D (Data register)	Read/Write

BAB 3
PERANCANGAN SISTEM

Untuk membangun sistem SCADA berbasis Raspberry Pi ini, dilakukan dua tahapan yaitu : pembuatan *hardware* dan pembuatan *software*. Diagram blok sistem ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram blok sistem

Pada sistem ini Raspberry Pi berperan sebagai *master* untuk memantau dan mengatur seluruh proses yang ada pada *plant* antara lain : mengontrol instalasi penerangan, *monitoring* suhu ruangan, *monitoring* proses produksi pada *plant* konveyor, dan mengontrol proses produksi pada *plant* konveyor. Raspberry Pi terhubung dengan dua mikrokontroler ATMega16 dan satu Arduino Mega menggunakan komunikasi RS485.

Mikrokontroler ATMega16 digunakan untuk memantau suhu pada 8 ruangan menggunakan sensor LM35 dan menghidup matikan lampu pada 8 ruangan. Arduino Mega berperan untuk menerima perintah dari *master* melalui komunikasi RS485 dan meneruskannya menuju PLC LG Master K200S menggunakan komunikasi RS232C. PLC LG Master K200S berperan untuk *monitoring* dan mengendalikan proses produksi pada *plant* konveyor.

3.1 Pembuatan *Hardware*

Dalam sistem ini terdapat tiga tahap pembuatan *hardware* meliputi :

1. *Level shifter* dan RS485 *master*
2. RS485 *slave* 1 dan 2
3. RS485 & RS232C *slave* 3 dan PLC

3.1.1 *Level Shifter* dan RS485 *Master*

Untuk bisa melakukan komunikasi RS485 dengan ketiga *slave* mikrokontroler, Raspberry Pi membutuhkan sebuah *level shifter* dan RS485 *converter*. Hubungan dari ketiga perangkat ini dapat dilihat pada Gambar 3.2

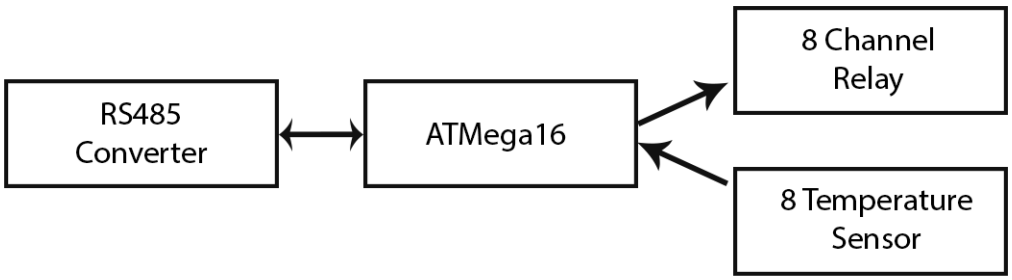
Gambar 3. 2 Diagram blok *master*

Fungsi dari *level shifter* adalah untuk menyetarakan tegangan antara *serial* Raspberry Pi dengan tegangan *serial* RS485. Raspberry Pi memiliki I/O dan *serial* yang bekerja pada *level* tegangan 3.3V sedangkan RS485 *converter* yang menggunakan modul MAX485 bekerja pada *level* tegangan 5V. Skematik secara lengkap dapat dilihat pada Lampiran 1.

3.1.2 RS485 *Slave* 1 dan *Slave* 2

Berbeda dari *master*, pada *slave* 1 dan *slave* 2 yang menggunakan mikrokontroler ATMega16 dapat langsung terhubung dengan RS485 *converter* tanpa harus menggunakan *level shifter*. Hal tersebut dikarenakan *level* tegangan I/O dan *serial* pada ATMega16 dan *level* tegangan pada RS485 *converter* sama – sama bekerja pada tegangan 5V. Diagram blok

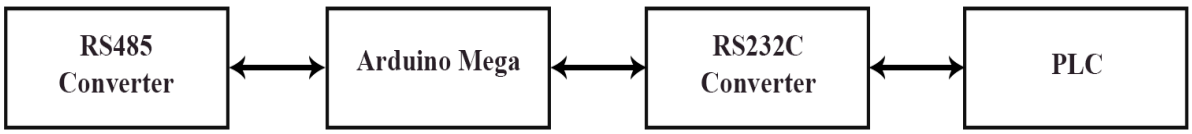
rangkaian *slave* 1 dan 2 dapat dilihat pada Gambar 3.3. Skematik secara lengkap dapat dilihat pada Lampiran 2.



Gambar 3.3 Diagram blok *slave* 1 dan 2

3.1.3 RS485 dan RS232C *Slave* 3

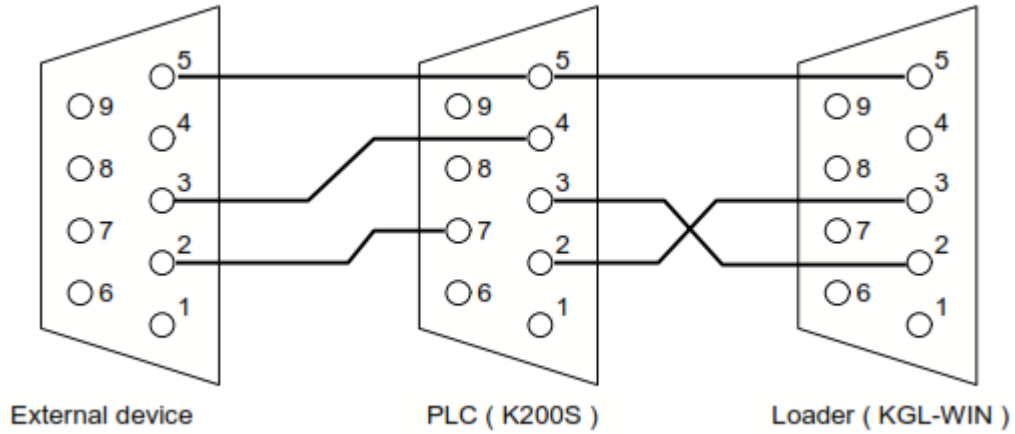
Fungsi utama *slave* 3 pada sistem ini adalah untuk melakukan akusisi data *register* PLC dan meneruskannya saat terdapat *request* dari *master*. Seperti halnya sebuah *Random Access Memory* (RAM) pada komputer, dengan cara ini proses pembacaan *register* PLC menjadi lebih cepat karena data telah diolah dan disimpan di memori *slave* 3. Tujuannya adalah untuk mendapatkan hasil pembacaan *register* secepat mungkin tanpa harus menunggu satu periode *sampling* dari *master*. Diagram blok sistem *slave* 3 dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram blok *slave* 3

Slave 3 menggunakan prinsip yang sama seperti halnya *slave* 1 dan 2, namun terdapat tambahan rangkaian RS232C *converter* untuk melakukan komunikasi dengan PLC LG Master K200S. Mikrokontroler yang digunakan pada *slave* 3 adalah Arduino Mega berbasis ATMega2560 yang memiliki 4 *port serial*. Tujuan digunakannya Arduino Mega pada rangkaian ini adalah karena *slave* 3 membutuhkan dua *port* komunikasi serial. *Port serial* pertama digunakan untuk melakukan komunikasi RS485 dengan *master*. Sedangkan *port serial* kedua digunakan untuk komunikasi RS232C dengan PLC. Skematik secara lengkap dapat dilihat pada lampiran 3.

Konfigurasi kabel yang digunakan pada RS232C yang digunakan untuk melakukan komunikasi dengan *external device* juga berbeda dengan konfigurasi kabel RS232C pada umumnya.



