RANCANG BANGUN *REMOTE CONTROL* UNIVERSAL SEBAGAI PENGATUR *ON/OFF* PERALATAN LISTRIK

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
NUR ANNISA MUFIDAH
NIM. 0610630075-63

KEMENTERIAN PENDIDIKAN NASIONAL
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2010

LEMBAR PERSETUJUAN

RANCANG BANGUN *REMOTE CONTROL* UNIVERSAL SEBAGAI PENGATUR *ON/OFF* PERALATAN LISTRIK

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
NUR ANNISA MUFIDAH
NIM. 0610630075-63

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Ponco Siwindarto, MS.	**Mochammad Rif'an, ST., MT.**
NIP. 19590304 198903 1 001	**NIP. 19710301 200012 1 001**

LEMBAR PENGESAHAN

RANCANG BANGUN *REMOTE* KONTROL UNIVERSAL SEBAGAI PENGATUR *ON/OFF* PERALATAN LISTRIK

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

NUR ANNISA MUFIDAH
NIM. 0610630075-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 3 Agustus 2010

DOSEN PENGUJI

Panca Mudjirahardjo, ST., MT.
NIP. 19700329 200012 1 001

Ir. Nurussa'adah, MT.
NIP. 19680706 199203 2 001

Dr. Agung Darmawansyah, ST., MT.
NIP. 19721218 199903 1 002

Mengetahui
Ketua Jurusan Teknik Elektro

Rudy Yuwono ST., M.Sc.
NIP. 19710615 199802 1 003

Pengantar

Puji dan syukur penulis panjatkan kepada Allah SWT, karena dengan rahmat, taufik dan hidayah-Nya lah skripsi ini dapat diselesaikan.

Skripsi berjudul "Rancang Bangun *Remote Control* Universal sebagai Pengatur *On/Off* Peralatan Listrik" ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

- Ibu, Bapak dan seluruh keluarga besar atas segala nasehat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Bapak Rudy Yuwono, ST., M.Sc selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya dan Bapak M. Aziz Muslim, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Ir. M. Julius St, MS selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Ir. Ponco Siwindarto, MS. dan Bapak Mochammad Rif'an, ST., MT. selaku Dosen Pembimbing 1 dan Dosen Pembimbing 2 atas segala bimbingan, nasehat, pengarahan, motivasi, saran dan masukan yang telah diberikan,
- Agung Setiabudi, ST. atas bantuan dan saran yang telah diberikan,
- Aditya Ariyandhi, ST. dan semua sahabatku untuk perhatian dan dukungannya selama ini,
- Teman-teman Elkamania, Ge Force, teman-teman di kelembagaan, senior serta semua pihak yang tidak mungkin bagi penulis untuk mencantumkan namanya satu-persatu, terima kasih banyak atas bantuan dan dukungannya.

Pada akhirnya, penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun. Penulis berharap semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi.

Malang, Agustus 2010

Penulis

Abstrak

Nur Annisa Mufidah, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Juni 2010, *Rancang Bangun Remote Control Universal sebagai Pengatur On/Off Peralatan Listrik*, Dosen Pembimbing: Ir. Ponco Siwindarto, MS. dan Mochammad Rif'an, ST., MT.

Peralatan elektronik seperti televisi, *tape recorder*, *air conditioner* (AC), lampu penerangan dan yang lainnya telah banyak digunakan. Namun saat ini banyak penggunaan peralatan listrik yang kurang efisien, misalnya peralatan listrik tetap menyala saat tidak dibutuhkan. Hal tersebut menyebabkan bertambahnya biaya untuk listrik yang harus dikeluarkan. Ada berbagai alasan mengapa pengguna membiarkan peralatan listrik tetap menyala saat tidak dibutuhkan, misalnya karena letak saklar yang jauh dari jangkauannya atau karena pemakainya lupa atau tertidur. Oleh karena itu perlu dibuat sebuah sistem yang dapat mengatur *on/off* peralatan listrik dari semua tempat dengan jangkauan tertentu dan bisa bersifat otomatis.

Maka dalam skripai ini akan dirancang *Remote Control* Universal sebagai Pengatur *On/Off* Peralatan Listrik. Sistem ini terdiri atas *remote control*, unit pengolah data dan rangkaian *driver*. *Remote* control berfungsi mengirimkan sinyal informasi, unit pengolah data yang akan mengolah informasi dari *remote* dan rangkaian driver yang melakukan pensaklaran peralatan listrik.

Remote control dapat mengirimkan data ke unit pengolah data sesuai dengan tombol *keypad* yang ditekan dengan kecepatan 9600 bps dan jarak transmisi maksimum dari unit pengolah data di dalam ruangan adalah 20 meter. Unit pengolah data menerima data yang dikirimkan *remote control*, menyeleksi data apakah sesuai dengan format paket data yang dirancang dan memberikan respon dengan memberikan tegangan *trigger* pada rangkaian *driver relay*. *Driver relay* kemudian melakukan pensaklaran terhadap peralatan listrik. Peralatan listrik yang diatur dalam sistem ini dibatasi sebanyak 16 buah.

Kata Kunci: Remote control, modul RF, peralatan listrik

Daftar Isi

Pengantar i
Abstrak ii
Daftar Isi iii
Daftar Gambar v
Daftar Tabel vii
Daftar Tabel vii
BAB I 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 3
1.5 Sistematika Penulisan 3
BAB II 4
2.1 Transmisi Data 4
2.1.1 Bagian Pemancar (TLP434A) 5
2.1.2 Bagian Penerima (RLP 434A) 6
2.1.3 Modulasi Digital ASK 7
2.1.4 Komunikasi Serial 8
2.2 Mikrokontroler ATMega8 8
2.2.1 Struktur dan Operasi Port 9
2.2.2 Komunikasi Serial 11
2.3 LCD LMB162ADC 12
2.4 Keypad Matrik 4x4 14
2.5 RTC DS1307 15
2.5.1 Komunikasi I2C 16
2.6 Transistor 9014 18
2.6 *Relay* LY2N 19
BAB III 21
3.1 Perancangan dan Pembuatan Alat 21
3.2 Pengujian Alat 21
BAB IV 22

4.1 Penentuan Spesifikasi Alat 22
4.2 Diagram Blok Sistem 22
4.3 Perancangan Perangkat Keras (Hardware) 24
4.3.1 Perancangan Perangkat Keras Remote *Control* 24
4.3.2 Perancangan Perangkat Keras Unit Pengolah Data 27
4.3.3 Perancangan Rangkaian *Driver Relay* 32
4.4 Perancangan Format Paket Data 33
4.5 Perancangan Perangkat Lunak (*Software*) 34
4.5.1 Perancangan Perangkat Lunak pada *Remote* 34
4.5.2 Perancangan Perangkat Lunak pada Unit Pengolah Data 36
BAB V 38
5.1 Pengujian Tampilan LCD 38
5.2 Pengujian Antarmuka *Keypad* dengan Mikrokontroler 39
5.3 Pengujian *Remote Control* 39
5.4 Pengujian Rangkaian Pemancar TLP 434A dan Penerima RLP 434A 45
5.4.1 Pengujian bentuk sinyal dan frekuensi modul RF 46
5.4.2 Pengujian kecepatan transfer data (*baud rate*) modul RF 48
5.4.3 Pengujian komunikasi data dan jarak jangkau transmisi modul RF 49
5.5 Pengujian Sistem Pewaktu RTC DS1307 51
5.6 Pengujian Rangkaian *Driver Relay* 52
5.7 Pengujian Keseluruhan Sistem 53
BAB VI 60
6.1 Kesimpulan 60
6.2 Saran 61
LAMPIRAN I 63

Daftar Gambar

Gambar 2.1 Bentuk TLP 434A 5
Gambar 2.2 Konfigurasi pin modul RF RLP 434A 6
Gambar 2.3 Sinyal termodulasi ASK 7
Gambar 2.4 Format frame data serial USART 8
Gambar 2.5 Konfigurasi pin mikrokontroler ATmega8 10
Gambar 2.6 Format frame data serial USART 12
Gambar 2.7 Blok diagram LCD LMB162ADC 13
Gambar 2.8 Rangkaian keypad matrik 4x4 14
Gambar 2.9 Susunan pin DS1307 15
Gambar 2.10 Diagram blok DS1307 15
Gambar 2.11 Susunan perangkat dengan jalur komunikasi I2C 16
Gambar 2.12 Representasi sinyal komunikasi I2C 17
Gambar 2.13 Urutan data dalam komunikasi I2C 17
Gambar 2.14 Konfigurasi pin transistor 9014 18
Gambar 2.15 Karakteristik transistor 9014 19
Gambar 2.16 Susunan terminal relay LY2N 20
Gambar 4.1 Diagram blok sistem secara keseluruhan 23
Gambar 4.2 Diagram blok *remote control* 24
Gambar 4.3 Rangkaian keypad 4x4 25
Gambar 4.4 Rangkaian pengendali utama *remote control* 26
Gambar 4.5 Rangkaian LCD 27
Gambar 4.6 Antarmuka modul RF TLP434A dengan mikrokontroler 27
Gambar 4.7 Diagram blok unit pengolah data 28
Gambar 4.8 Rangkaian pengendali utama unit pengolah data 29
Gambar 4.9 Rangkaian sistem pewaktu 29
Gambar 4.10 Analisis resistor *pull up* output RTC 30
Gambar 4.11 Antarmuka modul RLP434A dengan mikrokontroler ATMega 8 32
Gambar 4.12 Rangkaian *driver relay* 33
Gambar 4.13 Diagram alir rutin pada *remote control* 35
Gambar 4.14 Diagram alir rutin pada unit pengolah data 36
Gambar 5.1 Diagram blok pengujian tampilan LCD 38

Gambar 5.2 Tampilan hasil pengujian tampilan LCD 38
Gambar 5.3 Diagram blok pengujian antarmuka *keypad* dengan LCD 39
Gambar 5.4 Tampilan hasil pengujian antarmuka *keypad* dengan mikrokontroler 39
Gambar 5.5 Diagram blok pengujian *remote control* 40
Gambar 5.6 Tampilan hasil pengujian *remote control* mode otomatis 40
Gambar 5.7 Tampilan hasil pengujian *remote control* mode mati 41
Gambar 5.8 Tampilan hasil pengujian *remote control* mode hidup.....42
Gambar 5.9 Tampilan hasil pengujian *remote control* mode hidup 30 menit 43
Gambar 5.10 Tampilan hasil pengujian *remote control* mode hidup 1 jam 44
Gambar 5.11 Tampilan hasil pengujian *remote control* mode hidup 2 jam 45
Gambar 5.12 Diagran blok pengujian bentuk sinyal dan frekuensi modul pemancar TLP 434A dan penerima RLP 434A 46
Gambar 5.13 Tampilan osiloskop hasil pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A untuk frekuensi 100 Hz.....47
Gambar 5.14 Tampilan osiloskop hasil pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A untuk frekuensi 27 kHz.....47
Gambar 5.15 Diagram blok pengujian kecepatan transfer data (*baud rate*) pemancar TLP 434A dan penerima RLP 434A 48
Gambar 5.16 Tampilan hasil pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A dengan *baud rate* 4800 bps..... 48
Gambar 5.17 Tampilan hasil pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A dengan *baud rate* 38400 bps 49
Gambar 5.18 Tampilan hasil pengujian transmisi data rangkaian pemancar TLP 434A dan penerima RLP 434A dengan jarak 20 m 50
Gambar 5.19 Tampilan hasil pengujian transmisi data rangkaian pemancar TLP 434A dan penerima RLP 434A dengan jarak lebih dari 20 m 51
Gambar 5.20 Diagram blok pengujian sistem pewaktu RTC DS1307 51
Gambar 5.21 Tampilan hasil pengujian sistem pewaktu RTC DS1307.....52
Gambar 5.22 Diagram blok pengujian rangkaian *driver relay*.....52
Gambar 5.23 Pengujian rangkaian driver relay 53
Gambar 5.24 Pengujian keseluruhan sistem dengan beban setrika..... 57
Gambar 5.25 Pengujian keseluruhan sistem dengan beban televisi..... 58

Daftar Tabel

Tabel 2.1 Tabel konfigurasi TLP 434A.....5
Tabel 2.2 Konfigurasi karakteristik modul RF RLP 434A.....6
Tabel 2.3 Perhitungan baud rate USART.....11
Tabel 2.4 Fungsi Pin LCD.....14
Tabel 2.5 Alamat dan isi register DS1307.....16
Tabel 2.6 Karakteristik *relay* LY2N.....19
Tabel 5.1 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode mati.....41
Tabel 5.2 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode hidup.....42
Tabel 5.3 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode hidup 30 menit.....43
Tabel 5.4 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode hidup 1 jam.....44
Tabel 5.5 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode hidup 2 jam.....45
Tabel 5.6 Hasil pengujian rangkaian *driver relay*.....53
Tabel 5.7 Hasil pengujian keseluruhan sistem mode hidup.....54

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dunia elektronika telah mengalami perkembangan yang sangat pesat. Hasil inovasi dan penemuan-penemuan di bidang ini berlangsung terus-menerus. Saat ini hampir semua pekerjaan dilakukan secara otomatis. Semua perubahan tersebut tidak lain hanya bertujuan untuk mempermudah kehidupan manusia dalam melaksanakan pekerjaan.

Dalam kehidupan sehari-hari, peralatan elektronik seperti televisi, *tape recorder*, *air conditioner* (AC), lampu penerangan dan yang lainnya telah banyak digunakan. Namun saat ini banyak penggunaan peralatan listrik yang kurang efisien, misalnya peralatan listrik tetap menyala saat tidak dibutuhkan. Hal tersebut menyebabkan bertambahnya biaya untuk listrik yang harus dikeluarkan dan juga bisa menimbulkan bahaya, baik terhadap peralatan itu sendiri maupun terhadap lingkungannya, misalnya kebakaran yang terjadi karena arus hubung singkat.

Ada berbagai alasan mengapa peralatan listrik tetap menyala saat tidak dibutuhkan, misalnya pengguna membiarkan peralatan listrik tetap menyala karena letak saklar yang jauh dari jangkauannya atau karena pemakainya lupa atau tertidur.

Dengan latar belakang tersebut, maka perlu dirancang suatu sistem yang dapat memecahkan masalah tersebut. Sistem ini harus dapat mengatur *on/off* peralatan listrik dari semua tempat dengan jangkauan tertentu dan bisa bersifat otomatis. Sehingga dalam skripsi ini dirancang sebuah sistem pengatur dan otomasi *on/off* peralatan listrik menggunakan *remote control* universal. Sistem ini dapat memberikan kemudahan, penghematan energi dan mencegah bahaya yang ditimbulkan peralatan tersebut. Apabila seseorang lupa mematikan lampu suatu ruangan saat orang tersebut sudah berada di ruangan lain, maka remote ini bisa digunakan untuk mematikannya pada jarak jauh. *Remote control* ini juga bisa menunjang pemberian pewaktu pada peralatan elektronik sehingga dapat diatur kapan peralatan elektronik tersebut menyala dan kapan peralatan tersebut mati.

Sistem ini terdiri atas tiga bagian, yaitu bagian *remote control* yang berfungsi mengirimkan sinyal informasi, bagian pusat pengolah data yang akan mengolah

informasi dari *remote* dan rangkaian *driver* yang melakukan pensaklaran peralatan listrik.

Penelitian ini merupakan pengembangan dari skripsi Suci Rositawati, tahun 2002 yang berjudul ”Sistem Pengendali Otomatis untuk Peralatan Listrik dengan Menggunakan *Passive Infra Red* (PIR) dan *Remote Control* Universal”. Pengembangan dilakukan dari sisi pengiriman data dengan menggunakan modul *radio frequency* TLP dan RLP 434A. Modul ini dipilih karena kemampuannya di dalam pengiriman dan penerimaan data yang cukup baik dan harganya tergolong relatif murah. Pengembangan lainnya yaitu pada sistem pewaktu dengan menggunakan RTC DS1307 sehingga sistem memiliki fitur otomatis berdasarkan waktu.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan sebelumnya, maka rumusan masalah dalam skripsi ini ditekankan pada:

1) Bagaimana rancangan sistem *remote control* yang dapat mengirimkan sinyal informasi ke pusat pengolah data menggunakan modul pemancar RF.
2) Bagaimana rancangan unit pengolah data yang dapat menerima dan mengolah data dari *remote control* menggunakan modul penerima RF.
3) Bagaimana rancangan rangkaian *driver* yang dapat melakukan pensaklaran peralatan.
4) Bagaimana rancangan perangkat lunak sistem mikrokontroler sebagai media kendali utama sistem *remote control* universal pengatur *on/off* peralatan listrik.

1.3 Batasan Masalah

Mengacu pada permasalahan yang ada, maka ruang lingkup pada skripsi ini dibatasi pada:

1) Parameter keberhasilan alat adalah sesuai dengan spesifikasi alat yang diinginkan.
2) Pada skripsi ini proses transmisi data menggunakan modul *radio frequency* dibatasi pada perancangan protokol pengiriman data dan format data yang digunakan.
3) Mode otomatis berdasarkan waktu yang digunakan adalah 30 menit menyala, 1 jam menyala, serta menyala dan mati pada jam tertentu.
4) Pengaturan peralatan listrik dibatasi pada pengaturan *on/off*.

1.4 Tujuan

Tujuan skripsi ini adalah merancang dan membuat *remote control* universal yang dapat mengatur *on/off* peralatan listrik. Dengan demikian dapat memberikan kemudahan, penghematan energi dan pencegahan bahaya yang ditimbulkan peralatan listrik tersebut.

1.5 Sistematika Penulisan

Sistematika penulisan dalam laporan ini adalah:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, ruang lingkup, tujuan, dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan alat.

BAB III Metodologi Penelitian

Berisi tentang metode-metode yang dipakai dalam melakukan perancangan, pengujian, dan analisis data.

BAB IV Perencanaan dan Pembuatan Alat

Perancangan dan perealisasian alat yang meliputi spesifikasi, perencanaan diagram blok, prinsip kerja dan realisasi alat.

BAB V Pengujian Alat

Memuat aspek pengujian meliputi penjelasan tentang cara pengujian dan hasil pengujian. Aspek analisis meliputi penilaian atau komentar terhadap hasil-hasil pengujian. Pengujian dan analisis ini terhadap alat yang telah direalisasikan berdasarkan masing-masing blok dan sistem secara keseluruhan.

BAB VI Kesimpulan dan Saran

Memuat intisari hasil pengujian dan menjawab rumusan masalah serta memberikan rekomendasi untuk perbaikan kualitas penelitian dimasa yang akan datang.

BAB II
TINJAUAN PUSTAKA

Skripsi yang berjudul ”Rancang Bangun *Remote Control* Universal sebagai Pengatur *On/Off* Peralatan Listrik” ini dirancang untuk dapat mengontrol *on/off* peralatan listrik menggunakan sebuah *remote control*. Sistem yang dirancang terdiri atas *remote control*, unit pengolah data dan rangkaian *driver relay*. *Remote control* terdiri atas keypad, mikrokontroler dan pemancar *radio frequency*. Unit pengolah data terdiri atas penerima *radio frequency*, mikrokontroler, dan RTC. Rangkaian *driver relay* terdiri atas transistor dan *relay* yang kemudian terhubung dengan berbagai peralatan listrik.

Dalam perencanaan dan pembuatan alat diperlukan pemahaman komponen pendukung yang akan digunakan. Pemahaman yang diperlukan dalam perancangan ini meliputi pemahaman tentang transmisi data, mikrokontroler ATMega8, RTC DS1307, transistor 9014 dan relay LY2N.

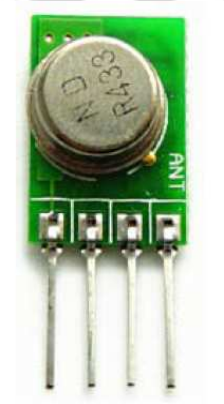
2.1 Transmisi Data

Transmisi data merupakan proses pengiriman data dari satu sumber ke penerima data. Ada tiga komponen utama dalam proses transmisi data, yaitu: sumber data (*source*), media transmisi (*transmission media*), dan penerima (*receiver*). Media transmisi yang dapat digunakan sebagai jalur transmisi atau carrier dari data yang dikirimkan dapat berupa kabel maupun gelombang elektromagnetik. Bila sumber dan penerima data jaraknya cukup jauh, media transmisinya dapat digunakan gelombang elektromagnetik yang dipancarkan melalui udara terbuka yang dapat berupa sistem laser, gelombang radio atau sistem satelit.