Gambar 3.5 Perbedaan konfigurasi kabel RS232C pada *external device*

Pada Gambar 3.5 dapat dilihat perbedaan skematik saat PLC terhubung dengan komputer/*loader* (KGL-Win) dan saat PLC terhubung dengan *external device*. Saat terhubung dengan komputer, *pin* yang digunakan sama seperti kabel RS232 pada umumnya dimana *pin* 2 dan 3 pada konektor DB9 terhubung saling silang. Sedangkan saat PLC terhubung dengan *external device*, *pin* yang gunakan pada konektor DB9 PLC adalah *pin* 4 dan 7.

3.2 Pembuatan Program

Pembuatan program akan dilakukan dalam 3 tahap meliputi :

1. Diagram *Ladder* PLC
2. Pembuatan GUI
3. Protokol komunikasi RS485
4. Protokol komunikasi RS232

3.2.1 Diagram *Ladder* PLC

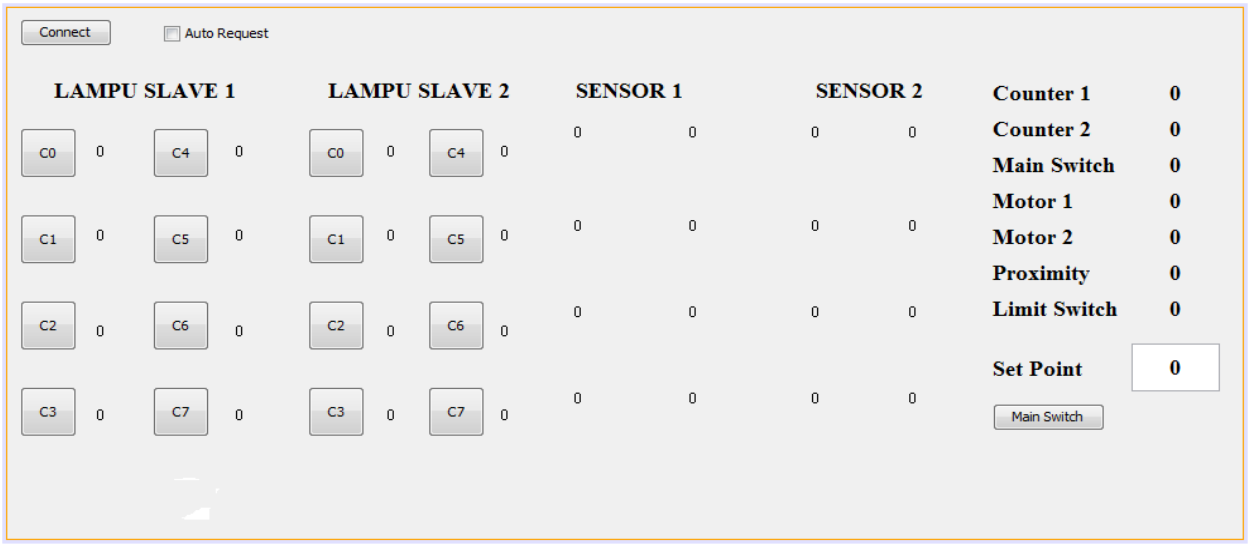
Pemrograman diagram *ladder* dibangun menggunakan *software* KGL-Win. Pengalamatan *register* yang digunakan untuk menjalankan *plant* konveyor seperti ditunjukkan Tabel 3.1. *Source code* diagram *ladder* dapat dilihat pada Lampiran 4.

Tabel 3.1 Pengalamatan *register plant* konveyor

Register	**Fungsi**
M0000	*Main switch* untuk menjalankan sistem
P0000	*Input proximity sensor*
P0001	*Input limit switch*
P0020	Menggerakkan motor 1
P0021	Menggerakkan motor 2
C000	*Counter* barang di konveyor 1
C001	*Counter* barang di konveyor 2
D000	*Setpoint* jumlah barang yang ingin dikerjakan

3.2.2 Pembuatan GUI

Proses pembuatan GUI dibangun dengan JAVA SWING berbasis pemrograman JAVA menggunakan *software* Netbeans IDE 8.1. Tujuan utama peneliti menggunakan bahasa pemrograman JAVA adalah karena fleksibilitas dari bahasa JAVA yang dapat berjalan *multiplatform* pada sistem operasi Windows dan Linux dengan *source code* yang sama. Selain itu dengan menggunakan *software* Netbeans IDE, pemrograman dapat dilakukan melalui komputer dan di-*run* langsung pada Raspberry Pi tanpa harus melakukan *copy-paste source code*.



Gambar 3.6 Tampilan GUI

Proses *bulding source code* dari komputer menjadi sebuah *file* berekstensi ".jar" menuju Raspberry Pi dilakukan dalam jaringan *Local Area Network* (LAN). Proses ini disebut dengan

Remote Java Standard Edition yang telah didukung sejak JAVA JRE dan JDK versi 8.0 hingga versi terbaru. Maka harus dipastikan pada komputer dan Raspberry Pi sudah ter-*install* JAVA versi tersebut.

Untuk mengakses segala fitur *hardware* yang ada pada Raspberry Pi melalui JAVA, dibutuhkan dua *library* khusus yang dapat diunduh secara gratis melalui internet yaitu : Pi4j dan WiringPi.

3.2.3 Protokol Komunikasi RS485

Protokol komunikasi RS485 dirancang sendiri oleh peneliti sesuai dengan kebutuhan untuk memudahkan proses *parsing* dan *splitting* data. Proses ini terbagi menjadi 2 tahap yaitu : proses *reading* dan proses *writing*.

3.2.3.1 Proses *Reading* Komunikasi RS485

Tabel 3.2 menunjukkan *frame* protokol yang dikirim dari *master* menuju *slave* untuk melakukan proses *reading*. Yang membedakan antara *slave* 1,2, dan 3 hanyalah alamat *slave* yang dituju.

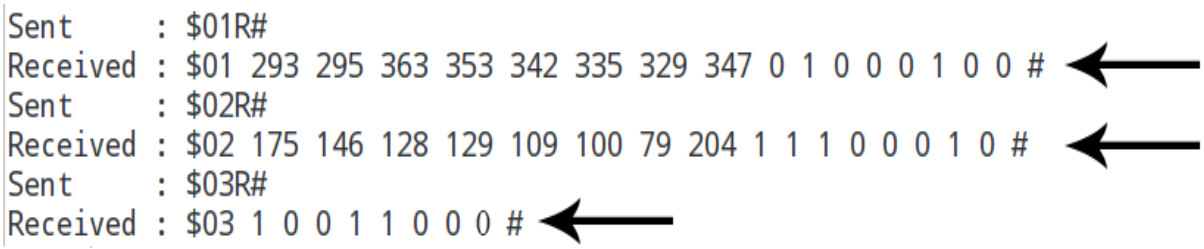
Tabel 3.2 *Frame* protokol *reading* data dari *master* menuju *slave*

Slave	Frame Protokol			
	Header (string)	Alamat (string)	Instruksi (string)	*Tail* (string)
1	$	01	R	#
2	$	02	R	#
3	$	03	R	#

Setelah *slave* menerima perintah *reading* dari *master*, maka *slave* yang sesuai dengan alamatnya akan memberikan balasan dengan *format* seperti ditunjukan Gambar 3.7

Gambar 3.7 *Frame* balasan dari *slave*

Bagian data memuat seluruh nilai suhu dan status lampu pada *slave* 1 atau 2. Pada *slave* 3 *frame* data memuat nilai seluruh *register* yang digunakan PLC untuk menjalankan *plant* konveyor. Untuk membedakan nilai – nilai yang termuat dalam *frame* data digunakan *separator* [spasi] seperti ditunjukan tanda panah pada Gambar 3.8.



Gambar 3.8 Nilai - nilai pada I/O dan *register* dipisahkan dengan *separator* [spasi]

Proses *reading* dari Raspberry Pi menuju seluruh *slave* melalui komunikasi RS485 membutuhkan penanganan khusus untuk mengatur trafik data. Hal tersebut ditujukan agar dalam satu waktu hanya terdapat satu *slave* yang membalas *request* dari *master*. Jika terdapat lebih dari satu *slave* yang membalas dalam waktu yang bersamaan, maka akan terjadi benturan data yang menyebabkan data tercampur sehingga tidak dapat diproses. Maka dari itu dibutuhkan sebuah *sampling time* untuk mengatur waktu jeda saat melakukan *request* dari satu *slave* menuju *slave* yang lain. Dalam masa jeda tersebut pula dilakukan pemrosesan data balasan dari *slave* untuk ditampilkan ke GUI.

Untuk menemukan *sampling time* terbaik dilakukan dengan pengujian terhadap beberapa waktu *sampling time* dengan mempertimbangkan hasil balasan yang didapat. Hasil balasan tersebut berupa total waktu yang dibutuhkan untuk mengakses seluruh *slave* dan tingkat keberhasilan data balasan yang masuk menuju Raspberry Pi.

3.2.3.2 Proses *Writing* Komunikasi RS485

Proses *writing* melalui komunikasi RS485 dilakukan untuk menghidupkan dan mematikan lampu di *slave* 1, menghidupkan dan mematikan lampu di *slave* 2, membaca nilai *register* pada PLC, menghidupkan dan mematikan konveyor. Untuk melakukan perintah penulisan menuju *slave* 1 dan 2 digunakan *format* protokol seperti ditunjukan Gambar 3.9. Sedangkan untuk melakukan proses penulisan menuju *slave* 3 digunakan format protokol seperti ditunjukkan Gambar 3.10.

Header ($)	Alamat (01 / 02)	Instruksi (W)	Port (B)	Bit (0 s/d 7)	Value (0/1)	Tail (#)

Gambar 3. 9 *Frame* penulisan menuju *slave* 1 dan 2

Header ($)	Alamat (03)	Instruksi (W)	Register (M / D)	Value	Tail (#)

Gambar 3.10 *Frame* penulisan menuju *slave* 3

Pada Gambar 3.10 *frame value* memiliki nilai *range* yang berbeda bergantung pada *register* yang ingin dieksekusi:

- Jika *register* yang dieksekusi adalah M, maka *value* bernilai bit 0 atau 1.
- Jika *register* yang dieksekusi adalah D, maka *value* bernilai heksadesimal 0000 s/d FFFF.

3.2.4 Protokol Komunikasi RS232C

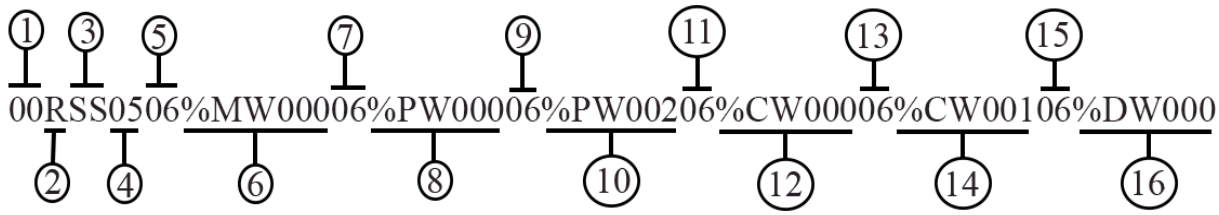
Protokol komunikasi RS232C digunakan sesuai dengan ketentuan yang telah dituliskan pada *datasheet* LG Master K200S. Proses ini terdiri dari dua tahap yaitu : Proses *reading* dan proses *writing*.

3.2.4.1 Proses *Reading* Komunikasi RS232C

Berdasarkan diagram *ladder* yang telah dibuat, *register* PLC yang akan diakses dalam proses *reading* ini meliputi : M000, P000, P002, C000, C001. Untuk melakukan hal tersebut, perintah yang akan dikirimkan menuju PLC dapat dilakukan hanya dengan satu baris *frame* seperti ditunjukkan Gambar 3.11. Data dibuat dengan tipe data string, sedangkan *header* bernilai 5 heksadesimal dan *tail* bernilai 4 heksadesimal merupakan suatu ketetapan.

Gambar 3.11 *Frame* untuk membaca 6 *register* PLC

Segmentasi *frame* data dari gambar diatas dapat dilihat pada Gambar 3.12.



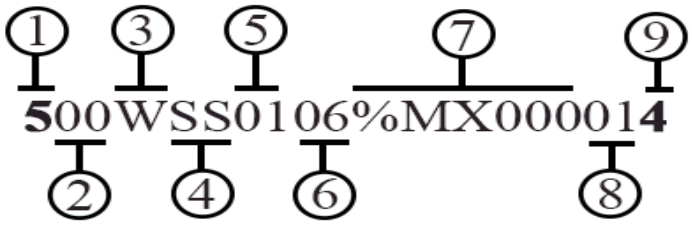
Gambar 3.12 Pembagian frame pada bagian data

Keterangan :

1. *Station number* : *Station number* PLC
2. *Main instruction* : Perintah *reading*
3. *Instruction type* : Membaca *single* bits
4. *Number of block* : *Register* yang ingin dibaca sebanyak 6 buah *register*
5. *Length of device definition* : Jumlah karakter pada “%MW000” ada 6 karakter
6. *Device definition* : M000 dengan tipe data word
7. *Length of device definition* : Jumlah karakter pada “%PW000” ada 6 karakter
8. *Device definition* : P000 dengan tipe data word
9. *Length of device definition* : Jumlah karakter pada “%PW002” ada 6 karakter
10. *Device definition* : P002 dengan tipe data word
11. *Length of device definition* : Jumlah karakter pada “%CW000” ada 6 karakter
12. *Device definition* : C000 dengan tipe data word
13. *Length of device definition* : Jumlah karakter pada “%CW001” ada 6 karakter
14. *Device definition* : C000 dengan tipe data word
15. *Length of device definition* : Jumlah karakter pada “%DW000” ada 6 karakter
16. *Device definition* : D000 dengan tipe data word.

3.2.4.2 Proses *Writing* Komunikasi RS232C

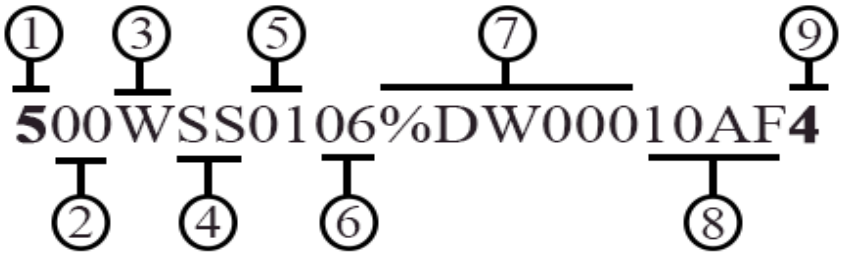
Berdasarkan diagram *ladder* yang telah dibuat, register PLC yang akan diakses dalam proses *writing* meliputi : M000 bit 0 (M0000) dan *register* D000. Gambar 3.13 menunjukkan cara memberikan nilai 1 menuju *register* M000 dan Gambar 3.14 menunjukkan cara memberikan nilai 10AF menuju *register* D000.