Pengiriman data serial melalui media udara menggunakan gelombang radio sebagai pembawa data. Secara sederhana proses pengiriman data menggunakan gelombang radio adalah, sinyal informasi atau data yang akan dikirimkan ditumpangkan terlebih dahulu ke sinyal pembawa. Proses menumpangkan sinyal informasi ini disebut dengan modulasi. Gabungan antara kedua sinyal tersebut kemudian akan dipancarkan oleh transmiter. Pada receiver, gelombang pembawa yang membawa sinyal informasi tersebut diterima, kemudian dipisahkan antara gelombang pembawa dan sinyal informasi, sehingga diperoleh kembali sinyal informasi. Proses ini disebut demodulasi.

2.1.1 Bagian Pemancar (TLP434A)

Bagian pemancar dari sistem ini menggunakan TLP 434A yang merupakan keluaran dari LAIPAC Technology. Pin 2 dari modul ini merupakan pin data masukan. Data masukan yang diberikan dapat berupa data digital. Bentuk dari TLP 434A ditunjukkan dalam Gambar 2.1.



Gambar 2.1 Bentuk TLP 434A

Sumber: LAIPAC, 2005:1

Spesifikasi teknis modul TLP 434A yaitu :

- Menggunakan tegangan catu daya DC dengan range antara 2.0 – 12.0 volt
- Sistem modulasi yang digunakan yaitu modulasi ASK
- Kecepatan data *rate*nya mulai dari 512 bps sampai dengan nilai 200 kbps
- Data masukan berupa data masukan asinkron
- Frekuensi pemancar yaitu 433.92 MHz
- Arus rata-rata yang digunakan yaitu 10mA

Tabel 2.1 menunjukkan konfigurasi teknis modul TLP 434A.

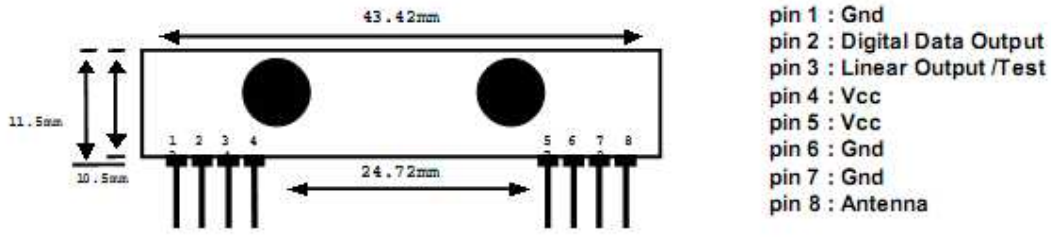
Tabel 2.1 Tabel konfigurasi TLP 434A

Parameter	Sym.	Min.	Typ.	Max.	Unit
Operating Frequency (200KHz)	Vcc		433.92		MHz
Data Rate	ASK			8K	Kbps
Transmitter Performance(OOK@2.4kbps)					
Peak Input Current,12 Vdc Supply	ITP			45	mA
Peak Output Power	PO		10		mW
Tum On/ Tum Off Time	T ON/T OFF			1	US
Power Supply Voltage Range	Vcc	3		12	VDC
Operating Ambient Temperature	TA	-20		+85	centigrade
Tx Antenna Out (3V) +2.4dB	Vcc				mA

Sumber: LAIPAC, 2005:1

2.1.2 Bagian Penerima (RLP 434A)

Pada bagian penerima yang menggunakan modul RF RLP 434A, data yang dikirim oleh bagian pemancar yang menggunakan modul RF TLP 434A akan didemodulasi sehingga diperoleh sinyal data. Keluaran dan pembacaan data dari modul RF ini yaitu berupa data digital sehingga nantinya dapat dikomunikasikan dengan sistem komunikasi data serial. Gambar 2.2 menunjukkan konfigurasi pin RLP 434A.



Gambar 2.2 Konfigurasi pin modul RF RLP 434A

Sumber: LAIPAC, 2005:1

Modul RLP ini menggunakan modulasi digital ASK (*Amplitudo Shift Keying*). Modul ini memiliki frekuensi kerja yang sama antara penerima dan pemancar dengan frekuensi 433,92 MHz. Menggunakan tegangan catu daya DC dengan range antara 3.3–6.0 volt. Mempunyai daya output RF sebesar 2-12mW. Kecepatan data serial *typical* yang ditransmisikan adalah sebesar 4,8 kbps. Tabel 2.2 menunjukkan konfigurasi karakteristik lengkap RLP 434A.

Tabel 2.2 Konfigurasi karakteristik modul RF RLP 434A

Symbol	Parameter	Conditions	Min	Typ	Max	
Vcc	Operating supply voltage		3.3	5.0V	6.0	V
Itot	Operating Current		-	4.5		mA
Vdata	Data Out	Idata = +200 uA (High)	Vcc-0.5	-	Vcc	V
		Idata = -10 uA (Low)	-	-	0.3	V

Electrical Characteristics					
Characteristics	SYM	Min	Typ	Max	Unit
Operation Radio Frequency	FC	315, 418 and 433.92			MHz
Sensitivity	Pref		-110		dBm
Channel Width			+-500		Khz
Noise Equivalent BW			4		Khz
Receiver Turn On Time			5		ms
Operation Temperature	Top	-20	-	80	C
Baseboard Data Rate			4.8		KHz

Sumber: LAIPAC, 2005:1

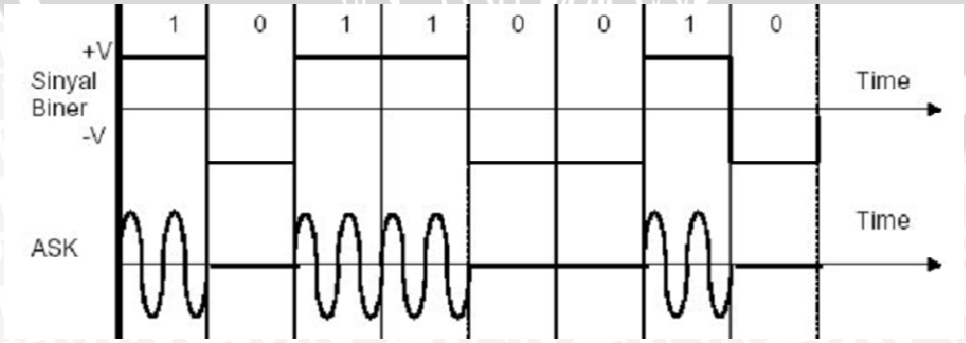
2.1.3 Modulasi Digital ASK

Informasi dapat dipancarkan secara sederhana dengan pensaklaran *on* dan *off* dari *carrier* yang sesuai. Proses pensaklaran ini disebut dengan *keying*. Contoh sederhananya adalah pentransmisian pesan dengan menggunakan gelombang radio. Karakteristik dari metode ini adalah penerima data hanya dapat mendekodekan suatu data bila penerima memiliki dan menggunakan kode yang sama dengan yang dipakai oleh pengirim data. Dalam modulasi digital, *carrier*nya analog/harmonik sedangkan sinyal pemodulasinya digital.

Suatu *carrier* harmonik terbentuk dari 3 parameter : Amplitudo (A), Frekuensi (f), dan Phase (φ). Ketiganya dapat menjadi obyek modulasi. Karena itu, terdapat 3 macam modulasi digital :

1) *Amplitudo Shift Keying* (ASK)
2) *Frequency Shift Keying* (FSK)
3) *Phase Shift Keying* (PSK)

Dalam ASK, amplitudo *carrier* tersaklar *on* dan *off* sesuai dengan kecepatan sinyal pemodulasi. Proses pensaklaran (*switching*) membangkitkan komponen spektral yang baru yang tidak ada dalam spektrum sinyal sebelum pensaklaran. ASK mempunyai kelemahan yaitu bila terjadi gangguan pada saluran transmisi atau adanya sedikit kesalahan pada saat pengiriman, kondisi biner 0 tidak akan dapat dibedakan oleh *receiver*. Pentransmisian sinyal akan selalu disertai oleh kesalahan transmisi. Karena itu dimungkinkan adanya mekanisme *control* otomatis untuk memonitor pentransmisian data. Gambar 2.3 menunjukkan contoh sinyal *carrier* yang dimodulasi oleh sinyal biner menggunakan modulasi digital ASK.

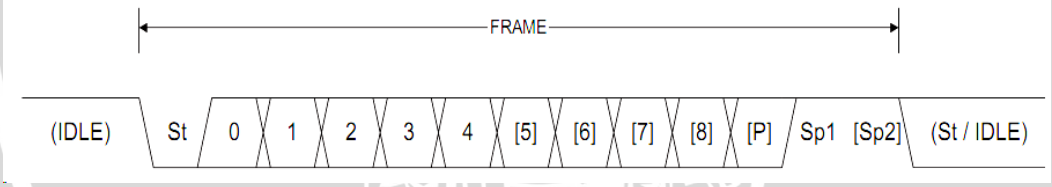


Gambar 2.3 Sinyal termodulasi ASK
Sumber: Stallings, 2004

2.1.4 Komunikasi Serial

Pada prinsipnya, komunikasi serial ialah komunikasi dimana pengiriman data dilakukan per bit, sehingga lebih lambat dibandingkan komunikasi parallel seperti pada port printer yang mampu mengirim 8 bit sekaligus dalam sekali waktu. Devais pada komunikasi serial port dibagi menjadi 2 (dua) kelompok yaitu *Data Communication Equipment* (DCE) dan *Data Terminal Equipment* (DTE). Pengiriman data secara serial dapat berupa sinkron, yaitu pengiriman clock dilakukan bersamaan dengan data, atau berupa asinkron, yaitu pengiriman clock tidak bersamaan dengan data namun secara dua tahap, saat data dikirim dan data diterima. Untuk istilah yang sering digunakan untuk mengirim dan menerima data secara asinkron biasa disebut *Universal Asynchronous Receiver Transmitter* (UART). Komunikasi data serial menggunakan UART sangat umum dan mudah penggunaannya, misalnya pada port serial PC. Pada UART jalur pengiriman dan penerimaan data serial dipisahkan.

Setiap pengiriman data pada UART menggunakan bit tanda *start* bit dan *stop* bit. Jalur data yang digunakan hanya satu untuk setiap pengiriman data. Data-data serial dikirim melewati jalur data satu persatu setiap satuan waktu. Gambar 2.4 menunjukkan format pengiriman data serial secara asinkron.



Gambar 2.4 Format frame data serial USART
Sumber: Atmel, 2007:137

2.2 Mikrokontroler ATMega8

Sebagai sentral/pusat dari pengontrolan sistem ini digunakan mikrokontroller ATMega8. Mikrokontroller ini yang bertugas membaca data dari tombol dan mengolah data tersebut menjadi format data yang siap dikirimkan serta mengirimkannya secara serial melalui modul RF. Mikrokontroller ATMega8 ini diproduksi oleh ATMEL *Company* Amerika Serikat dan merupakan salah satu anggota keluarga dari jenis AVR. Mikrokontroller ini memiliki fasilitas komunikasi serial untuk melakukan pengiriman/penerimaan data dengan handphone. IC jenis ini berorientasi pada *control* yang dapat diprogram ulang. Mikrokontroller ATMega8 mempunyai karakteristik utama sebagai berikut:

- Register serbaguna sejumlah 32x8 bit.
- *In-system Self-programmable* memori program *Flash* sebesar 8 K*bytes*.
- EEPROM (*Electrical Erasable Programable Read Only Memory*) sebesar 512*bytes*.
- SRAM internal 1 K*bytes*.
- ADC (*Analog to Digital Converter*) 10 bit sebanyak 8 buah.
- I/O sejumlah 23 jalur yang dapat dipakai semua.
- *Timer*/*counter* 16 bit 1 buah.
- *Timer*/*counter* 8 bit 2 buah.
- *Programmable* serial USART.
- Frekuensi kerja 0 sampai 16 MHz.
- Tegangan operasi antara 4,5 volt sampai 5,5 volt.

2.2.1 Struktur dan Operasi Port

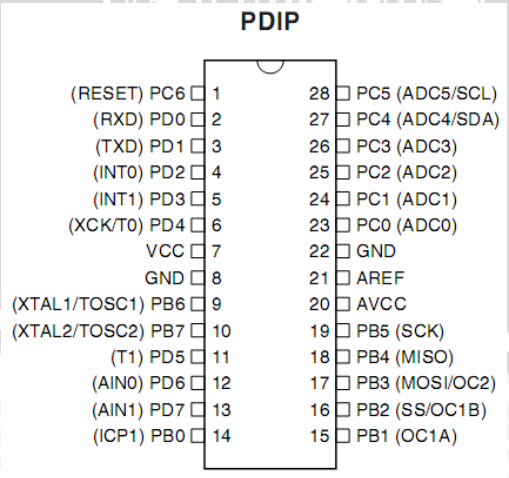
Masing-masing kaki dalam mikrokontroler ATMega8 mempunyai fungsi tersendiri. Dengan mengetahui fungsi masing-masing kaki mikrokontroller ATMega8, perancang aplikasi mikrokontroler ATMega8 akan lebih mudah merencanakan dan membuat sistem yang dirancang. ATMega8 mempunyai 28 pin, susunan masing-masing pin dapat dilihat dalam Gambar 2.5.

Fungsi pin-pin secara keseluruhan dari mikrokontroller ATMega8 adalah sebagai berikut:

- *Port* B (Pin B0-7), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus Port B diantaranya adalah : Port B0 sebagai ICP (*Timer/counter input capture pin*), Port B1 sebagai OC1A (*Timer/counter* 1 *output compare* A *match output*), Port B2 sebagai SS (SPI *slave select input*) dan OC1B (*Timer/counter* 1 *output compare* B *match output*), Port B3 sebagai OC2 (*timer/counter 2 compare match output*) dan MOSI (SPI bus master *output*/slave *input*), Port B4 sebagai MISO (SPI bus master *input*/slave *output*), Port B5 sebagai SCK (SPI bus serial *clock*), Port B6 dan Port B7 adalah TOSC1 (*Timer Oscilator* pin1) dan TOSC2 (*Timer oscillator* pin2).
- *Port* C (Pin C0-6), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus dari Port C diantaranya adalah : Port C0-5 sebagai saluran masukan ADC *channel* 0-5, selain itu, Port C4 sebagai SDA (*Two-*

Wire serial bus data *input*/*output* line), Port C5 sebagai SCL (*Two-Wire serial bus clock line*),dan Port C6 sebagai pin RESET.

- *Port* D (Pin D0-7), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus dari Port D diantaranya adalah : Port D0 sebagai RXD (USART *input* pin), Port D1 sebagai TXD (USART *output* pin), Port D2 sebagai INT0 (Eksternal interupt 0 *input*), Port D3 sebagai INT1 (Eksternal interupt 1 *input*), Port D4 sebagai T0 (*timer/counter 0* eksternal *counter input*) & XCK (USART eksternal *clock input/output*), Port D5 sebagai T1 (*timer/counter* eksternel *counter input*), Port D6 sebagai AIN0 (*Analog comparator positive input*), dan Port D7 sebagai AIN1 (*Analog comparator negative input*).
- Pin 1 RESET, merupakan saluran dua masukan untuk me-*reset* mikrokontroller dengan cara memberi masukan logika rendah.
- Pin 7 VCC, merupakan saluran masukan untuk catu daya positif sebesar 5 volt DC.
- Pin 8 GND, merupakan Ground dari seluruh rangkaian.
- Pin 9 dan 10 (XTAL1 dan XTAL2), merupakan saluran untuk mengatur pewaktuan sistem. Pewaktuan dapat menggunakan pewaktuan internal maupun eksternal.
- Pin 21 AREF, merupakan Pin analog referensi untuk masukan ADC.
- Pin 22 GND, merupakan ground dari ADC.
- Pin 20 AVcc, merupakan pin catu tegangan untuk A/D Converter pada port C (0-3). Pin ini harus dihubungkan dengan Vcc, walaupun fitur ADC tidak digunakan. Saat ADC digunakan, AVcc sebaiknya dihubungkan dengan Vcc melalui low-pass filter.



Gambar 2.5 Konfigurasi pin mikrokontroler ATmega8

Sumber: Atmel, 2007:2

2.2.2 Komunikasi Serial

USART (*Universal Synchronous and Asynchronous serial Receiver and Transmitter*) adalah seperangkat komunikasi serial dengan tingkat fleksibilitas yang tinggi. Beberapa fitur yang dimiliki oleh USART yaitu:

- Komunikasi *full-duplex* dengan *register* serial untuk penerima dan pengirim data.
- Beroperasi pada mode sinkron dan asinkron.
- Memiliki pembangkit *baud rate* beresolusi tinggi.
- Pengiriman data serial dengan format *frame* 5, 6, 7, 8, dan 9 bit dengan 1 atau 2 bit stop.
- Dukungan perangkat keras terhadap pengecekan paritas genap atau paritas ganjil.
- Pendeteksi kesalahan pada format *frame* data yang dikirim.
- Pendeteksi pengiriman kelebihan data.
- Memiliki 3 layanan interupsi yaitu TX *Complete*, TX *Data Register Empty*, dan RX *Complete*.
- Mode komunikasi *multiprocessor*.

Nilai USART *Baud Rate Register* (UBRR) digunakan untuk menentukan kecepatan transfer data pada komunikasi serial dapat dihitung menggunakan persamaan yang terdapat dalam Tabel 3.

Tabel 2.3 Perhitungan baud rate USART

Mode Operasi	Persamaan perhitungan Baud Rate	Persamaan perhitungan UBRR Rate
Asinkron: Normal mode	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asinkron: Double speed mode	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Sinkron: Master mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Sumber: Atmel, 2007:136

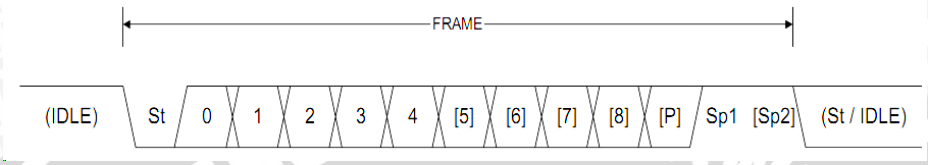
Keterangan:

BAUD : *Baud Rate* dalam *bit per second* (bps).

f_{OSC} : Frekuensi *clock* dari sistem osilator.

UBRR : *Register Baud Rate* yang terdiri atas *register* UBRR*Low* dan *register* UBRR*High*.

Format *frame* data komunikasi serial didefinisikan sebagai sebuah karakter dari bit-bit data yang memiliki bit-bit sinkronisasi yaitu *start* bit dan *stop* bit, serta pilihan berupa bit paritas untuk mendeteksi terjadinya kesalahan pengiriman data. Format *frame* pengiriman data serial USART ditunjukkan dalam Gambar 2.6.



Gambar 2.6 Format frame data serial USART

Sumber: Atmel, 2007:137

Keterangan:

St : Bit *start* selalu berlogika rendah.

P : Bit paritas genap atau paritas ganjil.

Sp : Bit *stop* selalu berlogika tinggi (bit stop bisa berjumlah 1 atau 2).

IDLE : Tidak ada data yang ditransfer pada RX dan TX, IDLE selalu berlogika tinggi.

Perhitungan bit paritas dilakukan dengan cara menggunakan logika *exclusive*-or (XOR) terhadap semua bit-bit data yang ditransmisikan. Jika yang digunakan adalah paritas ganjil, maka hasil dari keluaran logika *exclusive*-or dinegasikan. Hubungan antara bit paritas dengan bit-bit data sesuai dengan persamaan berikut:

$$P_{even} = d_{n-1} \oplus ... \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus ... \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1 \quad (2\text{-}3)$$

Keterangan:

P_{even} : paritas yang digunakan adalah paritas genap.