Gambar 3.13 Perintah untuk memberikan nilai ke *register* M000

Keterangan :

1. *Heade*r : 5 heksadesimal
2. *Station number* : *Station number* PLC
3. *Main instruction* : Perintah *writing*
4. *Instruction type* : Menulis *single* bits
5. *Number of block* : *Register* yang ingin ditulis sebanyak 1 buah *register*
6. *Length of device definition* : Jumlah karakter pada "%MX000" ada 6 karakter
7. *Device definition* : M000 dengan tipe data bit
8. Data : Nilai bit yang ingin diberikan dalam bentuk 2 byte
9. *Tail* : 4 heksadesimal



Gambar 3.14 Perintah untuk memberikan nilai ke *register* D000

Keterangan :

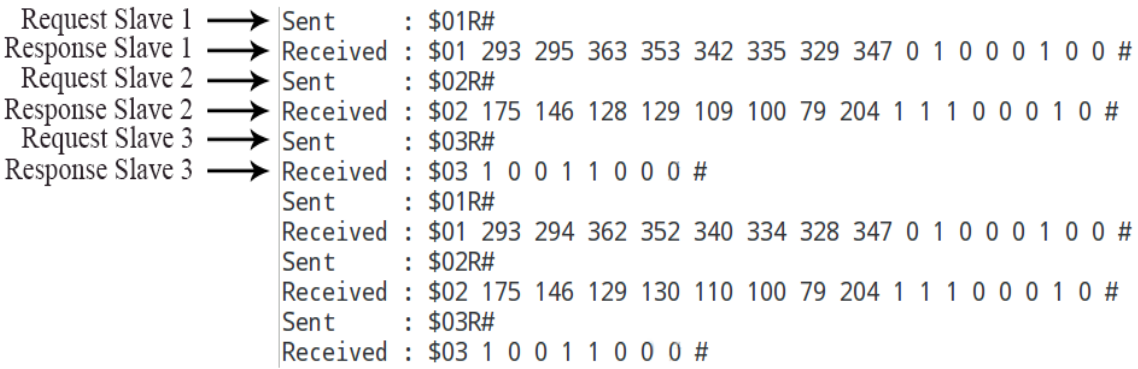
1. *Header* : 5 heksadesimal
2. *Station number* : *Station number* PLC
3. *Main instruction* : Perintah *writing*
4. *Instruction type* : Menulis *single* bits

5. *Number of block* : *Register* yang ingin ditulis sebanyak 1 buah *register*
6. *Length of device definition* : Jumlah karakter pada “%DX000” ada 6 karakter
7. *Device definition* : D000 dengan tipe data word
8. Data : Nilai bit yang ingin diberikan dalam bentuk 4 byte
9. *Tail* : 4 heksadesimal

BAB 4
HASIL DAN PEMBAHASAN

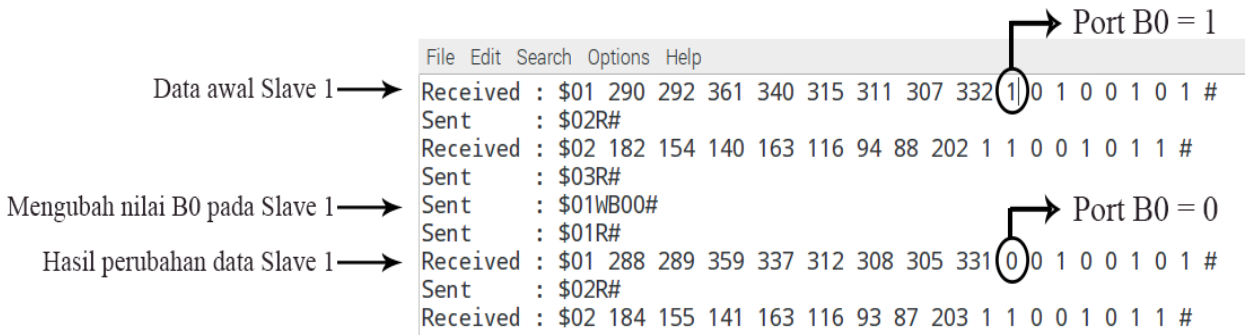
4.1 Pengujian dan Analisa Komunikasi Raspberry Pi dengan Seluruh Slave

Dari hasil pengujian yang dilakukan terhadap komunikasi RS485 antara *master* dengan seluruh *slave*, format data protokol RS485 yang dirancang dapat bekerja seperti yang diinginkan. Menggunakan frekuensi *clock* UART *default* Raspberry Pi 3 sebesar 48 Mhz dan baudrate 9600 yang berarti mampu mengirimkan 9600 bits dalam waktu 1 detik. Semua proses *writing/reading* data untuk *slave* tertentu juga berhasil dieksekusi sesuai dengan alamat *slave* yang dituju. Gambar 4.1 menunjukkan proses *reading* data dari seluruh *slave* bekerja dengan semestinya.



Gambar 4.1 Komunikasi RS485 berjalan dengan semestinya

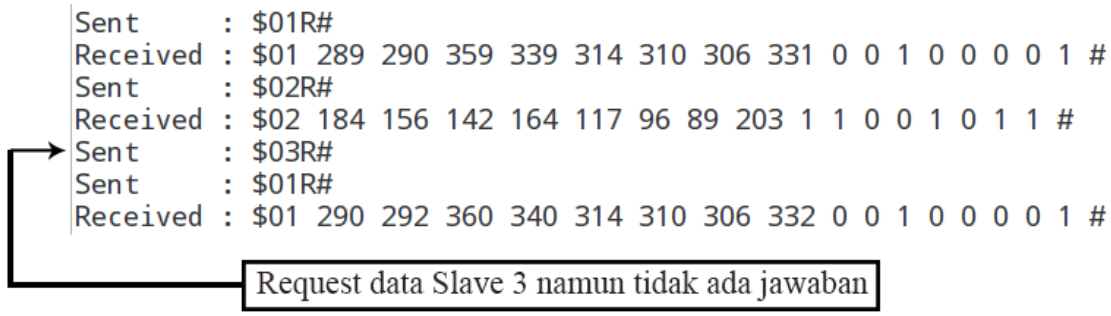
Gambar 4.2 menunjukkan proses untuk mengubah nilai *port* B0 pada *slave* 1 yang pada awalnya bernilai 1 dan diubah menjadi 0.



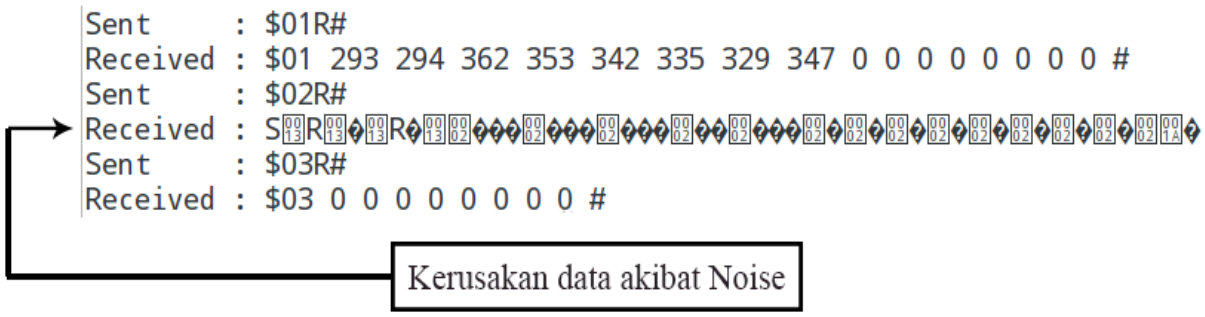
Gambar 4.2 Proses *writing* port B0 slave 1

Terdapat kelemahan pada sistem ini yaitu selama proses pengujian komunikasi RS485, seringkali ditemukan kerusakan data saat *slave* membalas perintah dari *master* dan begitu pula sebaliknya. Salah satu penyebab kerusakan data adalah pengaruh dari *noise* yang disebabkan oleh kualitas komponen, tingkat kerapian dalam proses penyolderan, dan tingkat kerapian *wiring*. Pengaruh dari *noise* menyebabkan format protokol data tidak lagi sesuai dengan format protokol yang telah ditetapkan. Saat hal tersebut terjadi maka data yang diterima tidak akan diproses, sehingga tampilan pada SCADA tidak akan di-*update* (menggunakan data yang sebelumnya).

Gambar 4.3 menunjukkan *slave* tidak merespon *request* dari *master* akibat kerusakan data yang terjadi saat pengiriman dari *master* menuju *slave*. Sedangkan Gambar 4.4 menunjukkan kerusakan data akibat *noise* yang terjadi saat pengiriman data dari *slave* menuju *master*.



Gambar 4.3 Pengiriman data dari *master* tidak mendapat tanggapan dari *slave* 3



Gambar 4.4 Data yang diterima dari *slave* 2 mengalami kerusakan akibat *noise*

Faktor lain yang menyebabkan kerusakan data dalam sistem ini adalah kecepatan *sampling* data dan *timing* pengubahan logika untuk melakukan kegiatan *read/write* pada *pin* $\overline{RE}/DE$ IC MAX485.

Tabel 4.1 Kerusakan data pada komunikasi RS485

Jumlah *Request* Data	***Sampling Time* (*ms*)**	**Jumlah *error* pada percobaan**			**Rata - rata**	**% Rata - rata**
		1	**2**	**3**		
300	100	140	168	152	152	50,66 %
	200	98	98	118	98	32,66 %
	300	95	83	76	83	27,66 %
	333	82	61	68	68	22,66 %
	350	59	73	66	66	22 %
	400	55	70	67	67	22,33 %

Pada Tabel 4.1 memperlihatkan kerusakan data yang terjadi dalam komunikasi RS485 yang disebabkan oleh *noise*. Berdasarkan data tersebut, kecepatan *sampling* yang digunakan dalam sistem ini adalah sebesar $333\ ms$ untuk setiap *slave*. Hal tersebut dikarenakan persentase rata – rata antara $333\ ms$, $350\ ms$, dan $400\ ms$ tidak berbeda jauh. Selain itu dengan menggunakan $333\ ms$, total waktu yang digunakan untuk mengakses seluruh *slave* kurang dari 1 detik atau hanya sebesar $999\ ms$.

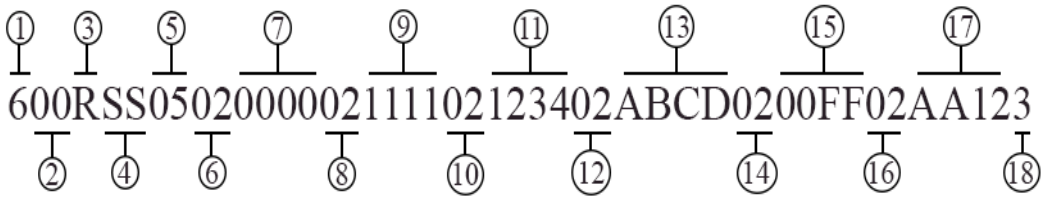
Pin $\overline{RE}/DE$ pada IC MAX485 berfungsi untuk menentukan instruksi yang ingin dilakukan. Jika *pin* $\overline{RE}/DE$ berlogika 0 maka proses yang akan dilakukan adalah proses *reading*. Sedangkan jika *pin* $\overline{RE}/DE$ berlogika 1 maka proses yang dilakukan adalah proses *writing*. Saat melakukan proses *writing* dibutuhkan waktu jeda antara mengubah *pin* $\overline{RE}/DE$ menjadi 1, mengirim paket data, dan mengembalikan *pin* $\overline{RE}/DE$ menjadi 0. Selama proses pengujian didapatkan waktu jeda terbaik untuk meminimalisir kerusakan data adalah sebesar $10\ ms$.

4.2 Pengujian dan Analisa Konverter RS485 – RS232C (Slave 3)

Pada pengujian komunikasi antara *slave* 3 dengan PLC LG Master K200S, format protokol komunikasi RS232C yang digunakan harus sesuai dengan format protokol yang telah ditetapkan oleh *vendor* pembuat. Dari hasil pengujian pada *plant* konveyor format protokol yang digunakan untuk proses *reading/writing* sesuai dengan apa yang telah dituliskan di *datasheet* LG Master K200S.

Untuk melakukan eksekusi *reading*, hanya membutuhkan satu baris *frame* untuk mengakses 5 buah *register* seperti yang telah ditunjukkan Gambar 3.12. Setelah instruksi

tersebut dikirimkan ke PLC, maka PLC akan membalas dengan format data seperti ditunjukkan oleh Gambar 4.6.



Gambar 4.5 *Frame* protokol balasan PLC

Keterangan :

1. *Header* : 6 heksadesimal
2. *Station number* : *Station number* PLC
3. *Main instruction* : Perintah *reading*
4. *Instruction type* : Membaca *single* bits
5. *Number of block* : *Register* yang ingin dibaca sebanyak 5 buah *register*
6. *Length of data* : Jumlah word pada “0000” terdiri dari 2 word
7. *Data* : Nilai *register* M000 dengan tipe data word
8. *Length of data* : Jumlah word pada “1111” terdiri dari 2 word
9. *Data* : Nilai *register* P000 dengan tipe data word
10. *Length of data* : Jumlah word pada “1234” terdiri dari 2 word
11. *Data* : Nilai *register* P002 dengan tipe data word
12. *Length of data* : Jumlah word pada “ABCD” terdiri dari 2 word
13. *Data* : Nilai *register* C000 dengan tipe data word
14. *Length of data* : Jumlah word pada “00FF” terdiri dari 2 word
15. *Data* : Nilai *register* C001 dengan tipe data word
16. *Length of data* : Jumlah word pada “AA12” terdiri dari 2 word
17. *Data* : Nilai *register* D000 dengan tipe data word
18. *Tail* : 3 heksadesimal

Nilai *header* dan *tail* pada *frame* balasan PLC berbeda dari *header* dan *tail* saat melakukan *request*. Dimana saat melakukan *request*, *header* bernilai 5 heksadesimal dan *tail* bernilai 4 heksadesimal. Sedangkan pada *frame* balasan, *header* bernilai 6 heksadesimal dan *tail* bernilai 3 heksadesimal.

Karena *slave* 3 menangani 2 jenis komunikasi yaitu RS485 dengan *master* dan RS232C dengan PLC, maka *sampling time* sangat berpengaruh agar interupsi tidak saling mengganggu satu sama lain. Dari hasil pengujian yang telah dilakukan dengan mengubah - ubah nilai *sampling time*, didapatkan *sampling time* yang terbaik pada komunikasi RS232C adalah sebesar 100 ms. Pengaruh dari *sampling time* RS232C yang terlalu cepat akan menyebabkan penerimaan data pada komunikasi RS485 terganggu sehingga *slave* 3 tidak merespon *request* dari *master*. Sedangkan *sampling time* RS232C yang terlalu lambat akan menyebabkan pembacaan *register* PLC menjadi lambat atau tidak *real time*.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

1. Format protokol komunikasi RS485 untuk menangani komunikasi lebih dari 2 perangkat dirancang dengan format *header*, alamat perangkat, data, dan *tail*.
2. Format protokol komunikasi RS232C LG Master K200S sesuai dengan format yang telah dituliskan di *datasheet*.
3. *Sampling time* terbaik yang dibutuhkan Raspberry Pi untuk mengakses seluruh *slave* adalah sebesar 333 ms per *slave*.
4. Kerusakan data komunikasi RS485 disebabkan oleh *noise* dan *sampling time* yang terlalu cepat.
5. Pengubahan logika pada *pin* $\overline{RE}/DE$ yang terlalu cepat dapat menyebabkan kerusakan data.
6. Waktu jeda terbaik dalam pengubahan logika pada *pin* $\overline{RE}/DE$ dalah sebesar 10 ms

5.2 Saran

1. Perlu ditambahkan sensor sebagai umpan balik untuk membaca status lampu.
2. Perlu dilakukan penelitian untuk Membangun modul I/O *Expansion* dengan komunikasi RS232C untuk PLC LG Master K *series*.
3. Perlu dilakukan penelitian untuk membangun sistem SCADA dengan komunikasi nirkabel (*wireless*) menggunakan PLC LG Master K *series*.
4. Perlu dilakukan penelitian untuk membangun sistem SCADA dengan menggunakan *smartphone*.