P_{odd} : paritas yang digunakan adalah paritas ganjil.

d_n : bit data ke-n dari sebuah karakter.

2.3 LCD LMB162ADC

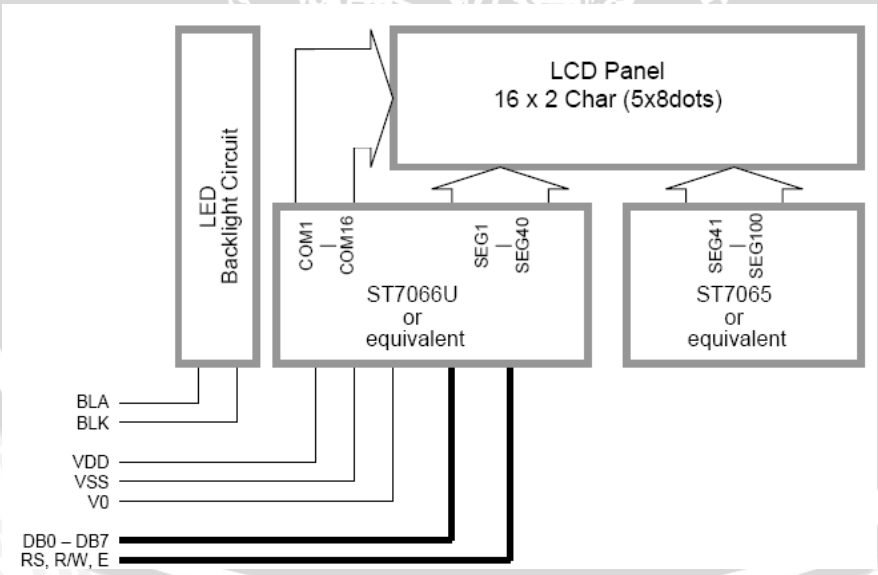
Dalam sistem ini, LCD digunakan sebagai penampil keluaran. Tipe LCD yang digunakan adalah LMB162ADC buatan Shenzhen TOPWAY Technology Co. Tipe ini

mempunyai konsumsi daya rendah dengan menggunakan teknologi CMOS. Spesifikasi dari LCD ini adalah sebagai berikut :

- Mempunyai 16 karakter dengan 5 x 8 dot matrik dan kursor
- Rasio kerja : 1/16
- *Power On Reset* secara otomatis
- Mempunyai 2 memori utama, yaitu Character Generator RAM (CGRAM) dan Display Data RAM (DDRAM)
- Antarmuka dengan MPU empat bit dan delapan bit
- Tegangan masukan sebesar 5V±6%

LCD ini menggunakan komunikasi 4 bit yaitu DB4-DB7. Hal ini dilakukan untuk menghemat pin mikrokontroller yang digunakan untuk komunikasi data ke LCD.

Masukan yang diperlukan untuk mengendalikan modul ini berupa bus data yang masih termultiplek dengan bus alamat serta 3 bit sinyal kontrol. Sementara pengendalian dot matrik LCD dilakukan secara internal oleh kontroller yang sudah terpasang pada modul LCD. Fungsi pin LCD LMB162ADC dapat dilihat dalam tabel 2.4 sedangkan untuk blok diagram dari LCD ditunjukkan dalam Gambar 2.7.



Gambar 2.7 Blok diagram LCD LMB162ADC

Sumber : Topway, 2004 : 3

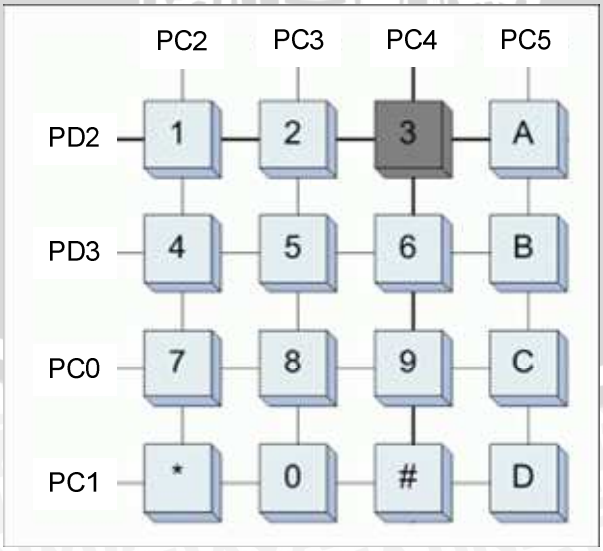
Tabel 2.4 Fungsi Pin LCD

Penyemat	Fungsi
DB_0-DB_7	Merupakan saluran data, berisi perintah dan data yang akan ditampilkan di LCD
Enable	Sinyal operasi awal, sinyal ini mengaktifkan data tulis atau baca
R/W	Sinyal seleksi tulis atau baca 0 : tulis 1 : baca
RS	Sinyal pemilih *register* 0 : instruksi *register* (tulis) 1 : data *register* (baca dan tulis)

Sumber : Topway, 2004 : 4

2.4 Keypad Matrik 4x4

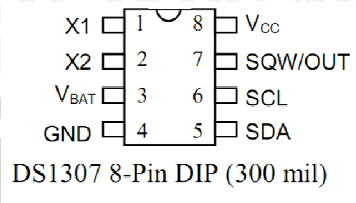
Tombol tekan atau keypad merupakan suatu sarana yang digunakan untuk memasukkan data ke suatu alat atau device. Keypad yang digunakan ini berupa keypad matrik 4x4 yang terhubung sebagai baris dan kolom. Jika tombol 1 ditekan maka yang terhubung adalah kolom 1 dan baris 1, jika yang ditekan tombol 2 maka yang terhubung adalah kolom 2 dan baris 1, dan seterusnya. Keypad dapat dihubungkan langsung ke port dari mikrokontroller AVR tanpa komponen tambahan. Gambar dari Keypad matrix 4x4 ditunjukkan dalam Gambar 2.8.



Gambar 2.8 Rangkaian keypad matrik 4x4

2.5 RTC DS1307

DS1307 merupakan sebuah IC RTC (*Real Time Clock*) yang dapat merekam dan memberikan informasi waktu lengkap mulai informasi detik, menit, jam, tanggal, bulan hingga tahun. Susunan pin IC RTC DS1307 ditunjukkan dalam Gambar 2.9.



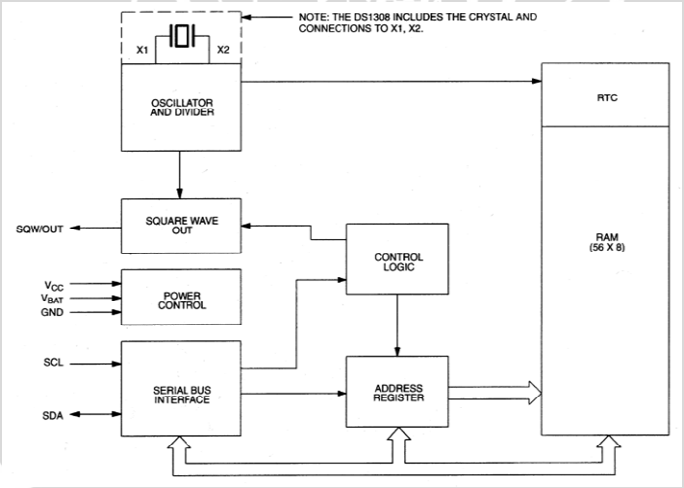
Gambar 2.9 Susunan pin DS1307

Sumber: Dallas, 2000: 1

Berbagai fitur yang disediakan chip ini antara lain:

- Perhitungan *real time clock* mulai dari detik menit, jam, tanggal, bulan hingga tahun
- RAM untuk penyimpanan data sebesar 56 byte
- *Interface* I2C yang dikembangkan oleh Philips
- Dapat memberikan output berupa sinyal pulsa yang dapat diprogram
- Mengkonsumsi arus kurang dari 500 nA dalam mode baterai dengan *oscillator* yang tetap berjalan.

Diagram blok DS1307 ditunjukkan dalam Gambar 2.10.



Gambar 2.10 Diagram blok DS1307

Sumber: Dallas, 2000: 2

X_1 dan X_2 dihubungkan dengan X_{tal} berukuran 32,768 kHz. V_{bat} dihubungkan dengan catu daya baterai sebesar 3 V. Jenis komunikasi yang digunakan oleh DS1307 adalah komunikasi serial I2C yang dikembangkan oleh Philips. Komunikasi serial I2C

menggunakan dua jalur transmisi, yaitu SCL yang berfungsi sebagai jalur *clock* dan SDA yang berfungsi sebagai jalur data.

Informasi waktu dapat ditulis dan dibaca menggunakan mikrokontroler. Informasi waktu diformat dalam bentuk 2 byte bilangan BCD yang disimpan dalam register-register berbeda. Tabel 2.5 menunjukkan alamat dan isi register dalam DS1307.

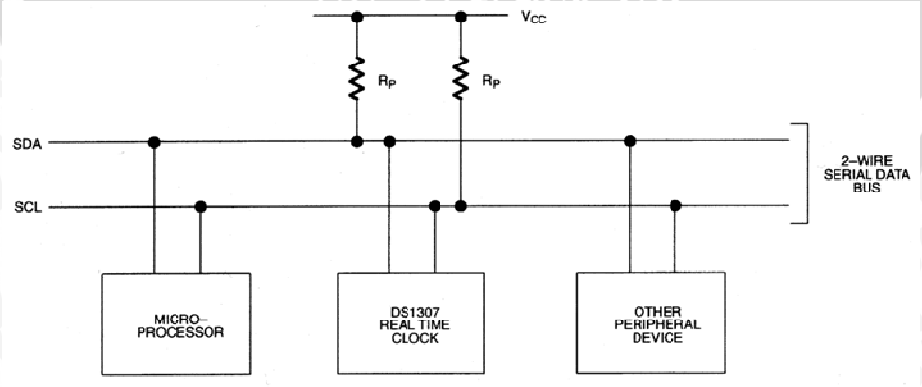
Tabel 2.5 Alamat dan isi register DS1307

	BIT7							BIT0	
00H	CH	10 SECONDS			SECONDS				00–59
	X	10 MINUTES			MINUTES				00–59
	X	12/24	10 HR A/P	10 HR	HOURS				01–12 / 00–23
	X	X	X	X	X	DAY			1–7
	X	X	10 DATE		DATE				01–28/29 / 01–30 / 01–31
	X	X	X	10 MONTH	MONTH				01–12
	10 YEAR				YEAR				00–99
07H	OUT	X	X	SQWE	X	X	RS1	RS0	

Sumber: Dallas, 2000: 4

2.5.1 Komunikasi I2C

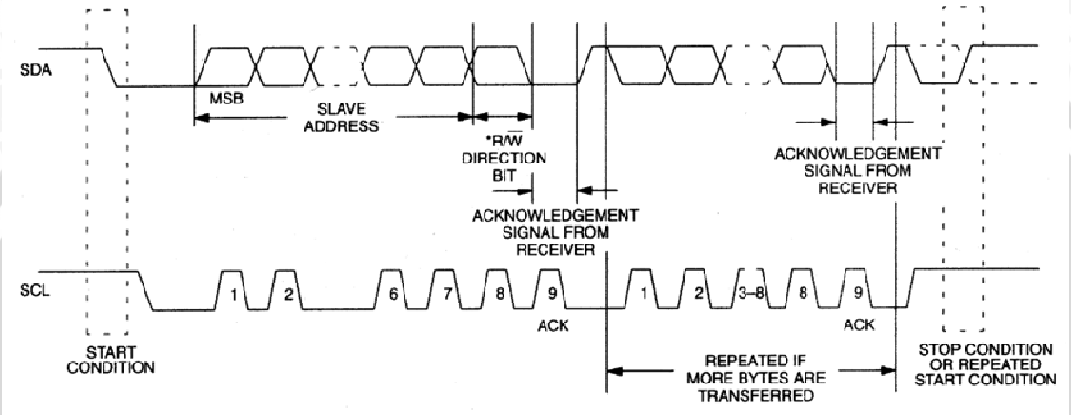
Secara fisik, jalur komunikasi I2C terdiri atas dua kabel. Dua kabel tersebut umumnya dikenal sebagai SDA dan SCL. SDA adalah jalur data baik masuk maupun keluar, sedangkan SCL adalah jalur *clock*. I2C dapat dihubungkan pada lebih dari satu *slave*. Bentuk umum susunan perangkat yang berkomunikasi dengan I2C ditunjukkan dalam Gambar 2.11.



Gambar 2.11 Susunan perangkat dengan jalur komunikasi I2C

Sumber: Dallas, 2000: 5

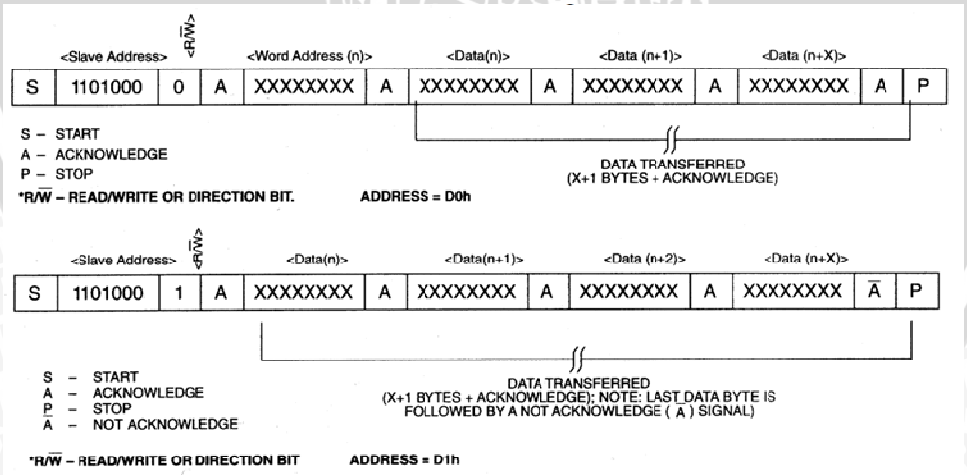
Urutan komunikasi dalam protokol I2C adalah sebagai berikut. Pertama, mikrokontroler memberikan kondisi start. Kemudian mikrokontroler mengirimkan sinyal alamat perangkat yang akan diakses. Tiap IC *slave* akan membandingkan alamat yang dikirim dengan alamat yang dimiliki. Jika tidak sesuai, maka *slave* akan menunggu hingga master mengirim sinyal stop. Namun jika sesuai, maka *slave* mengirim sinyal *acknowledge*. Saat mikrokontroler menerima sinyal *acknowledge*, maka mikrokontroler dapat mengirimkan data. Setelah selesai, mikrokontroler mengirim sinyal stop. Gambar 2.12 menunjukkan representasi sinyal komunikasi I2C.



Gambar 2.12 Representasi sinyal komunikasi I2C

Sumber: Dallas, 2000: 6

Gambar 2.13 Menunjukkan urutan data dalam komunikasi I2C.



Gambar 2.13 Urutan data dalam komunikasi I2C

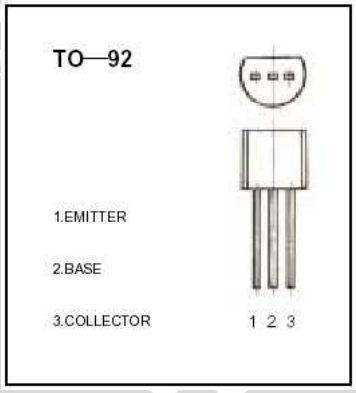
Sumber: Dallas, 2000: 7

2.6 Transistor 9014

Transistor adalah komponen yang sangat penting dalam dunia elektronik modern. Transistor pada umumnya mempunyai tiga kaki, kaki pertama disebut *basis*, kaki berikutnya dinamakan *kolektor* dan kaki yang ketiga disebut *emitor*. Suatu arus listrik yang kecil pada basis akan menimbulkan arus yang jauh lebih besar diantara kolektor dan emitornya, maka dari itu transistor digunakan untuk memperkuat arus (amplifier). Terdapat dua jenis transistor ialah jenis NPN dan jenis PNP. Pada transistor jenis NPN tegangan basis dan kolektornya positif terhadap emitor, sedangkan pada transistor PNP tegangan basis dan kolektornya negatif terhadap tegangan emitor.

Pada prinsipnya, suatu transistor terdiri atas dua buah dioda yang disatukan. Agar transistor dapat bekerja, kepada kaki-kakinya harus diberikan tegangan, tegangan ini dinamakan *bias voltage*. Basis emitor diberikan forward voltage, sedangkan basis kolektor diberikan reverse voltage. Sifat transistor adalah bahwa antara kolektor dan emitor akan ada arus (transistor akan menghantar) bila ada arus basis. Makin besar arus basis makin besar penghatarannya.

Transistor 9014 merupakan transisitor NPN dengan konfigurasi pin ditunjukkan dalam Gambar 2.14. Gambar 2.15 menunjukkan karakteristik transistor 9014.



Gambar 2.14 Konfigurasi pin transistor 9014

Sumber: Wing Shing, 2002: 1

Parameter	Symbol	Test conditions	MIN	TYP	MAX	UNIT
Collector-base breakdown voltage	$V(BR)_{CBO}$	Ic= 100 μA, I_E=0	50			V
Collector-emitter breakdown voltage	$V(BR)_{CEO}$	Ic= 0. 1 mA, I_B=0	45			V
Emitter-base breakdown voltage	$V(BR)_{EBO}$	I_E= 100 μA, I_C=0	5			V
Collector cut-off current	I_{CBO}	V_{CB}=50 V, I_E=0			0.1	μA
Collector cut-off current	I_{CEO}	V_{CE}=35 V, I_B=0			0.1	μA
Emitter cut-off current	I_{EBO}	V_{EB}= 3 V, I_C=0			0.1	μA
DC current gain(note)	$H_{FE\ (1)}$	V_{CE}= 5 V, I_C= 1mA	60		1000	
Collector-emitter saturation voltage	V_{CE}(sat)	I_C= 100mA, I_B= 5 mA			0.3	V
Base-emitter saturation voltage	V_{BE}(sat)	I_C= 100 mA, I_B= 5mA			1	V
Transition frequency	f_T	V_{CE}= 5 V, I_C= 10mA f =30MHz	150			MHz

Gambar 2.15 Karakteristik transistor 9014

Sumber: Wing Shing, 2002:1

2.6 *Relay* LY2N

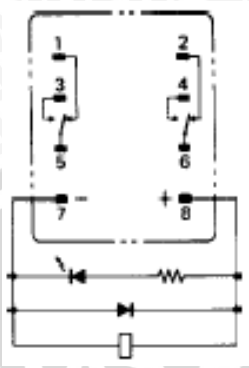
Relay LY2N merupakan rangkaian yang bersifat elektronis sederhana dan tersusun oleh saklar, medan elektromagnet (kawat koil) dan poros besi. Cara kerja komponen ini dimulai pada saat mengalirnya arus listrik melalui koil, lalu membuat medan magnet sekitarnya mengubah posisi saklar sehingga menghasilkan arus listrik yang lebih besar. Tabel 2.6 menunjukkan karakteristik *relay* LY2N.

Tabel 2.6 Karakteristik *relay* LY2N

Contact resistance		50 mΩ max.
Operate time		25 ms max.
Release time		25 ms max.
Operating frequency	Mechanically	18,000 operations/hour
	Under rated load	1,800 operations/hour
Insulation resistance		100 MΩ min. (at 500 VDC)
Dielectric strength		2,000 VAC, 50/60 Hz for 1 minute 1,000 VAC, 50/60 Hz for 1 minute between contacts of same polarity
Vibration	Mechanical durability	10 to 55 Hz, 1.00 mm (0.04 in) double amplitude
	Malfunction durability	10 to 55 Hz, 1.00 mm (0.04 in) double amplitude
Shock	Mechanical durability	1,000 m/s² (approx. 100 G)
	Malfunction durability	200 m/s² (approx. 20 G)
Ambient temperature	Operating	-40° to 70°C (-40° to 158°F)
Humidity		35 to 85% RH
Service Life	Mechanically	AC: 50 million operations min. (at operating frequency of 18,000 operations/hour) DC: 100 million operations min. (at operating frequency of 18,000 operations/hour)
	Electrically	See "Characteristic Data"
Weight		SPDT, DPDT: Approx. 40 g (1.41 oz), 3PDT: Approx. 50 g (1.76 oz) 4PDT: Approx. 70 g (2.47 oz)

Sumber: Omron, 2000: 5

Relay LY2N terdiri dari 8 terminal, Gambar 2.16 menunjukkan sususan terminal relay LY2N.