DAFTAR PUSTAKA

[1] K.Papasideris, C.Landry, B.Sutter, dan A.Wilson, "Environtment Temperature Control Using Modbus and RS485 Communication Standards," Texas A&M University, Jan. 2010.

[2] Z.Xu, F.Pu, X.Fang, dan J.Fu. "Journal of Sensors." *Raspberry Pi Based Intelligent Wireless Sensor Node for Localized Torrential Rain Monitoring*, vol. 2016, pp. 1-12, 2016.

[3] J.H.Huh, N.Kim, dan K.Seo. "Asia-Pasific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology." *Implementation of PLC-based Foundation Technology of Electric Power Control/Monitoring for Micro grid*, vol. 5, pp 19-27, Dec. 2015.

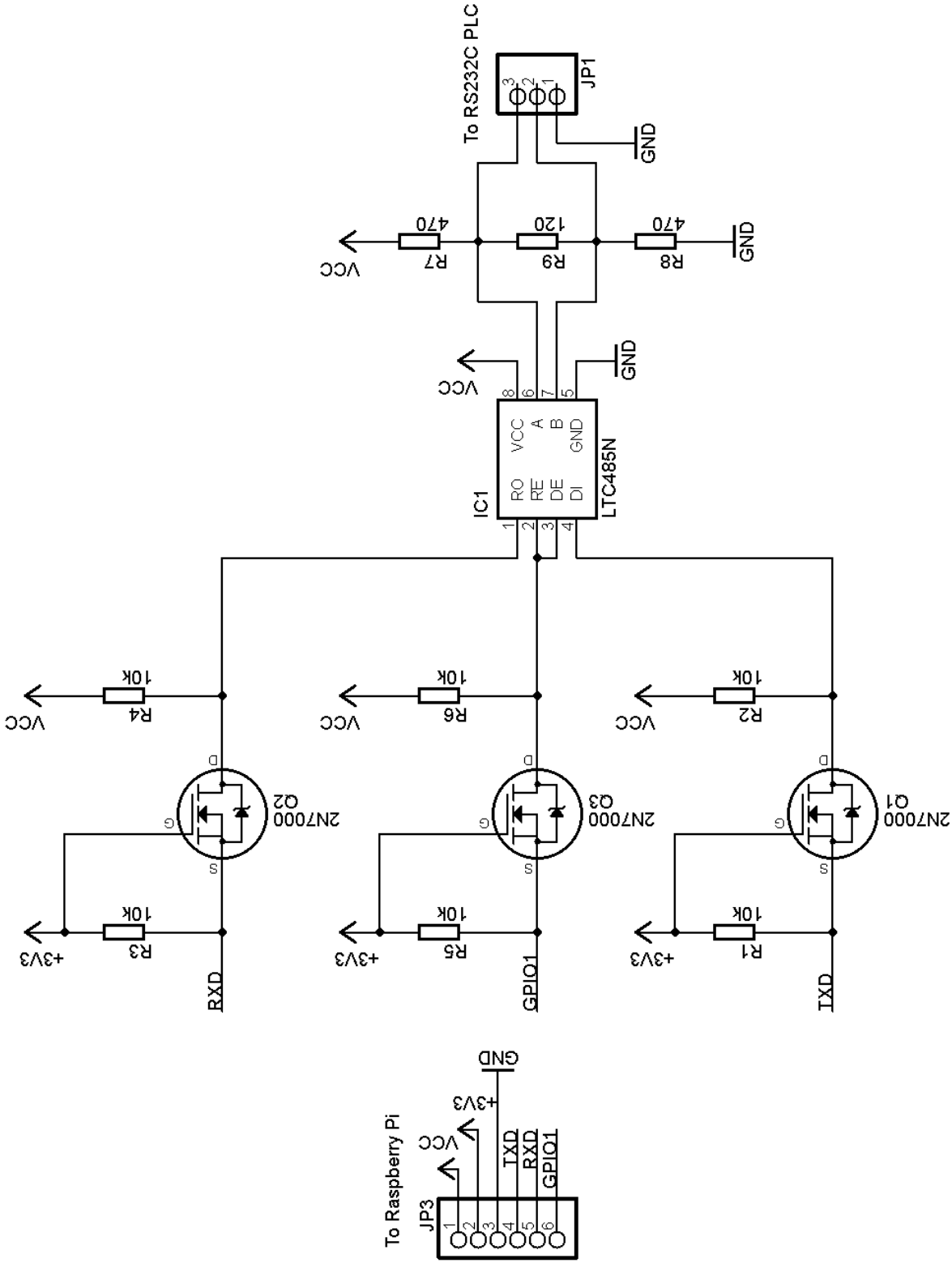
[4] Raspberrypi.org. (2018). *The Raspberry Pi UARTs – Raspbery Pi Documentation*. [online] Available at: https://www.raspberrypi.org/documentation/configuration/uart.md [Accessed 12 May 2018].

[5] BB SmartWorx, "A Practical Guide to Using RS-422 and RS-485 Serial Interfaces," *RS-422 and RS-485 Applications Ebook*, Oct. 2010. [Online] Available : http://www.bb-elec.com/Learning-Center/All-White-Papers/Serial/RS-422-and-RS-485-Applications-eBook.aspx

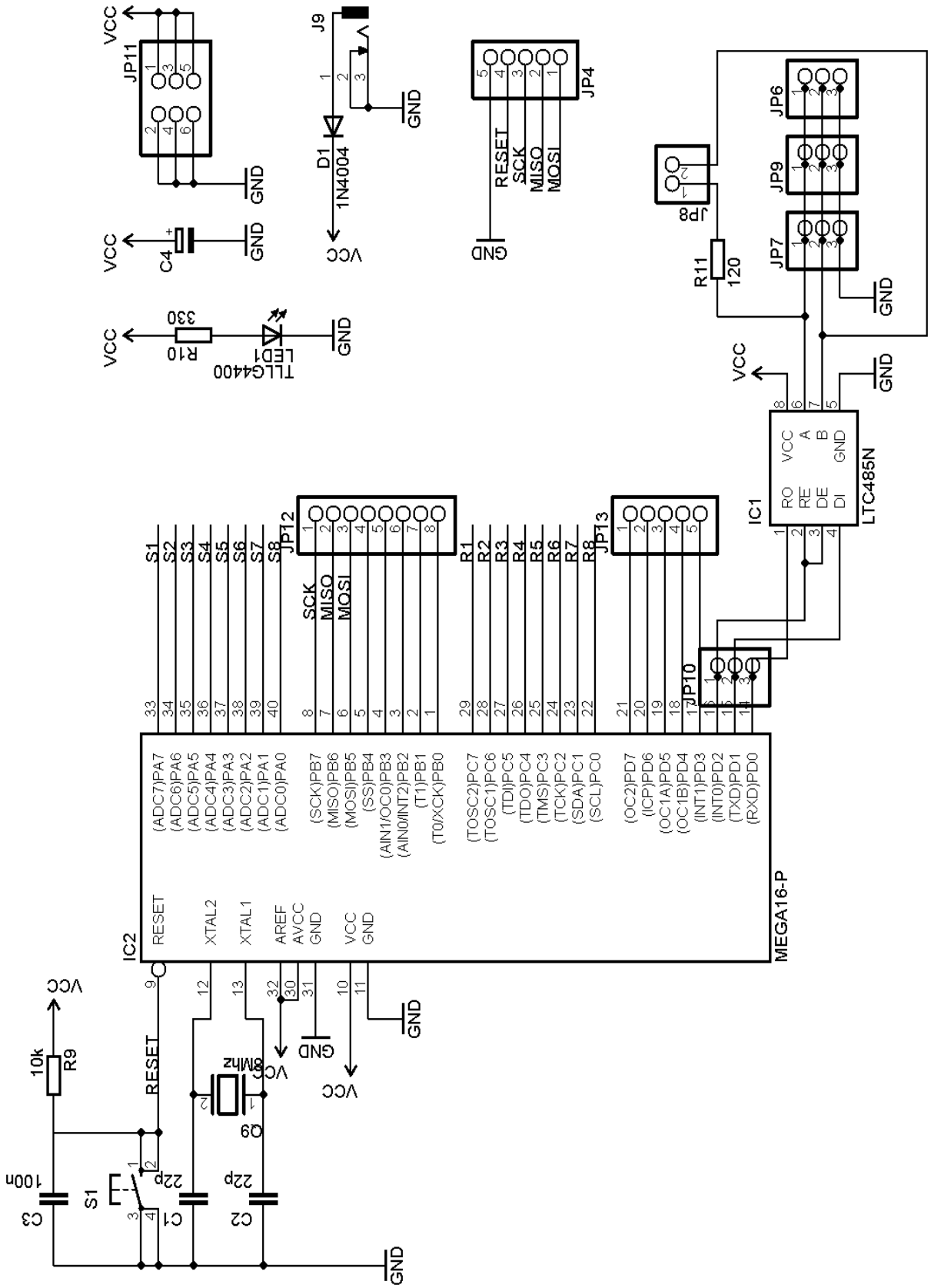
[6] LG Industrial Systems, "User's Manual LG Programmable Logic Controller Master-K," Datasheet, Accessed on Feb. 08, 2018. [Online] Available : http://www.ehaegypt.com/uploads/K200_300_1000S_yi2jwqxz.pdf

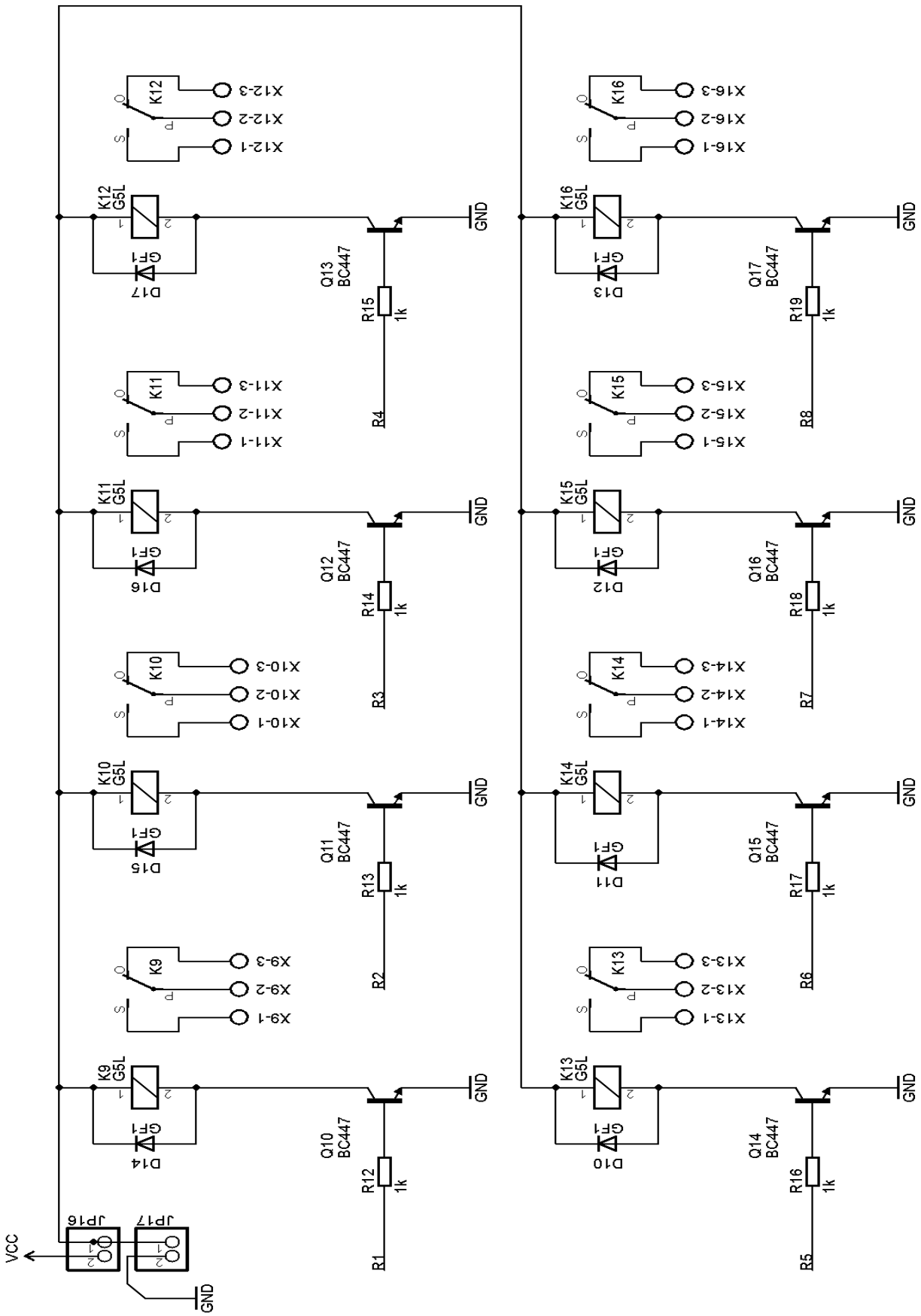
LAMPIRAN

Diagram Rangkaian Level Shifter dan RS485 Converter



Rangkaian Slave 1 & 2




VCC
GND
JP16
JP17
K9
G5L
K10
G5L
K11
G5L
K12
G5L
K13
G5L
K14
G5L
K15
G5L
K16
G5L
D10
D11
D12
D13
D14
D15
D16
D17
GF1
Q10
BC447
Q11
BC447
Q12
BC447
Q13
BC447
Q14
BC447
Q15
BC447
Q16
BC447
Q17
BC447
R1
1k
R2
1k
R3
1k
R4
1k
R5
1k
R6
1k
R7
1k
R8
1k
R12
1k
R13
1k
R14
1k
R15
1k
R16
1k
R17
1k
R18
1k
R19
1k
X9-1
X9-2
X9-3
X10-1
X10-2
X10-3
X11-1
X11-2
X11-3
X12-1
X12-2
X12-3
X13-1
X13-2
X13-3
X14-1
X14-2
X14-3
X15-1
X15-2
X15-3
X16-1
X16-2
X16-3