Gambar 2.16 Susunan terminal relay LY2N

Sumber: Omron, 2000: 12

BAB III
METODOLOGI PENELITIAN

Penyusunan skripsi ini didasarkan pada masalah yang bersifat aplikatif, yaitu perencanaan dan perealisasian alat agar dapat menampilkan unjuk kerja sesuai dengan yang direncanakan dengan mengacu pada rumusan masalah. Metode yang digunakan dalam penulisan skripsi ini adalah sebagai berikut:

3.1 Perancangan dan Pembuatan Alat

Pada perancangan alat dilakukan dengan membuat diagram blok keseluruhan alat. Penentuan komponen yang digunakan pada alat dilakukan secara bertahap sesuai dengan diagram blok. Perancangan perangkat lunak dilakukan dengan membuat diagram alir program untuk mengendalikan sistem secara keseluruhan.

Pembuatan alat dilakukan dengan membuat PCB (meliputi proses desain dengan menggunakan perangkat lunak Eagle versi 5.0.0, pengetsaan dan pengeboran), pemasangan komponen, dan pengemasan alat. Pembuatan perangkat lunak dilakukan dengan merealisasikan diagram alir ke dalam bahasa C meliputi penulisan kode, pengujian, dan kompilasi program dengan menggunakan perangkat lunak CodeVisionAVR.

3.2 Pengujian Alat

Untuk mengetahui apakah alat yang telah dibuat sesuai dengan yang direncanakan maka dilakukan pengujian rangkaian. Pengujian yang dilakukan terbagi dua, yaitu:

a) Pengujian perangkat keras

   Perangkat keras yang telah dibuat tahap demi tahap akan diuji satu persatu sesuai diagram blok. Pengujian perangkat keras ini meliputi pengujian LCD, pengujian antarmuka *keypad* dengan mikokontroler, pengujian *remote control*, pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A, pengujian sistem pewaktu RTC DS1307, dan pengujian rangkaian driver.

b) Pengujian secara keseluruhan

   Pengujian secara keseluruhan dilakukan dengan menghubungkan tiap perangkat keras sesuai dengan diagram blok dan menjalankan perangkat lunaknya.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Bab ini akan menjelaskan tentang perancangan sekaligus pembuatan sistem. Sistem yang dirancang dan dibuat terdiri atas tiga bagian, yaitu *remote control*, unit pengolah data dan rangkaian *driver*. Sistem ini dirancang untuk dapat mengontrol *on/off* peralatan listrik menggunakan sebuah *remote control*.

4.1 Penentuan Spesifikasi Alat

Spesifikasi alat secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi alat yang direncanakan adalah sebagai berikut:

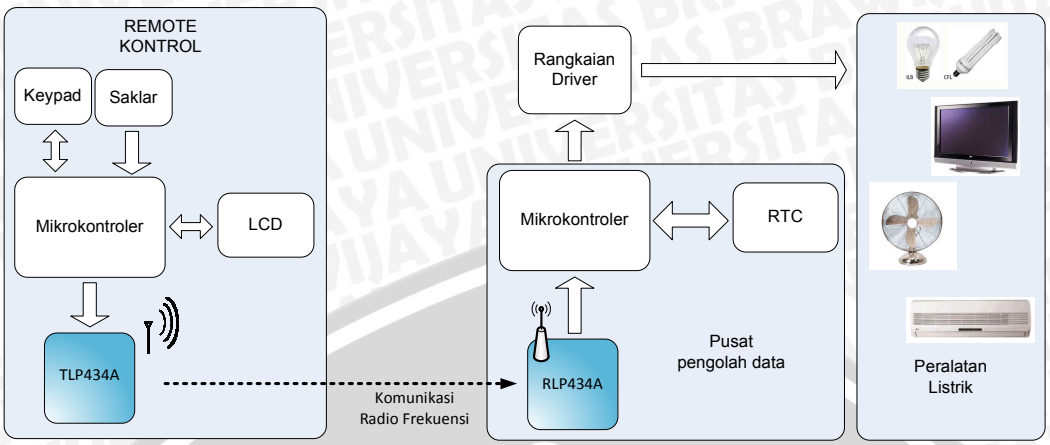
1) Jalur transmisi data yang digunakan adalah frekuensi radio dengan menggunakan modul RF (TLP434A dan RLP434A) dengan frekuensi *carrier* yang digunakan adalah 433,92 MHz.
2) Kecepatan transmisi data yang digunakan adalah 9600 bps.
3) Jarak maksimal antara *remote control* dengan pusat pengolah data di dalam ruangan adalah 20 meter.
4) Jumlah peralatan listrik yang bisa diatur adalah 16 buah.
5) Sistem menggunakan catu daya DC 5 volt.

4.2 Diagram Blok Sistem

Secara garis besar, diagram blok perancangan *hardware* sistem secara keseluruhan ditunjukkan dalam Gambar 4.1.

Fungsi masing-masing bagian dalam diagram blok ini adalah sebagai berikut:

1) Tombol-tombol berfungsi untuk mengatur atau memilih peralatan listrik yang akan dinyalakan atau dimatikan serta untuk mengakses fitur-fitur yang ada dalam sistem ini.
2) Saklar berfungsi untuk memilih fitur yang digunakan yang ada dalam sistem ini.
3) Mikrokontroler 1 berfungsi sebagai pengolah data-data dari tombol-tombol dan kemudian mengirimkan data hasil pengolahan secara serial ke pusat pengolah data melalui TLP434A.
4) LCD berfungsi untuk menampilkan peralatan listrik yang dipilih untuk dinyalakan atau dimatikan dan menampilkan pengaturan waktu dalam fitur timer.



Gambar 4.1 Diagram blok sistem secara keseluruhan

5) TLP 434A berfungsi untuk mengirimkan data serial dari mikrokontroler ke pusat pengolah data melalui *radio frequency*.
6) RLP 434A berfungsi untuk menerima data dari *remote* (TLP434A) dan mengirimkannya ke mikrokontroler 2.
7) Mikrokontroler 2 berfungsi untuk mengolah data yang dikirimkan oleh *remote* serta mengambil data waktu dari RTC dan mengatur ON/OFF peralatan listrik rumah.
8) RTC berfungsi sebagai penyedia data waktu.
9) Rangkaian *driver relay* berfungsi untuk menghubungkan mikrokontroler dengan peralatan listrik rumah.

Prinsip kerja sistem ini adalah pengguna dapat menyalakan atau mematikan peralatan listrik rumah dengan menekan tombol tertentu dalam *remote control*. Ketika tombol ditekan, maka mikrokontroler 1 akan mendeteksi tombol mana yang ditekan dan mode yang dipilih, mikrokontroler kemudian akan mengolah data dari hasil penekanan tombol tersebut yang kemudian ditampilkan oleh LCD. Mikrokontroler akan mengirimkan data hasil pengolahan tersebut secara serial ke pusat pengolah data. Data serial ini masuk ke modul pemancar RF TLP 434A dan dikirimkan melalui frekuensi radio ke penerima RF RLP 434A yang ada di pusat pengolah data. Setelah data diterima oleh RLP 434A maka data akan dibaca oleh mikrokontroler 2 dan diolah. Mikrokontroler kemudian akan memberikan sinyal kepada rangkaian *driver relay* untuk menyalakan atau mematikan peralatan listrik berdasarkan hasil pengolahan data.

Sistem ini juga memiliki fitur *timer* dan otomatis. Fitur *timer* ini akan menyalakan peralatan listrik rumah selama selang waktu tertentu kemudian

mematikannya secara otomatis. Fitur ini juga bisa menyalakan dan mematikan peralatan listrik rumah pada pada waktu-waktu tertentu secara otomatis.

4.3 Perancangan Perangkat Keras (Hardware)

Perancangan perangkat keras meliputi perancangan perangkat keras tiap-tiap blok sistem yang telah disusun. Perancangan perangkat keras yang dilakukan yaitu:

1. Perancangan Perangkat Keras *Remote Control*

   Perangkat keras yang digunakan dalam unit ini terdiri dari beberapa bagian yaitu:

   a. Rangkaian keypad
   b. Rangkaian pengendali utama *remote control*
   c. Rangkaian LCD (*Liquid Crystal Display*)
   d. Antarmuka modul *radio frequency* TLP 434A dengan mikrokontroler ATMega8

2. Perancangan Perangkat Keras Unit Pengolah Data

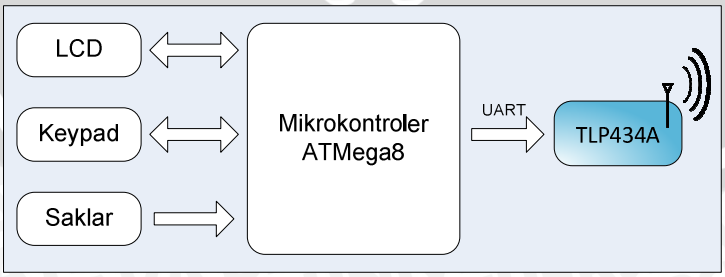
   Perangkat keras yang digunakan dalam unit pengolah data meliputi:

   a. Rangkaian pengendali utama unit pengolah data
   b. Rangkaian sistem pewaktu
   c. Antarmuka modul RLP 434A dengan mikrokontroler ATMega8

3. Perancangan Rangkaian *Driver Relay*

4.3.1 Perancangan Perangkat Keras Remote *Control*

Remote control berfungsi untuk mengontrol *on/off* peralatan listrik rumah dari jarak jauh. *Remote control* ini terdiri atas empat bagian utama, yaitu keypad, mikrokontroler, LCD dan modul TLP 434A. Diagram blok *remote control* ditunjukkan dalam Gambar 4.2.



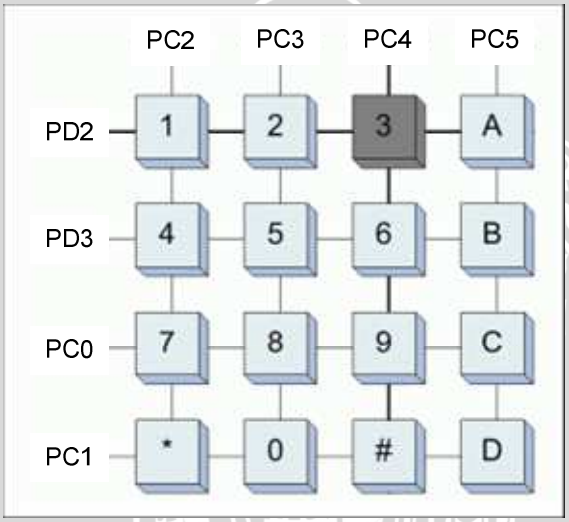
Gambar 4.2 Diagram blok *remote control*

Prinsip kerja *remote control* ini adalah pengguna dapat menyalakan atau mematikan peralatan listrik rumah dengan menekan *keypad* dan mengatur kombinasi

saklar. Ketika *keypad* ditekan, maka mikrokontroler 1 akan mendeteksi tombol mana yang ditekan dan mode yang digunakan, mikrokontroler kemudian akan mengolah data dari hasil penekanan tombol tersebut yang kemudian ditampilkan oleh LCD. Mikrokontroler akan mengirimkan data hasil pengolahan tersebut secara serial ke pusat pengolah data kemudian data serial ini masuk ke modul pemancar RF TLP434A.

4.3.1.1 Rangkaian *Keypad*

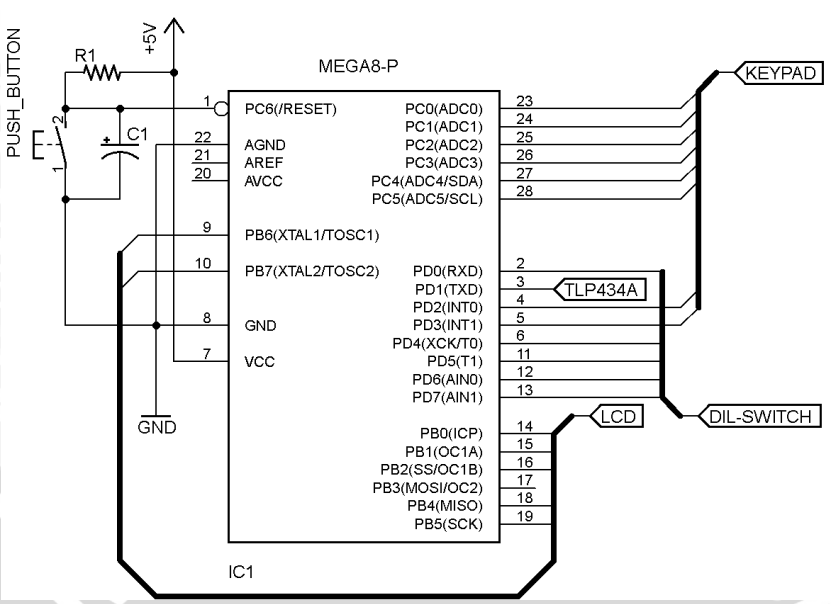
Dalam sistem ini keypad yang digunakan adalah keypad 4x4. Keypad digunakan untuk memberi masukan berupa data kepada mikrokontroler. Baris dari keypad terhubung dengan port mikrokontroller dikonfigurasikan sebagai jalur keluaran sedangkan kolom dari keypad terhubung dengan port mikrokontroler yang dikonfigurasikan sebagai jalur masukan. Rangkaian dari keypad dapat dilihat dalam Gambar 4.3.



Gambar 4.3 Rangkaian keypad 4x4

4.3.1.2 Rangkaian Pengendali Utama *Remote Control*

Mikrokontroler ATMega8 digunakan sebagai pengendali utama pada *remote control*. Proses yang dikendalikan meliputi pembacaan data dari *keypad* dan saklar, penampilan data di LCD dan pengiriman data melalui modul *radio frequency* TLP434A. Rangkaian pengendali utama *remote control* ditunjukkan dalam Gambar 4.4.



Gambar 4.4 Rangkaian pengendali utama *remote control*

Mikrokontroler ATmega8 mempunyai 23 jalur I/O yang dapat diprogram menjadi masukan atau keluaran. 23 jalur I/O ini dikelompokkan menjadi 3 kelompok, yaitu port B, C, dan D. Pada perancangan ini, pin-pin yang digunakan adalah sebagai berikut:

PB0 - PB7 : dihubungkan ke LCD

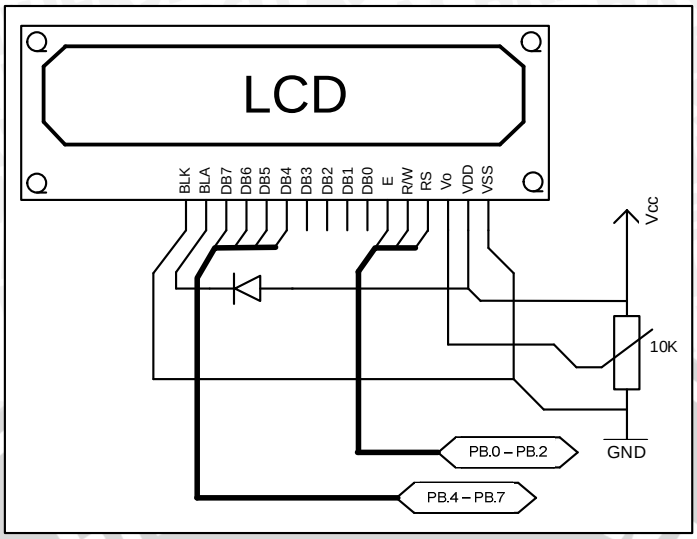
PC0 - PC5 & PD2 - PD3 : dihubungkan ke *keypad*

PD0 & PD4 - PD7 : dihubungkan ke saklar

PD1 : dihubungkan ke modul *radio frequency* TLP434A

4.3.1.3 Rangkaian LCD (*Liquid Crystal Display*)

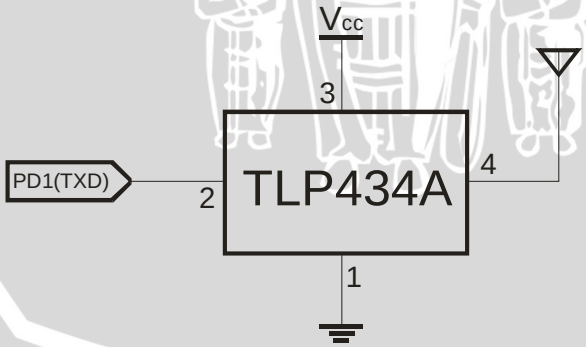
Dalam sistem ini LCD digunakan untuk menampilkan hasil pemrosesan data oleh *remote control*. Pin RS dari LCD terhubung ke PORTB.0 mikrokontroller, pin R/W dari LCD terhubung ke PORTB.1 dan pin *Enable* dari LCD terhubung ke PORTB.2 dari mikrokontroller. Sedangkan pin data dari LCD (PB4 sampai PB7) dihubungkan dengan PORTB nomor 4 sampai nomor 7 dari mikrokontroller. Gambar rangkaian dari LCD ditunjukkan dalam Gambar 4.5.



Gambar 4.5 Rangkaian LCD

4.3.1.4 Antarmuka Modul *Radio Frequency* TLP434A dengan Mikrokontroler ATMega8

Dalam sistem ini, modul *radio frequency* TLP434A digunakan untuk mengirimkan data dari *remote control* ke unit pengolah data secara *wireless* menggunakan *radio frequency* dengan frekuensi 433,92 MHz. Modul ini memiliki empat buah PIN, PIN nomor 1 terhubung dengan *ground*, PIN nomor 2 terhubung dengan PORTD.1 mikrokontroler, PIN nomor 3 terhubung ke V_{cc} dan PIN nomor 4 terhubung ke antena atau bisa tidak dihubungkan. Antarmuka modul TLP434A dengan mikrokontroler ditunjukkan dalam Gambar 4.6.

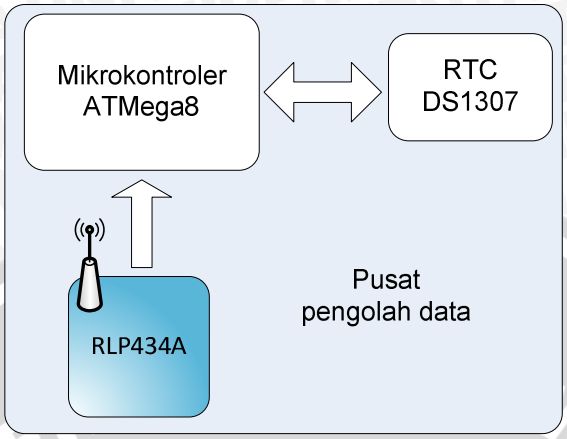


Gambar 4.6 Antarmuka modul RF TLP434A dengan mikrokontroler

4.3.2 Perancangan Perangkat Keras Unit Pengolah Data

Unit pengolah data berfungsi untuk mengolah data yang diterima dari *remote control* dan memberikan sinyal *trigger* ke rangkaian *driver relay* berdasarkan hasil

pengolahan data tersebut. Diagram blok unit pengolah data ditunjukkan dalam Gambar 4.7.



Gambar 4.7 Diagram blok unit pengolah data

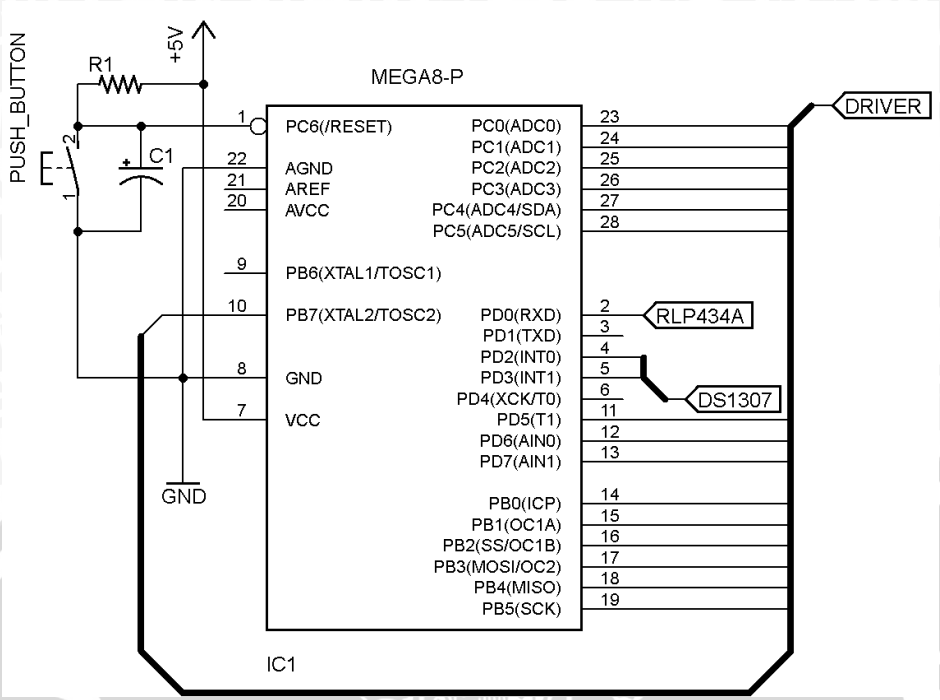
Dalam unit pengolah data ini, data yang dikirimkan oleh *remote control* akan diterima oleh modul RLP434A, kemudian data tersebut dikirimkan ke mikrokontroler ATMega8. Mikrokontroler kemudian mengolah data yang diterima dari modul RLP434A. Jika data yang diterima merupakan data mode *timer*/otomatis, maka mikrokontroler akan mengambil data waktu dari RTC DS1307.

4.3.2.1 Rangkaian Pengendali Utama Unit Pengolah Data

Mikrokontroler ATMega8 pada unit pengolah data ini digunakan sebagai pengendali utama beberapa proses meliputi penerimaan data melalui modul *radio frequency* RLP434A, pembacaan data waktu dari RTC DS1307 dan pemberian sinyal *trigger* ke rangkaian *driver relay*. Rangkaian pengendali utama unit pengolah data ditunjukkan dalam Gambar 4.8.

Mikrokontroler ATmega8 mempunyai 23 jalur I/O yang dapat diprogram menjadi masukan atau keluaran. 23 jalur I/O ini dikelompokkan menjadi 3 kelompok, yaitu port B, C, dan D. Pada perancangan ini, pin-pin yang digunakan adalah sebagai berikut:

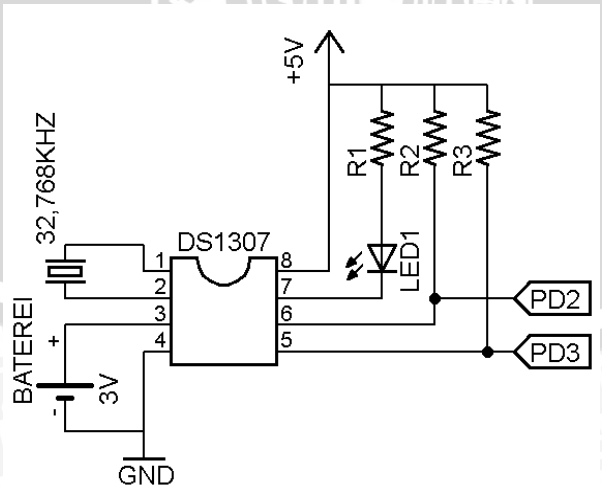
PC0 - PC5, PB0 - PB5, PB7 & PD5 – PD7 : dihubungkan ke *driver relay*
PD2 - PD3 : dihubungkan ke RTC DS1307
PD0 : dihubungkan ke modul *radio frequency* RLP434A



Gambar 4.8 Rangkaian pengendali utama unit pengolah data

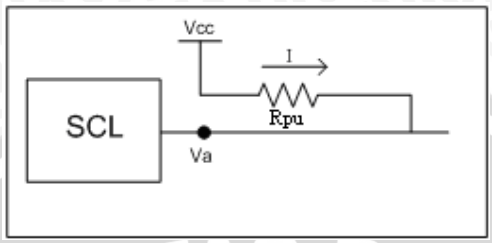
4.3.2.2 Rangkaian Sistem Pewaktu

Pewaktu yang digunakan dalam sistem ini adalah IC RTC *(Real Time Clock)* DS1307. RTC DS1307 berfungsi untuk merekam dan memberikan informasi waktu lengkap mulai informasi detik, menit, jam, tanggal, bulan hingga tahun. Catu utama yang diberikan sebesar 5 Volt. IC ini memerlukan Xtal eksternal sebesar 32,768 kHz. V_{bat} dihubungkan dengan catu daya baterai sebesar 3 V. Rangkaian RTC DS1307 ditunjukkan dalam Gambar 4.9.



Gambar 4.9 Rangkaian sistem pewaktu

Kedua pin yang digunakan sebagai jalur komunikasi, yaitu SCL dan SDA harus dihubungkan dengan resistor pull up untuk memastikan kondisi logika. Hal ini disebabkan karena kedua pin tersebut juga bersifat *open drain*. Gambar 4.10 menunjukkan analisis resistor pull up pada pin SCL.



Gambar 4.10 Analisis resistor *pull up* output RTC

Berdasarkan *SMBus and I2C Bus Design*, ada tiga hal yang harus dipertimbangkan saat menghitung nilai resistor pull up untuk bus I2C, yaitu V_{ih}, V_{il} dan *rise* dan *fall time*. Nilai V_{ih} dan V_{il} akan digunakan untuk menentukan ruang solusi, sedangkan *rise* dan *fall time* digunakan untuk menentukan nilai spesifik resistor *pull up* yang ada dalam ruang solusi. Langkah pertama yaitu menentukan nilai maksimum resistor *pull up* dengan persamaan:

$$Rp_{max} = (Vcc_{min} - (Vih_{mm} + NM_{min})) / Iih_{max} \quad (1)$$

Keterangan:

Rp_{max} : nilai resistor *pull up* maksimum

Vcc_{min} : nilai V_{cc} minimum dimana resistor *pull up* akan dihubungkan

Vih_{mm} : nilai Vih_{min} dari setiap agen bus (dipilih yang paling besar) saat Vcc_{min}

NM_{min} : *noise margin*

Iih_{max} : nilai Iih *slave* maksimum

Berdasarkan *datasheet* diketahui bahwa Vih_{mm} bernilai 0,7 Vcc_{min} atau bernilai 3,15 V, nilai NM_{min} yaitu 0,2, dan nilai Iih_{max} = 1 µA. Maka akan didapatkan nilai maksimum resistor *pull up* sebesar :

$Rp_{max} = (Vcc_{min} - (Vih_{mm} + NM_{min})) / Iih_{max}$

$= (4,5 - (3,15 + 0,2)) / 1\mu A$

$= 1,15 / 1\mu A$

$= 1,15\ M\Omega$

Langkah kedua yaitu menentukan nilai minimum resistor *pull up* dengan persamaan:

$$Rp_{min} = (Vcc_{max} - Vol_{min}) / Iol_{max} \quad (2)$$

Keterangan:

Rp_{min} : nilai resistor *pull up* minimum

Vcc_{max} : nilai V_{cc} maksimum dimana resistor *pull up* akan dihubungkan

Vol_{min} : nilai Vol mikrokontroler minimum

Iol_{max} : nilai Iol mikrokontroler maksimum saat nilai Vol mikrokontroler maksimum

Berdasarkan *datasheet* diketahui bahwa Vol_{min} bernilai 0 V, dan nilai Iol_{max} = 3 mA. Maka akan didapatkan nilai minimum resistor *pull up* sebesar :

$Rp_{min} = (Vcc_{max} – Vol_{min}) / Iol_{max}$

$= (5,5 – 0) / 3$ mA

$= 1,83$ kΩ

Langkah terakhir yaitu menentukan nilai spesifik resistor *pull up* yang ada dalam ruang solusi antara nilai maksimum dan nilai minimum. Persamaan untuk menentukan nilai spesifik resistor *pull up* ini adalah:

$Rp = - tr / (C. \ln [(Vih_{mm} + NM_{min} - Vcc_{min}) / (Vo – Vcc_{min})])$ (3)

Keterangan:

tr : nilai *rise time* maksimum setelah dikurangi margin (50 – 100nS)

C : nilai total kapasitansi bus

Vih_{mm} : nilai Vih_{min} dari setiap agen bus (dipilih yang paling besar) saat Vcc_{max}

Vo : tegangan minimum saat logika 0

Berdasarkan datasheet diketahui bahwa nilai tr_{max} adalah 300 ns, nilai C ($C_{I(MK)}$ + $CI_{(RTC)}$) adalah 20 pF, Vih_{mm} bernilai 0,7 Vcc_{max} = 3,85 V, Vcc_{min} bernilai 4,5 V dan Vo bernilai 0 V. Maka akan didapatkan nilai spesifik resistor *pull up* sebesar :

$Rp = - tr_{max} / (C. \ln [(Vih_{mm} + NM_{min} - Vcc_{min}) / (Vo – Vcc_{min})])$

$= - 220 . 10^{-9} / (20 . 10^{-12} \ln [(3,85 + 0,2 – 4,5) / (0 – 4,5)])$

$= - 220 . 10^{-9} / (20 . 10^{-12} . \ln [0,255])$

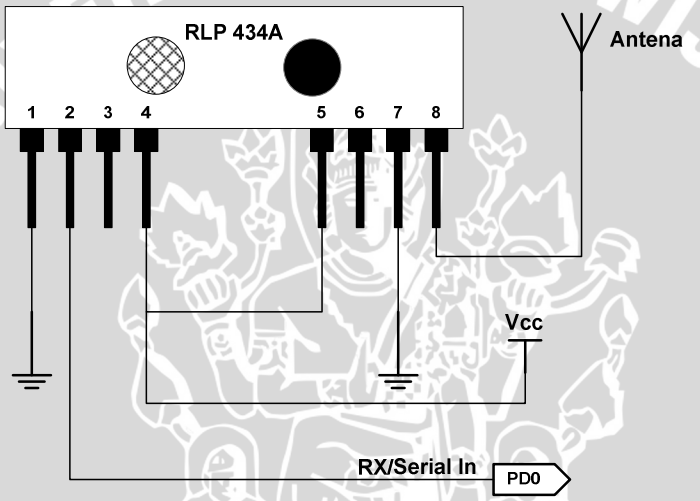
$= - 220 . 10^{-9} / - 46.052 . 10^{-12}$

$= 4,77$ kΩ

Dalam perancangan ini digunakan resistor *pull up* bernilai 4,7 kΩ karena lebih kecil dari 1,15 MΩ dan lebih besar dari 1,83 kΩ dan mendekati 4,77 kΩ dan 4,7 kΩ umum digunakan dalam komunikasi I2C.

4.3.2.3 Antarmuka Modul RF RLP434A dengan Mikrokontroler ATMega8

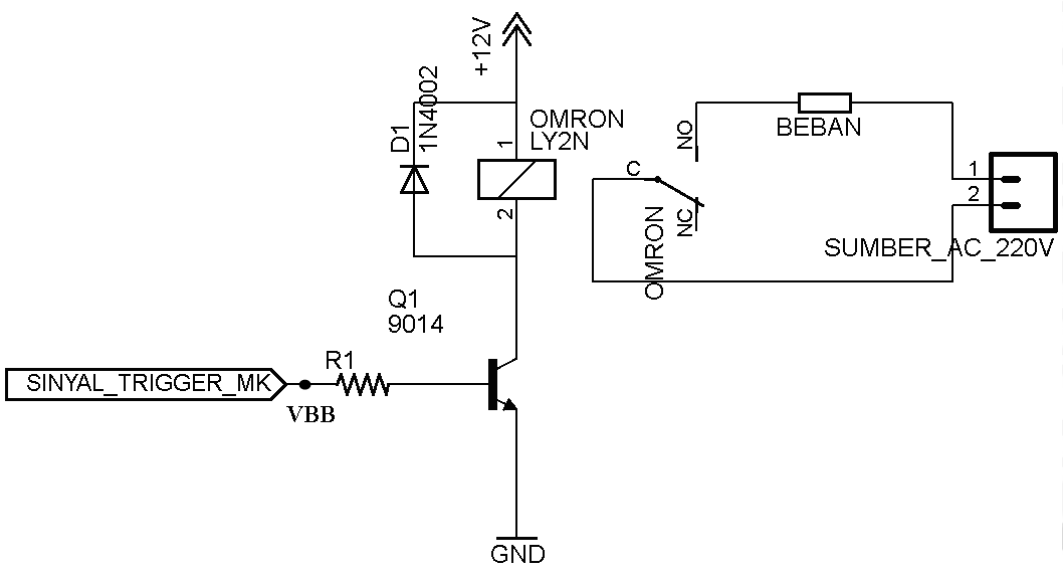
Dalam sistem ini, modul *radio frequency* RLP434A digunakan untuk menerima data yang dikirimkan *remote control* ke unit pengolah data secara *wireless* menggunakan *radio frequency* dengan frekuensi 433,92 MHz. Modul ini memiliki delapan buah PIN, PIN nomor 1 terhubung dengan *ground*, PIN nomor 2 terhubung dengan PORTD.0 mikrokontroler, PIN nomor 3 tidak dihubungkan, PIN nomor 4 dan PIN nomor 5 terhubung ke $V_{cc,}$ PIN nomor 6 terhubung dengan ground atau bisa tidak dihubungkan, PIN nomor 7 terhubung dengan ground dan PIN nomor 8 terhubung dengan antena atau bisa tidak dihubungkan. Antarmuka modul RLP434A dengan mikrokontroler ditunjukkan dalam Gambar 4.11.



Gambar 4.11 Antarmuka modul RLP434A dengan mikrokontroler ATMega 8

4.3.3 Perancangan Rangkaian *Driver Relay*

Rangkaian *driver relay* dalam sistem ini berfungsi untuk menghubungkan mikrokontroler dengan peralatan listrik rumah. Gambar 4.12 menunjukkan rangkaian *driver relay.* Sebelum dihubungkan dengan *relay,* keluaran mikrokontroler dihubungkan terlebih dahulu dengan *driver* yang berupa transistor.



Gambar 4.12 Rangkaian *driver relay*

Berdasarkan *datasheet* transistor 9014 diketahui bahwa β_{min} bernilai 60 dan berdasarkan *datasheet relay* OMRON LY2N arus yang dibutuhkan oleh *relay* sebesar 75 mA (I_C). Berdasarkan arus yang dibutuhkan *relay* tersebut, maka nilai I_B adalah:

$I_C = I_B \cdot \beta_{min}$

$I_B = I_C / \beta_{min}$

$= 75 \text{ mA} / 60$

$= 1{,}25 \text{ mA}$

Analisis perhitungan R_1 adalah sebagai berikut:

$V_{BB} = I_B \cdot R_1 + V_{BE}$

$5 \text{ V} = 1{,}25 \text{ mA} \cdot R_1 + 1 \text{ V}$

$R_1 = 4 \text{ V} / 1{,}25 \text{ mA}$

$= 3{,}2 \text{ k}\Omega$

Dari perhitungan di atas didapatkan R_1 sebesar 3,2 kΩ sehingga dalam perancangan ini digunakan R_1 bernilai 3 kΩ.

4.4 Perancangan Format Paket Data

Agar proses komunikasi dapat berjalan sesuai dengan perancangan dan data yang diterima dapat lebih mudah untuk dikenali dan diolah, maka digunakan format paket data khusus dalam komunikasi di antara *remote control* dan unit pengolah data. Format paket data terdiri atas dua bagian, yaitu kepala dan data. Bagian kepala berfungsi sebagai penanda atau pengenal paket data yang dikirimkan oleh *remote*

control. Bagian kepala ini berukuran tiga byte, sedangkan bagian data berukuran satu byte. Format paket data yang dirancang ditunjukkan dalam Gambar 4.15.

Kepala_1 (1 byte)	Kepala_2 (1 byte)	Kepala_3 (1 byte)	Data (1 byte)
85	90	95	-

Gambar 4.15 Format paket data komunikasi antara *remote* dan unit pengolah data

Penjelasan dari format paket data diatas adalah sebagai berikut:

Kepala_1 : terdiri atas 1 byte data yang diletakkan diawal sebagai penanda agar paket data bisa dikenali dan menandakan awal dari proses pengiriman data.

Kepala_2 : terdiri atas 1 byte data yang diletakkan setelah kepala_1 sebagai penanda agar paket data bisa dikenali.

Kepala_3 : terdiri atas 1 byte data yang diletakkan setelah kepala_2 sebagai penanda agar paket data bisa dikenali.

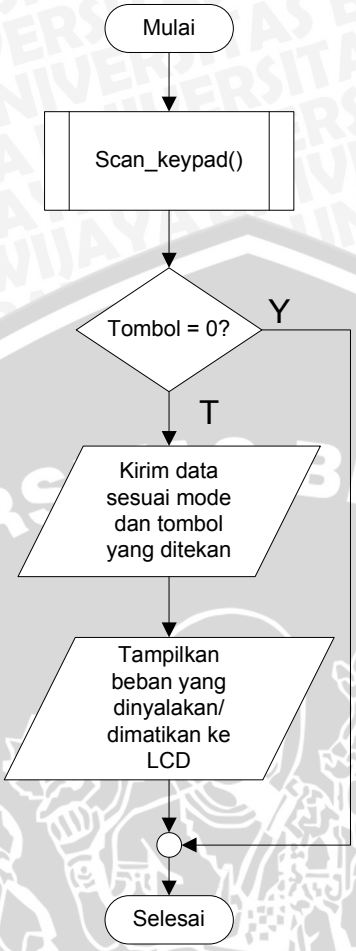
Data : terdiri atas 1 byte data yang berisi nilai variabel tombol dari *keypad* yang ditekan dan kombinasi saklar.

4.5 Perancangan Perangkat Lunak (*Software*)

Untuk mengendalikan perangkat keras dibutuhkan perangkat lunak mikrokontroler. Perangkat lunak dalam sistem ini terdiri atas dua bagian, yaitu perangkat lunak pada *remote control* dan perangkat lunak pada unit pengolah data.

4.5.1 Perancangan Perangkat Lunak pada *Remote*

Perangkat lunak pada *remote control* berfungsi untuk menjalankan proses yang meliputi mendeteksi penekanan tombol *keypad* dan kombinasi saklar, mengolah data hasil penekanan tombol dan kombinasi saklar kemudian mengirimkan data hasil pengolahan tersebut ke pusat pengolah data melalui modul pemancar RF TLP434A. Diagram alir rutin pada *remote control* ditunjukkan dalam Gambar 4.13.