Rangkaian Slave 3

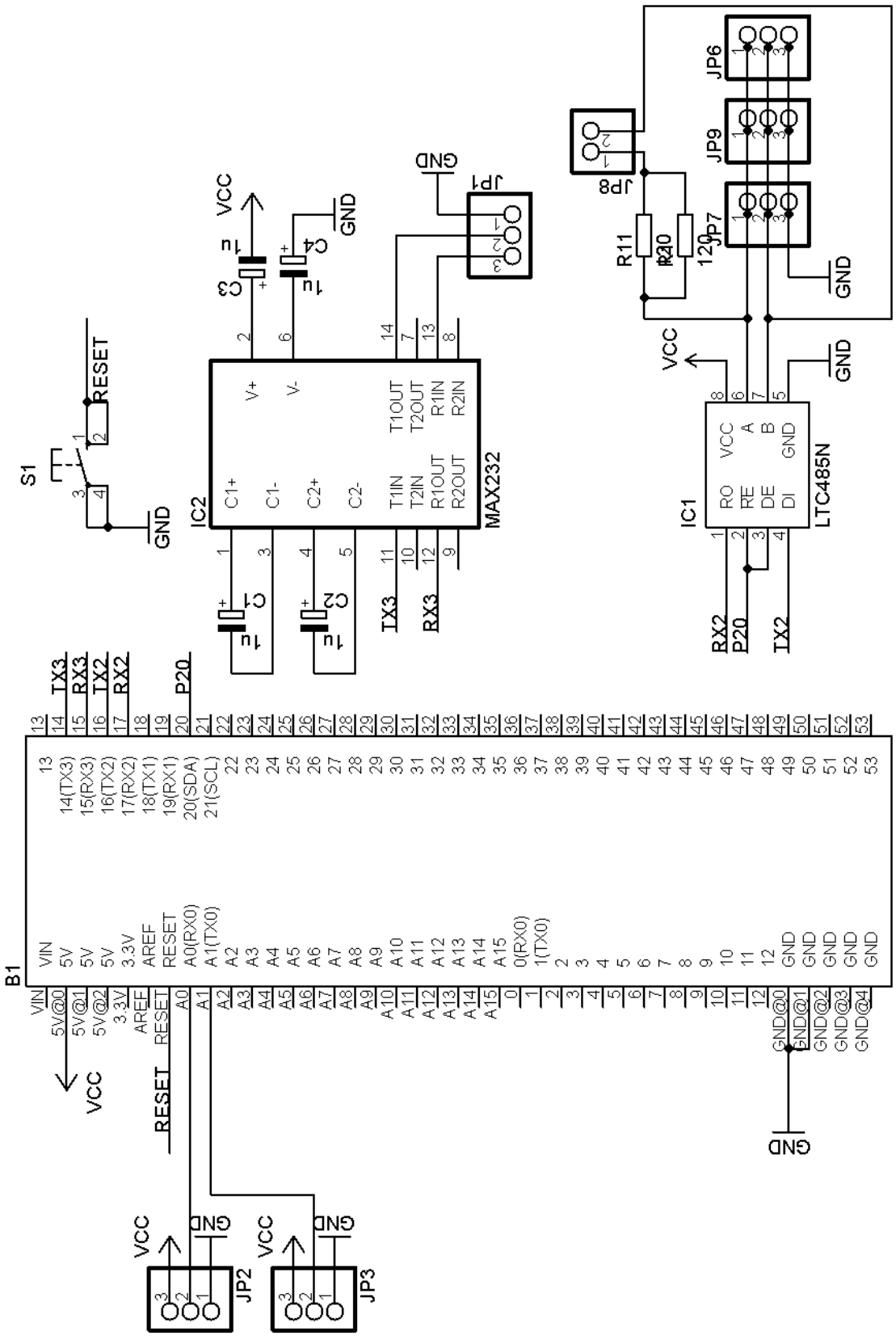
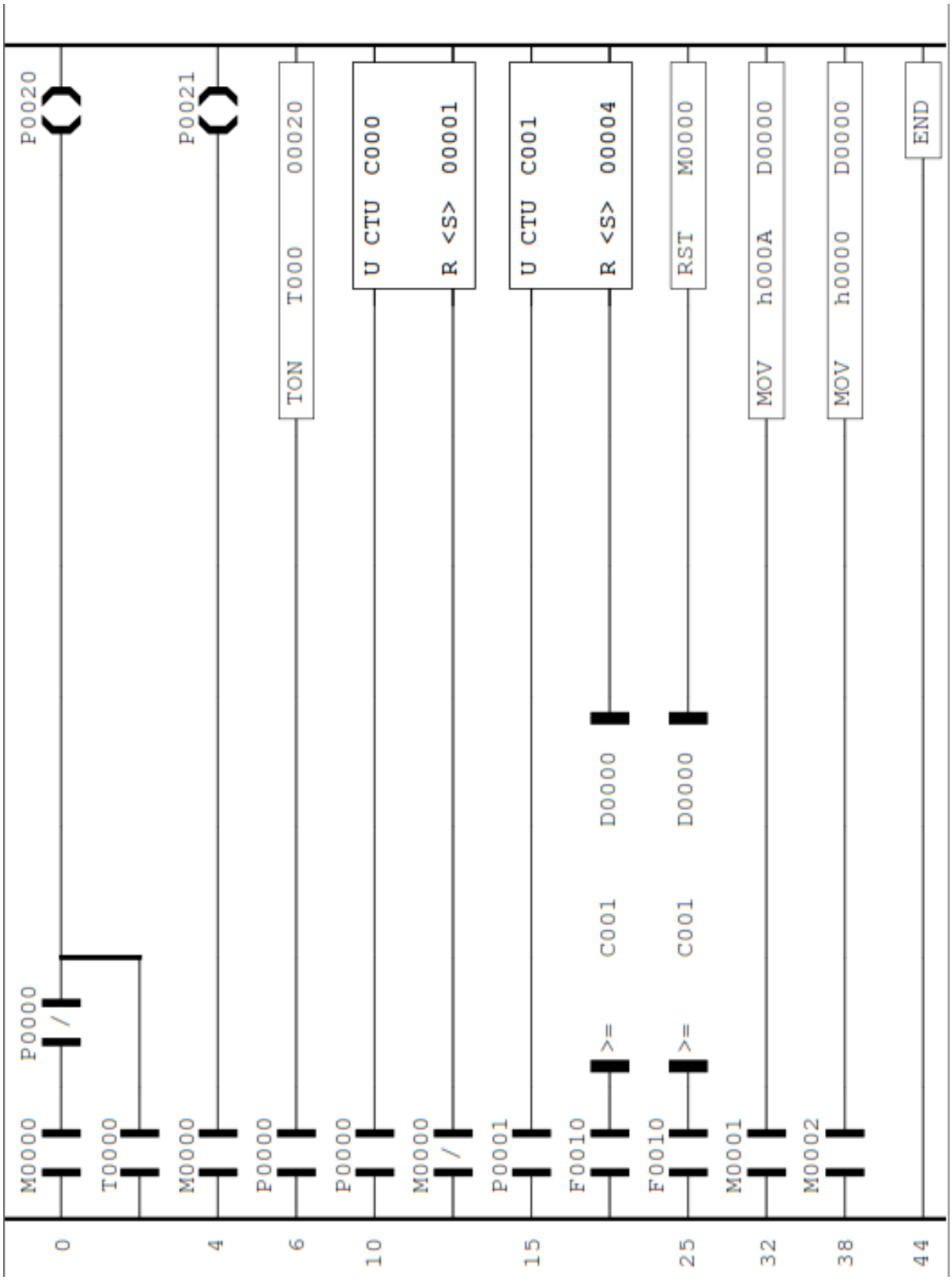


Diagram Ladder


P0020
P0021
M0000
P0000
T0000
M0000
P0000
TON T000 00020
P0000
U CTU C000
R <S> 00001
M0000
P0001
U CTU C001
R <S> 00004
F0010
>= C001 D0000
F0010
>= C001 D0000
RST M0000
M0001
MOV h000A D0000
M0002
MOV h0000 D0000
END
0
4
6
10
15
25
32
38
44