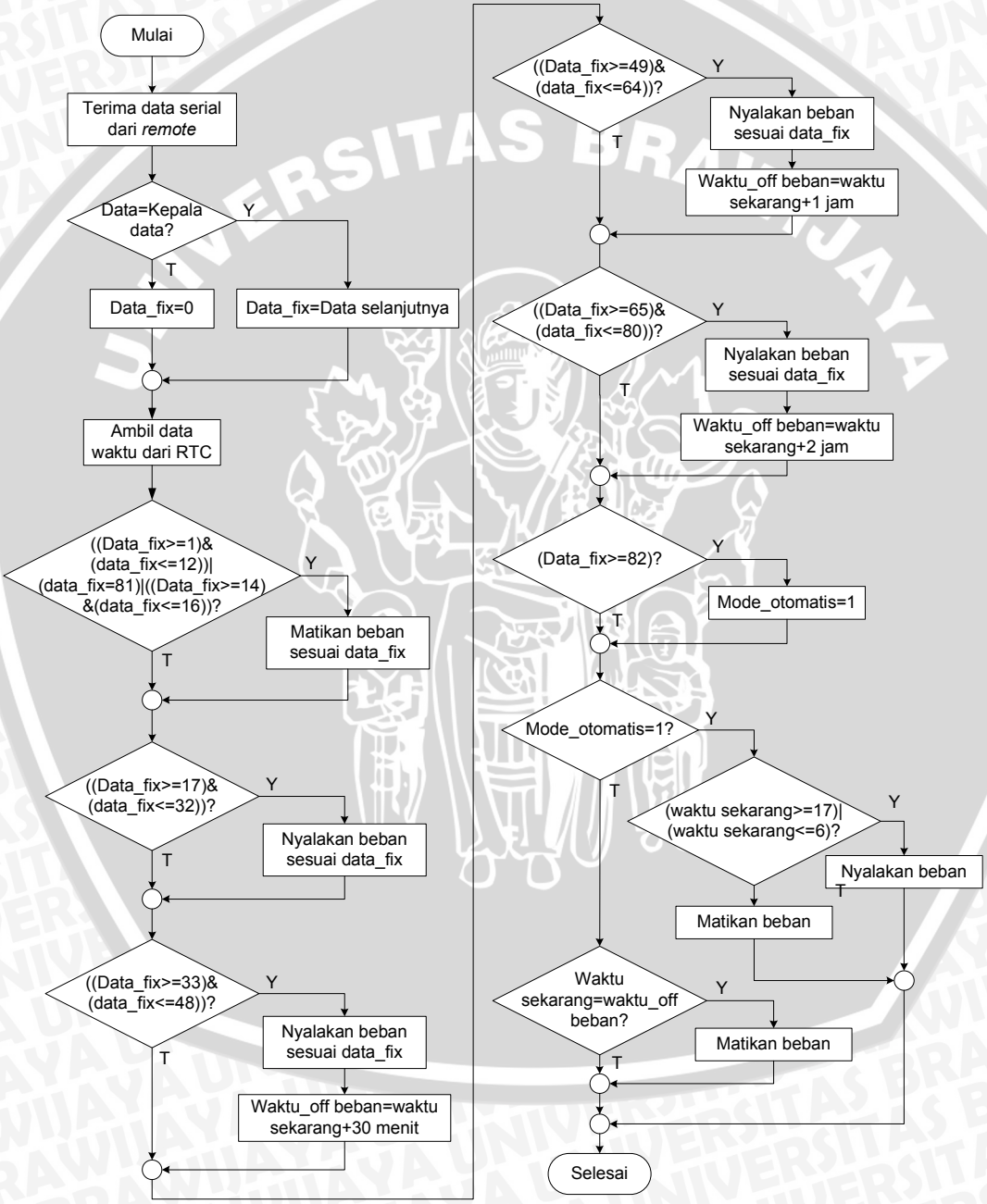
Gambar 4.13 Diagram alir rutin pada *remote control*

Ketika *remote control* diaktifkan, mikrokontroler akan melakukan *scan keypad*. *Scan keypad* ini berfungsi untuk mendeteksi penekanan tombol dan juga mengecek kombinasi saklar untuk mengetahui mode yang dipilih. Saat tidak terjadi penekanan tombol pada *keypad*, variabel tombol bernilai 0 sedangkan apabila terjadi penekanan tombol pada *keypad*, maka variabel tombol akan bernilai sesuai dengan tombol yang ditekan dan mode yang digunakan.

Setelah melakukan scan *keypad*, maka mikrokontroler akan mengecek apakah variabel tombol sama dengan 0. Jika tidak bernilai 0, maka mikrokontroler akan mengirimkan data sesuai mode dan tombol yang ditekan dan menampilkan mode yang dipilih dan beban yang akan dinyalakan atau dimatikan ke LCD.

4.5.2 Perancangan Perangkat Lunak pada Unit Pengolah Data

Perangkat lunak pada unit pengolah data berfungsi untuk menjalankan proses yang meliputi menerima data serial yang dikirimkan *remote control*, mengambil data waktu dari RTC dan menberikan tegangan *trigger* pada rangkaian *driver relay*. Diagram alir rutin pada unit pengolah data ditunjukkan dalam Gambar 4.14.



Gambar 4.14 Diagram alir rutin pada unit pengolah data

Unit pengolah data menerima data serial yang dikirimkan oleh *remote control*. Data serial yang diterima tersebut akan dicek apakah format paket data yang diterima sesuai dengan format paket data yang dirancang atau tidak. Jika sesuai, maka data tersebut disimpan dalam variabel data_fix, tetapi jika tidak maka variabel data_fix berisi 0. Setelah itu rutin unit pengolah data akan mengambil data waktu dari RTC untuk mengetahui waktu saat data diterima atau waktu sekarang.

Setelah itu mikrokontroler akan mengecek isi dari data_fix untuk mengetahui mode yang digunakan dan beban yang ingin dinyalakan atau dimatikan. Jika data_fix bernilai antara 1 sampai dengan 12 atau bernilai antara 14 sampai dengan 16 atau 81, maka beban dimatikan sesuai dengan nilai data_fix dalam perancangan.

Jika tidak sesuai, maka data_fix dicek lagi. Jika data_fix bernilai antara 17 sampai dengan 32, maka beban dinyalakan sesuai dengan data_fix. Jika tidak sesuai, maka data_fix dicek lagi. Jika data_fix bernilai antara 33 sampai dengan 48, maka beban dinyalakan sesuai dengan data_fix. Kemudian waktu sekarang yang didapat dari RTC ditambah 30 menit dan dimasukkan ke variabel waktu_off beban. Jika tidak sesuai, maka data_fix dicek lagi. Jika data_fix bernilai antara 49 sampai dengan 64, maka beban dinyalakan sesuai dengan data_fix. Kemudian waktu sekarang yang didapat dari RTC ditambah 1 jam dan dimasukkan ke variabel waktu_off beban. Jika tidak sesuai, maka data_fix dicek lagi. Jika data_fix bernilai antara 65 sampai dengan 80, maka beban dinyalakan sesuai dengan data_fix. Kemudian waktu sekarang yang didapat dari RTC ditambah 2 jam dan dimasukkan ke variabel waktu_off beban.

Jika tidak sesuai, maka data_fix dicek lagi. Jika data_fix bernilai 82, maka variabel mode_otomatis bernilai 1. Kemudian rutin melakukan pengecekan lagi. Jika variabel mode_otomatis tidak bernilai 1, maka rutin melakukan pengecekan lagi apakah waktu sekarang sama dengan waktu_off beban atau tidak. Jika waktu sekarang sama dengan waktu_off beban, maka beban dimatikan. Tetapi jika mode_otomatis sama dengan 1 maka dilakukan pengecekan lagi apakah waktu sekarang lebih besar dari 17 atau kurang dari 6. Jika sesuai maka beban dinyalakan, jika tidak maka beban dimatikan.

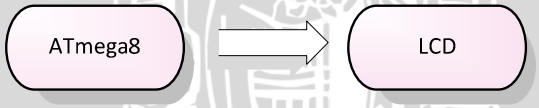
BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui dan menganalisis apakah sistem telah bekerja sesuai perancangan. Pengujian dilakukan per diagram blok kemudian secara keseluruhan. Adapun pengujian yang perlu dilakukan sebagai berikut:

a) Pengujian LCD
b) Pengujian antarmuka *keypad* dengan mikrokontroler
c) Pengujian *remote control*
d) Pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A
e) Pengujian sistem pewaktu RTC DS1307
f) Pengujian rangkaian *driver relay*
g) Pengujian keseluruhan sistem

5.1 Pengujian Tampilan LCD

Pengujian modul LCD ini bertujuan untuk mengetahui keberhasilan LCD menampilkan tulisan sesuai dengan karakter yang dikirimkan oleh mikrokontroler. Pengujian modul LCD dilakukan dengan menghubungkan LCD dengan mikrokontroler yang sudah berisi perangkat lunak untuk menampilkan tulisan tertentu. Diagram blok pengujian modul LCD ini ditunjukkan dalam Gambar 5.1.



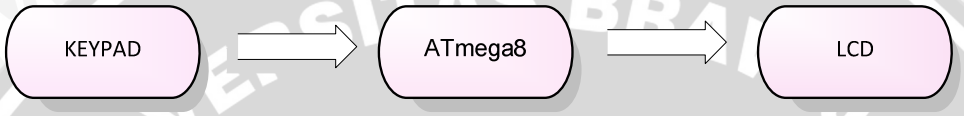
Gambar 5.1 Diagram blok pengujian tampilan LCD

Dalam pengujian ini, LCD dapat menampilkan tulisan ”Pengujian” pada baris pertama dan “LCD” pada baris kedua. Tampilan hasil pengujian modul LCD ditunjukkan dalam Gambar 5.2. Dari hasil pengujian dapat disimpulkan bahwa rangkaian LCD dapat menampilkan tulisan sesuai dengan karakter yang dikirimkan oleh mikrokontroler.

Gambar 5.2 Tampilan hasil pengujian tampilan LCD

5.2 Pengujian Antarmuka *Keypad* dengan Mikrokontroler

Pengujian antarmuka *keypad* dengan mikrokontroler bertujuan untuk mengetahui apakah perangkat keras *keypad* dan perangkat lunak dalam mikrokontroler dapat bekerja sesuai dengan perancangan. Pengujian dilakukan dengan cara menghubungkan keypad dengan mikrokontroler dan menghubungkan mikrokontroler dengan LCD. Dalam pengujian ini mikrokontroler berfungsi untuk mendeteksi tombol *keypad* yang ditekan dan memerintahkan LCD untuk menampilkan karakter tombol yang ditekan. Diagram blok pengujian antarmuka *keypad* dengan mikrokontroler ini ditunjukkan dalam Gambar 5.3.



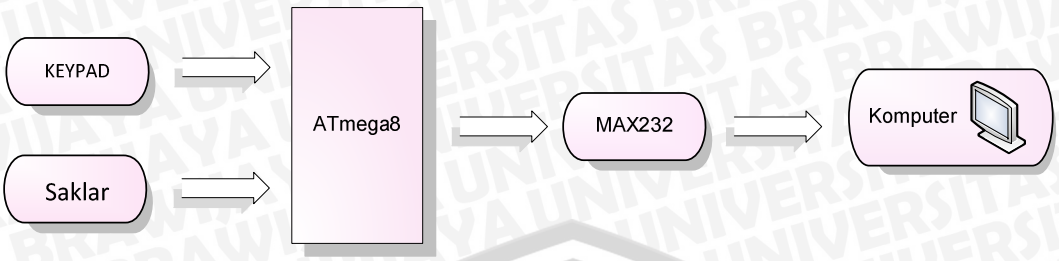
Gambar 5.3 Diagram blok pengujian antarmuka *keypad* dengan LCD

Dalam pengujian ini, semua tombol pada *keypad* ditekan dengan urutan “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “0”, “*”, “#”, “A”, “B”, “C”, “D” kemudian sebaliknya. Tampilan hasil pengujian rangkaian *keypad* ditunjukkan dalam Gambar 5.4. Hasil pengujian menunjukkan bahwa LCD dapat menampilkan karakter sesuai dengan tombol yang ditekan, sehingga dapat disimpulkan bahwa rangkaian keypad dan perangkat lunak yang telah dibuat dapat bekerja sesuai dengan perancangan.

Gambar 5.4 Tampilan hasil pengujian antarmuka *keypad* dengan mikrokontroler

5.3 Pengujian *Remote Control*

Pengujian ini bertujuan untuk mengetahui apakah *remote control* bekerja sesuai dengan perancangan. Pengujian *remote control* dilakukan dengan cara menghubungkan *keypad* dan saklar dengan mikrokontroler. Diagram blok pengujian *remote control* ini ditunjukkan dalam Gambar 5.5. Dalam pengujian ini mikrokontroler telah diprogram sesuai dengan perancangan yang terdiri dari mode otomatis, mode mati, mode hidup, mode hidup 30 menit, mode hidup 1 jam dan mode hidup 2 jam. Saklar digunakan untuk memilih mode yang akan dijalankan. Mikrokontroler dihubungkan dengan MAX232 dan MAX232 dihubungkan dengan komputer. Hasil pengujian dapat diamati pada *Hyperterminal* program *Code Vision AVR.*

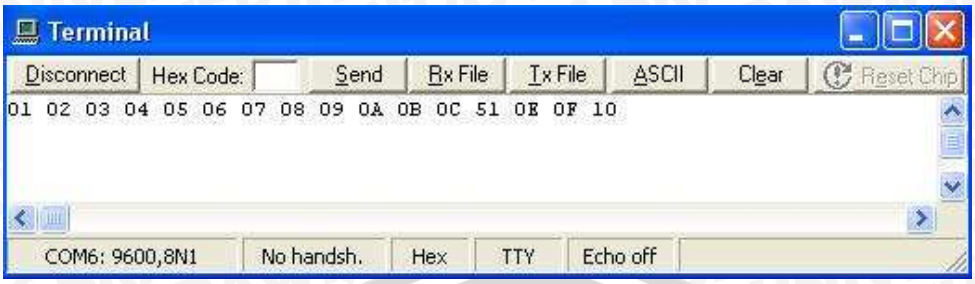


Gambar 5.5 Diagram blok pengujian *remote control*

Prosedur pengujiannya yaitu saklar diatur sesuai dengan mode yang diinginkan kemudian semua tombol pada *keypad* ditekan dengan urutan “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”, “0”, “*”, “#”, “A”, “B”, “C”, “D”. Pada mode otomatis setelah mengatur kombinasi saklar sesuai dengan perancangan, hanya tombol “*” pada *keypad* yang harus ditekan. Untuk mode otomatis data yang dikirim oleh mikrokontroler yaitu 82, sedangkan data yang diterima oleh tampilan komputer adalah 52. Data yang dikirim merupakan data desimal dan data yang diterima oleh tampilan komputer merupakan data heksadesimal. Hasil pengujian mode otomatis ini ditunjukkan dalam Gambar 5.6. Dari hasil ini dapat disimpulkan bahwa pada mode otomatis, *remote control* bekerja sesuai dengan perancangan.

Gambar 5.6 Tampilan hasil pengujian *remote control* mode otomatis

Hasil pengujian mode mati ditunjukkan dalam Gambar 5.7, sedangkan Tabel 5.1 menunjukkan perbandingan antara data yang dikirim mikrokontroler dengan data yang diterima oleh tampilan komputer. Dari hasil pengujian ini dapat disimpulkan bahwa pada mode mati, *remote control* bekerja sesuai dengan perancangan, yaitu data yang dikirim sama dengan data yang diterima oleh tampilan komputer.

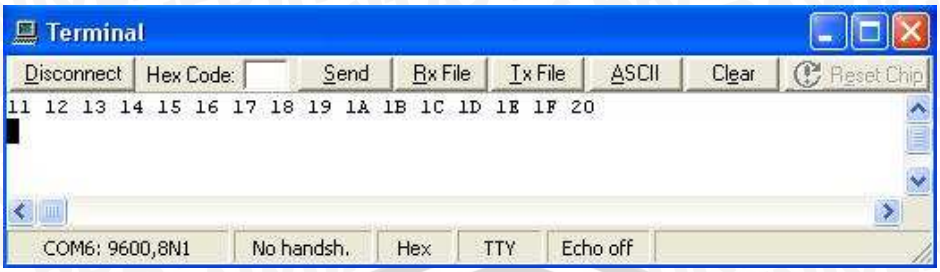


Gambar 5.7 Tampilan hasil pengujian *remote control* mode mati

Tabel 5.1 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode mati

Keypad	Data yang Dikirim	Data yang Diterima	Keterangan
1	1	01 H	Benar
2	2	02 H	Benar
3	3	03 H	Benar
4	4	04 H	Benar
5	5	05 H	Benar
6	6	06 H	Benar
7	7	07 H	Benar
8	8	08 H	Benar
9	9	09 H	Benar
0	10	0A H	Benar
*	11	0B H	Benar
#	12	0C H	Benar
A	81	51 H	Benar
B	14	0E H	Benar
C	15	0F H	Benar
D	16	10 H	Benar

Hasil pengujian mode hidup ditunjukkan dalam Gambar 5.8, sedangkan Tabel 5.2 menunjukkan perbandingan antara data yang dikirim mikrokontroler dengan data yang diterima oleh tampilan komputer. Dari hasil pengujian ini dapat disimpulkan bahwa pada mode mati, *remote control* bekerja sesuai dengan perancangan, yaitu data yang dikirim sama dengan data yang diterima oleh tampilan komputer.

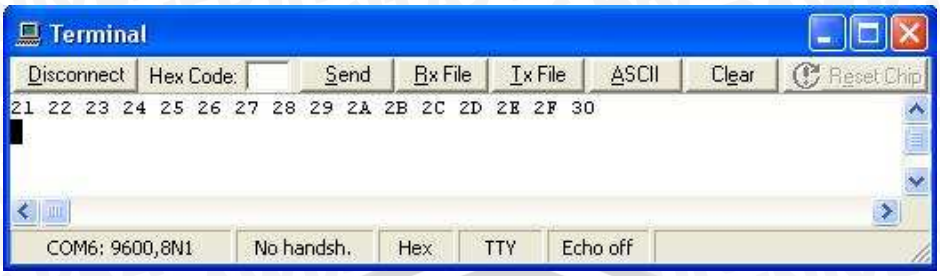


Gambar 5.8 Tampilan hasil pengujian *remote control* mode hidup

Tabel 5.2 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode hidup

Keypad	Data yang Dikirim	Data yang Diterima	Keterangan
1	17	11 H	Benar
2	18	12 H	Benar
3	19	13 H	Benar
4	20	14 H	Benar
5	21	15 H	Benar
6	22	16 H	Benar
7	23	17 H	Benar
8	24	18 H	Benar
9	25	19 H	Benar
0	26	1A H	Benar
*	27	1B H	Benar
#	28	1C H	Benar
A	29	1D H	Benar
B	30	1E H	Benar
C	31	1F H	Benar
D	32	20 H	Benar

Hasil pengujian mode hidup 30 menit ditunjukkan dalam Gambar 5.9, sedangkan Tabel 5.3 menunjukkan perbandingan antara data yang dikirim mikrokontroler dengan data yang diterima oleh tampilan komputer. Dari hasil pengujian ini dapat disimpulkan bahwa pada mode mati, *remote control* bekerja sesuai dengan perancangan, yaitu data yang dikirim sama dengan data yang diterima oleh tampilan komputer.

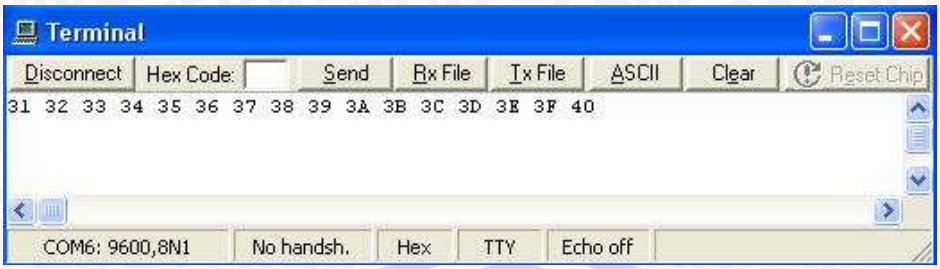


Gambar 5.9 Tampilan hasil pengujian *remote control* mode hidup 30 menit

Tabel 5.3 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode hidup 30 menit

Keypad	Data yang Dikirim	Data yang Diterima	Keterangan
1	33	21 H	Benar
2	34	22 H	Benar
3	35	23 H	Benar
4	36	24 H	Benar
5	37	25 H	Benar
6	38	26 H	Benar
7	39	27 H	Benar
8	40	28 H	Benar
9	41	29 H	Benar
0	42	2A H	Benar
*	43	2B H	Benar
#	44	2C H	Benar
A	45	2D H	Benar
B	46	2E H	Benar
C	47	2F H	Benar
D	48	30 H	Benar

Hasil pengujian mode hidup 1 jam ditunjukkan dalam Gambar 5.10, sedangkan Tabel 5.4 menunjukkan perbandingan antara data yang dikirim mikrokontroler dengan data yang diterima oleh tampilan komputer. Dari hasil pengujian ini dapat disimpulkan bahwa pada mode mati, *remote control* bekerja sesuai dengan perancangan, yaitu data yang dikirim sama dengan data yang diterima oleh tampilan komputer.

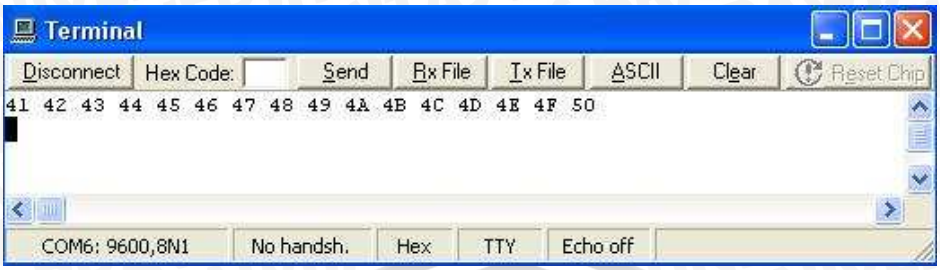


Gambar 5.10 Tampilan hasil pengujian *remote control* mode hidup 1 jam

Tabel 5.4 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode hidup 1 jam

Keypad	Data yang Dikirim	Data yang Diterima	Keterangan
1	49	31 H	Benar
2	50	32 H	Benar
3	51	33 H	Benar
4	52	34 H	Benar
5	53	35 H	Benar
6	54	36 H	Benar
7	55	37 H	Benar
8	56	38 H	Benar
9	57	39 H	Benar
0	58	3A H	Benar
*	59	3B H	Benar
#	60	3C H	Benar
A	61	3D H	Benar
B	62	3E H	Benar
C	63	3F H	Benar
D	64	40 H	Benar

Hasil pengujian mode hidup 2 jam ditunjukkan dalam Gambar 5.11, sedangkan Tabel 5.5 menunjukkan perbandingan antara data yang dikirim mikrokontroler dengan data yang diterima oleh tampilan komputer. Dari hasil pengujian ini dapat disimpulkan bahwa pada mode mati, *remote control* bekerja sesuai dengan perancangan, yaitu data yang dikirim sama dengan data yang diterima oleh tampilan komputer.



Gambar 5.11 Tampilan hasil pengujian *remote control* mode hidup 2 jam

Tabel 5.5 Perbandingan data yang dikirim mikrokontroler dan data yang diterima komputer mode hidup 2 jam

Keypad	Data yang Dikirim	Data yang Diterima	Keterangan
1	65	41 H	Benar
2	66	42 H	Benar
3	67	43 H	Benar
4	68	44 H	Benar
5	69	45 H	Benar
6	70	46 H	Benar
7	71	47 H	Benar
8	72	48 H	Benar
9	73	49 H	Benar
0	74	4A H	Benar
*	75	4B H	Benar
#	76	4C H	Benar
A	77	4D H	Benar
B	78	4E H	Benar
C	79	4F H	Benar
D	80	50 H	Benar

Dari hasil pengujian semua mode tersebut, dapat disimpulkan bahwa *remote control* dapat bekerja pada semua mode sesuai dengan perancangan.

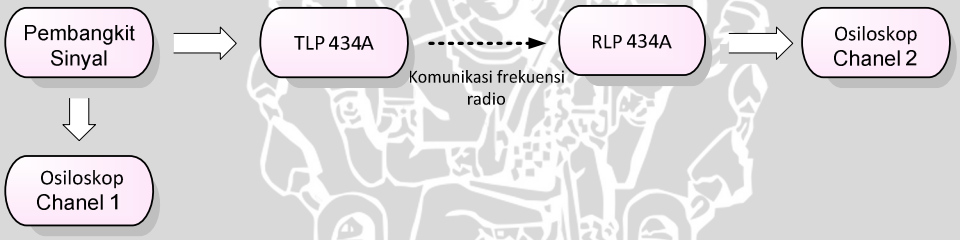
5.4 Pengujian Rangkaian Pemancar TLP 434A dan Penerima RLP 434A

Pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A terdiri atas tiga pengujian, yaitu pengujian bentuk sinyal dan frekuensi, pengujian kecepatan transfer data (*baud rate*), dan pengujian komunikasi data dan jarak jangkau transmisi radio frekuensi.

5.4.1 Pengujian bentuk sinyal dan frekuensi modul RF

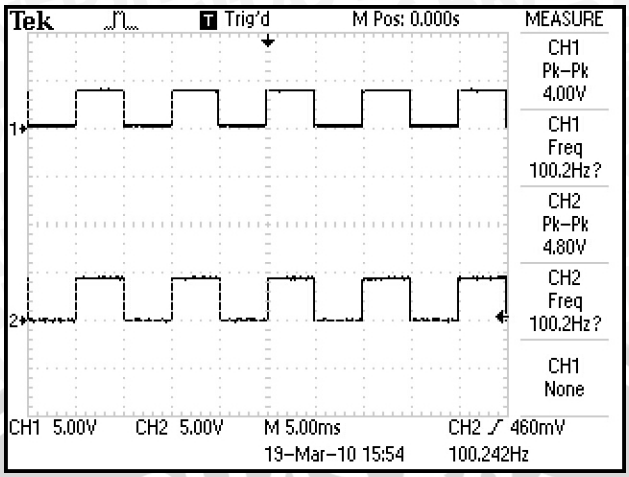
Pengujian bentuk sinyal rangkaian pemancar TLP 434A dan penerima RLP 434A bertujuan untuk mengetahui apakah data yang diterima oleh penerima RLP 434A sama dengan data yang dikirim pemancar TLP 434A. Sedangkan pengujian frekuensi bertujuan untuk mengetahui frekuensi informasi maksimum yang dapat dikirimkan melalui pemancar TLP 434A dan penerima RLP 434A.

Pengujian ini dilakukan dengan cara menghubungkan pembangkit sinyal dengan pemancar TLP 434A dan osiloskop. Pembangkit sinyal akan mengirimkan sinyal kotak dengan frekuensi tertentu (ditampilkan pada osiloskop *channel* 1) ke pemancar TLP 434A. Penerima RLP 434A dihubungkan dengan osiloskop *channel* 2 untuk melihat sinyal yang diterima oleh penerima RLP 434A. Kemudian membandingkan tampilan pada osiloskop antara *channel* 1 dan *channel* 2. Prosedur tersebut dilakukan untuk beberapa frekuensi yang berbeda-beda. Diagram blok pengujian rangkaian pemancar 434A dan penerima 434A ditunjukkan dalam Gambar 5.12.



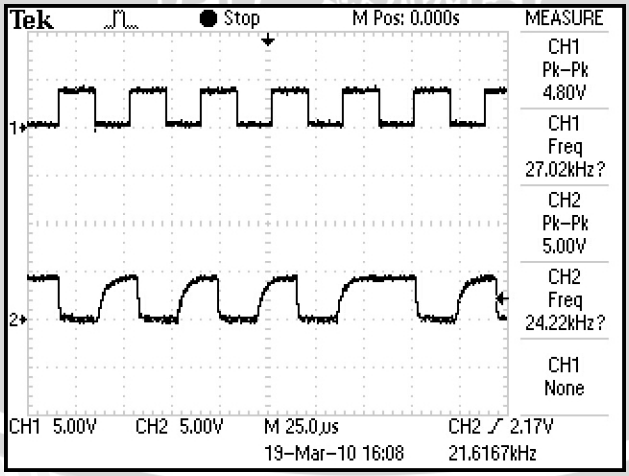
Gambar 5.12 Diagran blok pengujian bentuk sinyal dan frekuensi modul pemancar TLP 434A dan penerima RLP 434A

Gambar 5.13 menunjukkan hasil pengujian bentuk sinyal dan frekuensi modul RF dengan frekuensi 100 Hz. Sinyal kotak pada *chanel* 1 osiloskop merupakan sinyal yang dikirimkan pembangkit sinyal melalui pemancar TLP 434A (bagian atas), sedangkan sinyal kotak pada *channel* 2 merupakan sinyal yang diterima oleh penerima RLP 434A (bagian bawah). Dengan membandingkan tampilan pada osiloskop *channel* 1 dan *channel* 2 dapat disimpulkan bahwa pada frekuensi 100 Hz data yang dikirimkan TLP 434A dan data yang diterima RLP 434A sama.



Gambar 5.13 Tampilan osiloskop hasil pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A untuk frekuensi 100 Hz

Gambar 5.14 menunjukkan hasil pengujian bentuk sinyal dan frekuensi rangkaian modul RF dengan frekuensi 27 kHz. Sinyal kotak pada *channel* 1 osiloskop merupakan sinyal yang dikirimkan pembangkit sinyal ke pemancar TLP 434A (bagian atas), sedangkan sinyal kotak pada *channel* 2 merupakan sinyal yang diterima oleh penerima RLP 434A (bagian bawah). Dengan membandingkan tampilan pada osiloskop *channel* 1 dan *channel* 2 dapat disimpulkan bahwa pada frekuensi 27 kHz data yang dikirimkan TLP 434A dan data yang diterima RLP 434A berbeda, atau dengan kata lain pada frekuensi 27 kHz data yang diterima tidak valid.



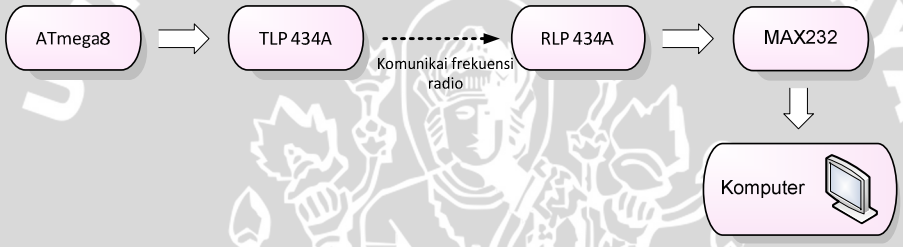
Gambar 5.14 Tampilan osiloskop hasil pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A untuk frekuensi 27 kHz

Dari hasil pengujian sinyal dan frekuensi ini dapat disimpulkan bahwa sinyal digital dari pembangkit sinyal yang dihubungkan dengan pemancar TLP434A (*channel* 1, bagian atas) dapat diterima dengan baik oleh modul penerima RLP434A (*channel* 2

bagian bawah) dan frekuensi informasi maksimum yang dapat dikirimkan melalui pemancar TLP 434A dan penerima RLP 434A adalah 20 kHz.

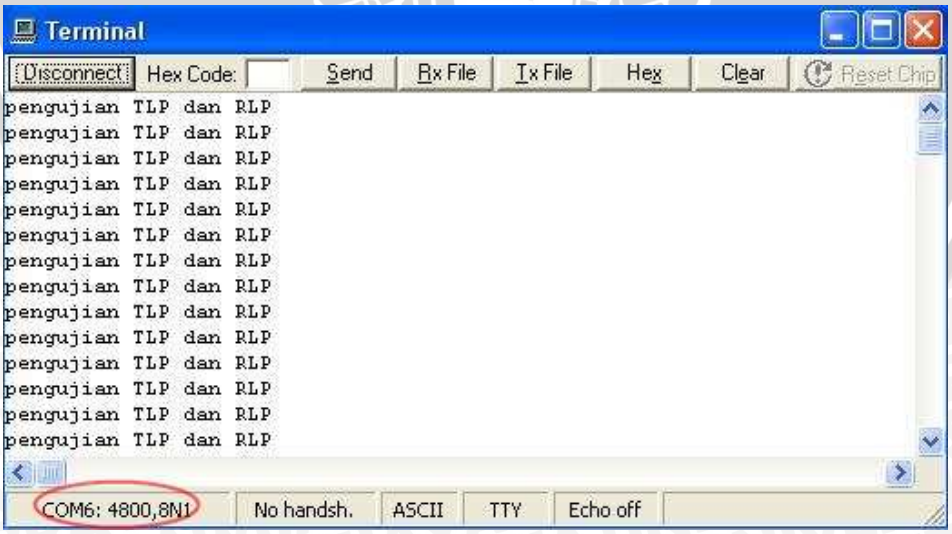
5.4.2 Pengujian kecepatan transfer data (*baud rate*) modul RF

Pengujian kecepatan transfer data (*baud rate*) bertujuan untuk mengetahui kecepatan transfer data maksimal yang masih bisa digunakan dalam proses transmisi data. Pengujian dilakukan dengan cara menghubungkan mikrokontroler dengan pemancar TLP 434A dan menghubungkan penerima RLP 434A dengan komputer. Pengujian dilakukan dengan mengubah kecepatan transfer data modul *radio frequency* sampai ditemukan *error* dalam proses transmisi data yaitu hilangnya atau berubahnya beberapa karakter yang diterima oleh RLP 434A. Diagram blok pengujian ini ditunjukkan dalam Gambar 5.15.

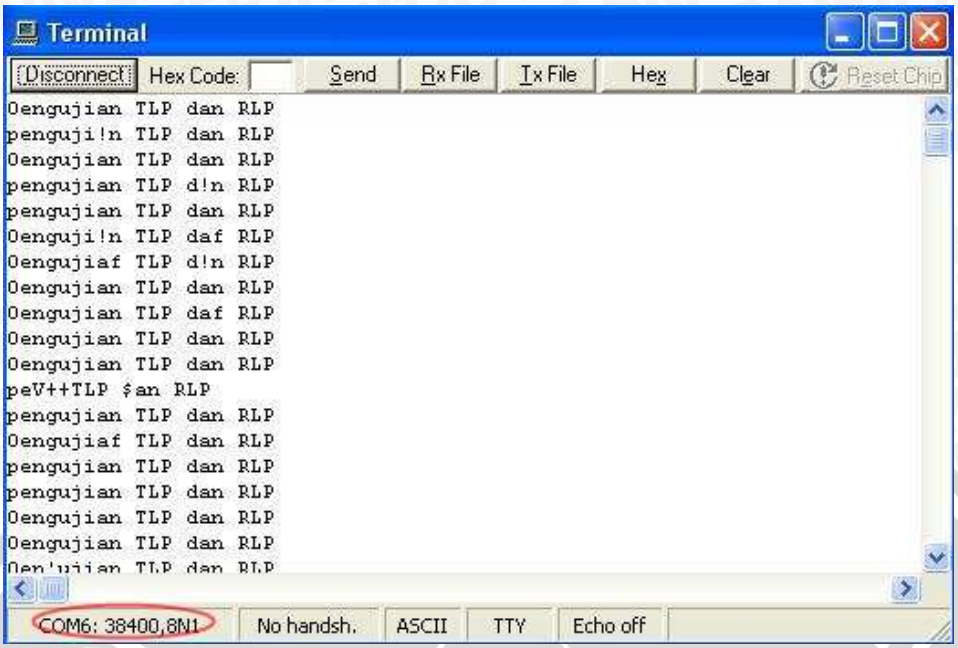


Gambar 5.15 Diagram blok pengujian kecepatan transfer data (*baud rate*) pemancar TLP 434A dan penerima RLP 434A

Hasil pengujian kecepatan transfer data rangkaian pemancar TLP 434A dan penerima RLP 434A ditunjukkan dalam Gambar 5.16 dan Gambar 5. 17.



Gambar 5.16 Tampilan hasil pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A dengan *baud rate* 4800 bps



Gambar 5.17 Tampilan hasil pengujian rangkaian pemancar TLP 434A dan penerima RLP 434A dengan *baud rate* 38400 bps

Dari hasil pengujian ini menunjukkan bahwa hingga kecepatan transfer data diubah menjadi 19200 bps tidak terdeteksi *error*. Tetapi untuk keamanan data, maka kecepatan transfer data dapat menggunakan *baud rate* 9600 bps.

5.4.3 Pengujian komunikasi data dan jarak jangkau transmisi modul RF

Berdasarkan karakteristik modul RF, jarak transmisi adalah 100 meter di luar ruangan (tempat terbuka) dan 30 meter di dalam ruangan. Berdasarkan perhitungan jarak jangkau transmisi didapatkan dengan persamaan:

$SOM = daya_{Tx} – redaman_{Tx} + penguatan\ antena_{Tx} - FSL + penguatan\ antena_{Rx} - redaman_{Rx} – sensitivity$

$54 = 16 – 3 + 0 – FSL + 0 – 3 – (-110)$

$54 = 120 – FSL$

$FSL = 66$

Keterangan :

SOM : system operating margin

FSL : freespace loss

Dari hasil FSL = 110 tersebut, dapat ditemukan jarak jangkau transmisi dengan persamaan:

$FSL = 20 \log f + 20 \log d + 92.4$

66 = 20 log 433,92 + 20 log d + 92.4

= 85,15 + 20 log d

d = 0,110 km = 110 meter

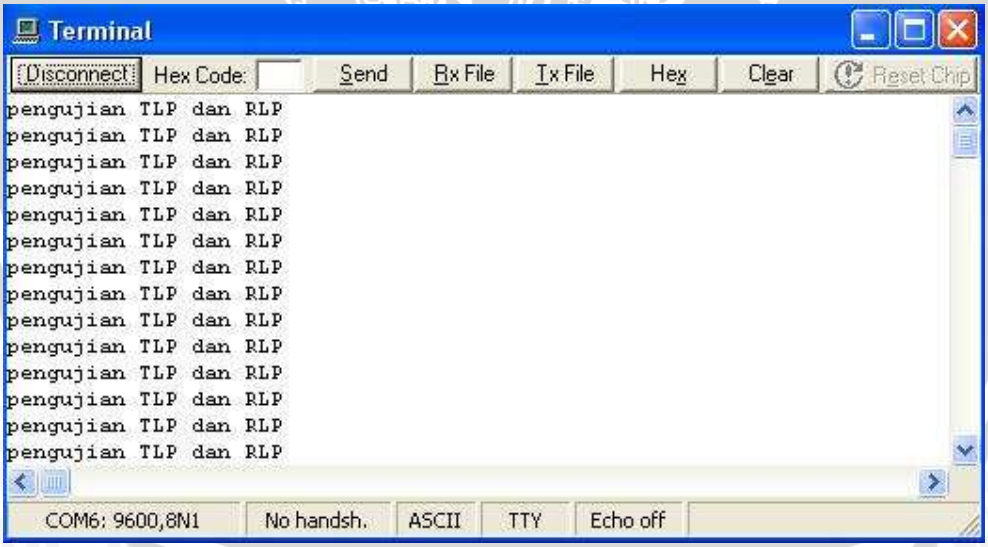
Keterangan :

f : frekuensi dalam GHz

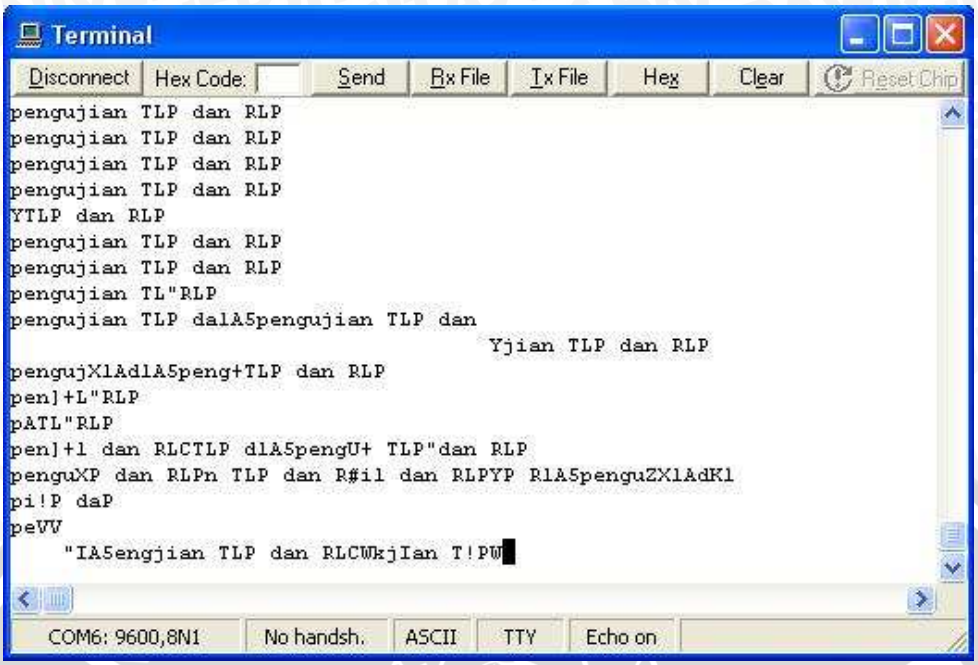
d : jarak antara *transmiter* dan *receiver* dalam km

Dari perhitungan tersebut didapatkan jarak jangkau transmisi adalah 110 meter.

Pengujian komunikasi data dan jarak jangkau transmisi *radio frequency* bertujuan untuk mengetahui keberhasilan dan jarak jangkau transmisi data antara rangkaian pemancar TLP 434A dan penerima RLP 434A. Pengujian dilakukan dengan cara yang sama dengan pengujian kecepatan transfer data, yaitu menghubungkan mikrokontroler dengan pemancar TLP 434A dan menghubungkan penerima RLP 434A dengan komputer. Diagram blok pengujian ini ditunjukkan dalam Gambar 5.15. Mikrokontroler diprogram untuk menampilkan tulisan “Pengujian TLP dan RLP”. Kecepatan transmisi data yang digunakan adalah 9600 bps. Pengujian dilakukan dengan mengubah jarak antara pemancar TLP 434A dan penerima RLP 434A hingga ditemukan *error* yaitu hilangnya atau berubahnya beberapa karakter yang diterima oleh RLP 434A.



Gambar 5.18 Tampilan hasil pengujian transmisi data rangkaian pemancar TLP 434A dan penerima RLP 434A dengan jarak 20 m

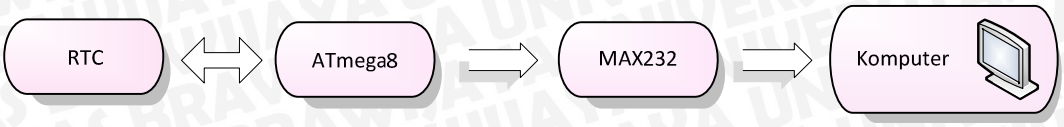


Gambar 5.19 Tampilan hasil pengujian transmisi data rangkaian pemancar TLP 434A dan penerima RLP 434A dengan jarak lebih dari 20 m

Hasil pengujian transmisi data rangkaian pemancar TLP 434A dan penerima RLP 434A ditunjukkan dalam Gambar 5.18 dan Gambar 5.19. Dari hasil pengujian ini dapat disimpulkan bahwa tidak ada karakter yang hilang atau rusak dalam setiap paket data yang diterima. Berdasarkan pengukuran jarak jangkau transmisi menunjukkan bahwa data yang diterima oleh unit penerima RF sampai jarak 20 meter tidak terdeteksi *error*. Sehingga disimpulkan bahwa transmisi komunikasi data dapat menjangkau sampai jarak 20 meter.

5.5 Pengujian Sistem Pewaktu RTC DS1307

Pengujian sistem pewaktu RTC DS1307 bertujuan untuk mengetahui apakah sistem pewaktu RTC DS1307 dapat bekerja dengan presisi atau tidak. Pengujian ini dilakukan dengan cara menghubungkan RTC dengan mikrokontroler yang telah diprogram. Kemudian mikrokontroler dihubungkan dengan MAX232 dan MAX232 dihubungkan dengan komputer. Hasil pengujian dapat diamati pada *Hyperterminal* program *Code Vision AVR*. Diagram blok pengujian ini ditunjukkan dalam Gambar 5.20.



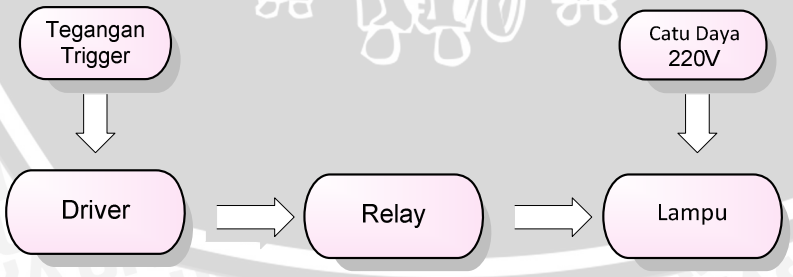
Gambar 5.20 Diagram blok pengujian sistem pewaktu RTC DS1307

Gambar 5.21 Tampilan hasil pengujian sistem pewaktu RTC DS1307

Hasil pengujian ditunjukkan dalam Gambar 5.21. Hasil pengujian menunjukkan bahwa sistem pewaktu RTC DS1307 dapat bekerja dengan presisi.

5.6 Pengujian Rangkaian *Driver Relay*

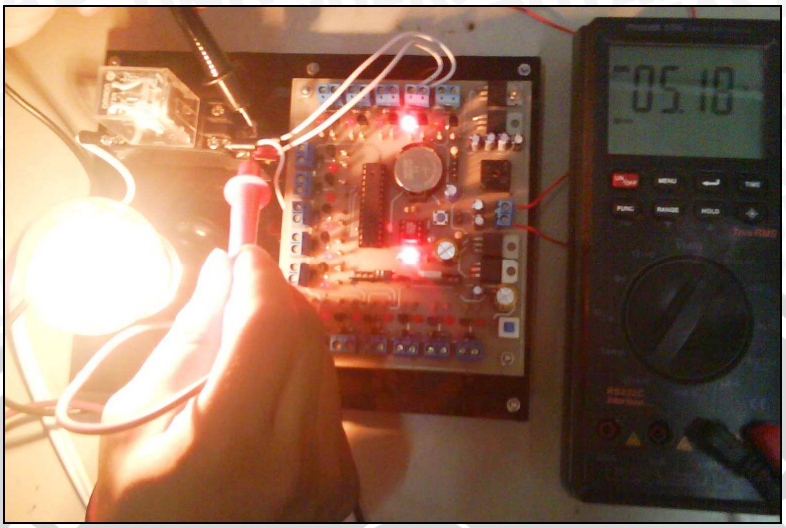
Pengujian rangkaian *driver relay* bertujuan untuk mengetahui keberhasilan rangkaian *driver relay* yang dirancang. Prosedur pengujian ini dilakukan dengan menghubungkan rangkaian *driver relay* ke beban yaitu lampu pijar kemudian memberikan tegangan trigger 0 volt atau 5 volt secara bergantian ke rangkaian *driver relay*. Diagram blok pengujian ditunjukkan dalam Gambar 5.22.



Gambar 5.22 Diagram blok pengujian rangkaian *driver relay*

Hasil pengujian rangkaian *driver relay* ini ditunjukkan dalam Gambar 5.23 dan Tabel 5.5. Berdasarkan hasil pengujian dapat disimpulkan bahwa rangkaian *driver relay* bekerja sesuai dengan perancangan, yaitu ketika rangkaian *driver relay* diberi tegangan

trigger 0 volt maka lampu mati dan ketika diberi tegangan trigger 5 volt maka lampu menyala.



Gambar 5.23 Pengujian rangkaian driver relay

Tabel 5.6 Hasil pengujian rangkaian *driver relay*

Tegangan *Trigger*	Lampu
0V	Mati
5V	Hidup

5.7 Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem bertujuan untuk mengetahui apakah sistem telah bekerja sesuai dengan perancangan atau tidak. Pengujian keseluruhan dilakukan dengan menggabungkan semua bagian dalam sistem dan melihat hasil kinerja sistem. *Remote control* akan mengirim data hasil pembacaan *keypad* dan saklar ke unit pengolah data. Data yang dikirim akan dibaca oleh perangkat lunak yang telah dirancang pada unit pengolah data dan kemudian akan memberikan sinyal kepada rangkaian *driver relay* untuk menyalakan atau mematikan peralatan listrik berdasarkan hasil pengolahan data. Pengujian keseluruhan sistem dilakukan dengan menggunakan beberapa macam beban, diantaranya lampu, setrika dan televisi.

5.7.1 Pengujian keseluruhan sistem dengan beban lampu

Pengujian dilakukan untuk semua mode, yaitu mode hidup, mode mati, mode hidup 30 menit, mode hidup 1 jam, mode hidup 2 jam, dan mode otomatis. Pengujian untuk mode hidup dilakukan dengan mengatur kombinasi saklar pada *remote control* untuk mode hidup dan menekan semua tombol *keypad* secara bergantian, kemudian

melihat apakah semua beban dapat menyala sesuai dengan penekanan tombol *keypad*. Tabel 5.7 menunjukkan hasil pengujian keseluruhan sistem untuk mode hidup.

Tabel 5.7 Hasil pengujian keseluruhan sistem mode hidup

Tombol *Keypad* yang Ditekan	Keterangan
1	Beban 1 menyala
2	Beban 2 menyala
3	Beban 3 menyala
4	Beban 4 menyala
5	Beban 5 menyala
6	Beban 6 menyala
7	Beban 7 menyala
8	Beban 8 menyala
9	Beban 9 menyala
0	Beban 10 menyala
*	Beban 11 menyala
#	Beban 12 menyala
A	Beban 13 menyala
B	Beban 14 menyala
C	Beban 15 menyala
D	Beban 16 menyala

Pengujian untuk mode mati diawali dengan menyalakan semua beban kemudian mengatur kombinasi saklar pada *remote control* untuk mode mati dan menekan semua tombol *keypad* secara bergantian, kemudian melihat apakah semua beban dapat mati sesuai dengan penekanan tombol *keypad*. Tabel 5.8 menunjukkan hasil pengujian keseluruhan sistem untuk mode mati.

Tabel 5.8 Hasil pengujian keseluruhan sistem mode mati

Tombol *Keypad* yang Ditekan	Keterangan
1	Beban 1 mati
2	Beban 2 mati
3	Beban 3 mati
4	Beban 4 mati
5	Beban 5 mati
6	Beban 6 mati

7	Beban 7 mati
8	Beban 8 mati
9	Beban 9 mati
0	Beban 10 mati
*	Beban 11 mati
#	Beban 12 mati
A	Beban 13 mati
B	Beban 14 mati
C	Beban 15 mati
D	Beban 16 mati

Pengujian untuk mode hidup 30 menit dilakukan dengan mengatur kombinasi saklar pada *remote control* untuk mode hidup 30 menit dan menekan semua tombol *keypad* secara bergantian. Kemudian melihat apakah semua beban dapat menyala sesuai dengan penekanan tombol *keypad* dan setelah 30 menit apakah semua beban tersebut mati. Pengujian ini dilakukan dimulai pada pukul 10.05 yang kemudian mati pada pukul 10.35. pada Tabel 5.9 menunjukkan hasil pengujian keseluruhan sistem untuk mode hidup.

Tabel 5.9 Hasil pengujian keseluruhan sistem mode hidup 30 menit

Tombol *Keypad* yang Ditekan	Keterangan	Waktu
1	Beban 1 menyala	10.05 - 10.35
2	Beban 2 menyala	10.05 - 10.35
3	Beban 3 menyala	10.05 - 10.35
4	Beban 4 menyala	10.05 - 10.35
5	Beban 5 menyala	10.05 - 10.35
6	Beban 6 menyala	10.05 - 10.35
7	Beban 7 menyala	10.05 - 10.35
8	Beban 8 menyala	10.05 - 10.35
9	Beban 9 menyala	10.05 - 10.35
0	Beban 10 menyala	10.05 - 10.35
*	Beban 11 menyala	10.05 - 10.35
#	Beban 12 menyala	10.05 - 10.35
A	Beban 13 menyala	10.05 - 10.35
B	Beban 14 menyala	10.05 - 10.35

C	Beban 15 menyala	10.05 - 10.35
D	Beban 16 menyala	10.06 - 10.36

Pengujian untuk mode hidup 1 jam dilakukan dengan mengatur kombinasi saklar pada *remote control* untuk mode hidup 1 jam dan menekan semua tombol *keypad* secara bergantian. Kemudian melihat apakah semua beban dapat menyala sesuai dengan penekanan tombol *keypad* dan setelah 1 jam apakah semua beban tersebut mati. Pengujian ini dilakukan dimulai pada pukul 10.40 yang kemudian mati pada pukul 11.40. Tabel 5.10 menunjukkan hasil pengujian keseluruhan sistem untuk mode hidup.

Tabel 5.10 Hasil pengujian keseluruhan sistem mode hidup 1 jam

Tombol *Keypad* yang Ditekan	Keterangan	Waktu
1	Beban 1 menyala	10.40 - 11.40
2	Beban 2 menyala	10.40 - 11.40
3	Beban 3 menyala	10.40 - 11.40
4	Beban 4 menyala	10.40 - 11.40
5	Beban 5 menyala	10.40 - 11.40
6	Beban 6 menyala	10.40 - 11.40
7	Beban 7 menyala	10.40 - 11.40
8	Beban 8 menyala	10.40 - 11.40
9	Beban 9 menyala	10.40 - 11.40
0	Beban 10 menyala	10.40 - 11.40
*	Beban 11 menyala	10.40 - 11.40
#	Beban 12 menyala	10.40 - 11.40
A	Beban 13 menyala	10.41 – 11.41
B	Beban 14 menyala	10.41 – 11.41
C	Beban 15 menyala	10.41 – 11.41
D	Beban 16 menyala	10.41 – 11.41

Pengujian untuk mode hidup 2 jam dilakukan dengan mengatur kombinasi saklar pada *remote control* untuk mode hidup 2 jam dan menekan semua tombol *keypad* secara bergantian. Kemudian melihat apakah semua beban dapat menyala sesuai dengan penekanan tombol *keypad* dan setelah 2 jam apakah semua beban tersebut mati. Pengujian ini dilakukan dimulai pada pukul 11.45 yang kemudian mati pada pukul 13.45. Tabel 5.11 menunjukkan hasil pengujian keseluruhan sistem untuk mode hidup.

Tabel 5.11 Hasil pengujian keseluruhan sistem mode hidup 2 jam

Tombol *Keypad* yang Ditekan	Keterangan	Waktu
1	Beban 1 menyala	11.45 – 13.45
2	Beban 2 menyala	11.45 – 13.45
3	Beban 3 menyala	11.45 – 13.45
4	Beban 4 menyala	11.45 – 13.45
5	Beban 5 menyala	11.45 – 13.45
6	Beban 6 menyala	11.45 – 13.45
7	Beban 7 menyala	11.45 – 13.45
8	Beban 8 menyala	11.45 – 13.45
9	Beban 9 menyala	11.45 – 13.45
0	Beban 10 menyala	11.46 – 13.46
*	Beban 11 menyala	11.46 – 13.46
#	Beban 12 menyala	11.46 – 13.46
A	Beban 13 menyala	11.46 – 13.46
B	Beban 14 menyala	11.46 – 13.46
C	Beban 15 menyala	11.46 – 13.46
D	Beban 16 menyala	11.46 – 13.46

Pengujian untuk mode otomatis dilakukan dengan mengatur kombinasi saklar pada *remote control* untuk mode otomatis, menekan tombol '*' kemudian menekan tombol '1', '2' dan '3'. Kemudian melihat apakah beban tersebut dapat menyala dan mati sesuai dengan waktu yang telah ditentukan. Pada perangkat lunak unit pengolah data telah diprogram pada mode otomatis beban akan menyala pada jam 17.00 dan mati pada jam 06.00. Tabel 5.12 menunjukkan hasil pengujian keseluruhan sistem untuk mode hidup.

Tabel 5.12 Hasil pengujian keseluruhan sistem mode otomatis

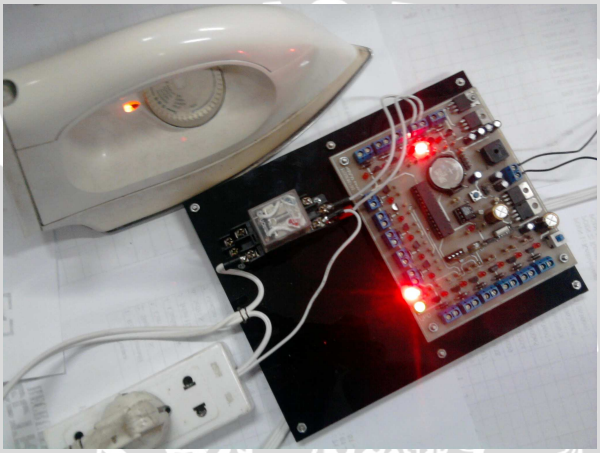
Keypad	Beban	Waktu Nyala	Waktu Mati
1	1	17.00	06.00
2	2	17.00	06.00
C	15	17.00	06.00

Berdasarkan hasil pengujian keseluruhan sistem dengan beban lampu ini dapat disimpulkan bahwa sistem dapat berfungsi sesuai dengan perancangan. Yaitu sistem dapat menyalakan dan mematikan beban sesuai dengan tombol keypad yang ditekan,

sistem dapat menyalakan beban selama selang waktu tertentu kemudian mematikannya secara otomatis dan sistem juga bisa menyalakan dan mematikan beban pada pada waktu-waktu tertentu secara otomatis.

5.7.2 Pengujian keseluruhan sistem dengan beban setrika

Pengujian keseluruhan sistem dengan beban setrika dilakukan dengan mengatur kombinasi saklar pada *remote control* untuk mode hidup dan menekan tombol *keypad*, kemudian melihat apakah setrika dapat menyala sesuai dengan penekanan tombol *keypad*.



Gambar 5.24 Pengujian keseluruhan sistem dengan beban setrika

Gambar 5.24 menunjukkan hasil pengujian keseluruhan sistem dengan beban setrika. Dari hasil pengujian tersebut dapat disimpulkan bahwa sistem dapat berfungsi sesuai perancangan, yaitu sistem dapat menyalakan setrika.

5.7.3 Pengujian keseluruhan sistem dengan beban televisi

Pengujian keseluruhan sistem dengan beban televisi dilakukan dengan mengatur kombinasi saklar pada *remote control* untuk mode hidup dan menekan tombol *keypad*, kemudian melihat apakah televisi dapat menyala sesuai dengan penekanan tombol *keypad*. Gambar 5.25 menunjukkan hasil pengujian keseluruhan sistem dengan beban setrika.

Gambar 5.25 Pengujian keseluruhan sistem dengan beban televisi

Dari hasil pengujian tersebut dapat disimpulkan bahwa sistem dapat berfungsi sesuai perancangan, yaitu sistem dapat menyalakan televisi.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil perancangan dan pengujian yang telah dilakukan, dapat ditarik kesimpulan sebagai berikut:

1) Sistem *remote control* dapat dirancang menggunakan beberapa rangkaian penyusun, diantaranya rangkaian *keypad*, rangkaian pengendali utama yang menggunakan mikrokontroler ATMega 8, rangkaian LCD dan rangkaian pemancar TLP 434A. Remote *control* yang dirancang dan dibuat telah dapat mengirimkan data valid dengan kecepatan 9600 bps dan jarak transmisi maksimum dari unit pengolah data di dalam ruangan adalah 20 meter.
2) Rancangan unit pengolah data terdiri atas rangkaian penerima RLP 434A, rangkaian pengendali utama yang menggunakan mikrokontroler ATMega 8, rangkaian sistem pewaktu. Unit pengolah data yang dirancang dan telah dibuat, dengan kecepatan sama dengan *remote control*, telah dapat menerima data, menyeleksi data sesuai dengan format paket data yang telah dirancang dan memberikan respon dengan memberikan tegangan *trigger* pada rangkaian *driver relay*.
3) Rangkaian *driver relay* terdiri atas transistor dan *relay*. *Driver relay* telah dapat melakukan pensaklaran peralatan listrik sesuai dengan data yang dikirim oleh unit pengolah data.
4) Sistem *remote control* pengatur *on/off* peralatan listrik menggunakan dua mikrokontroler, yaitu pada *remote control* dan pada unit pengolah data. Perangkat lunak mikrokontroler pada *remote control* berfungsi untuk mendeteksi penekanan tombol, mengolah data hasil penekanan tombol dan mengirimkannya ke pusat pengolah data melalui modul pemancar RF. Perangkat lunak mikrokontroler pada unit pengolah data berfungsi mengatur kerja mikrokontroler untuk menerima data serial yang dikirimkan *remote control*, mengambil data waktu dari RTC dan menberikan tegangan *trigger* pada rangkaian *driver relay*. Proses pendeteksian penekanan tombol hingga proses pemberian tegangan *trigger* pada *driver relay* telah dapat berfungsi dengan baik.

6.2 Saran

Saran-saran dalam pengimplementasian maupun peningkatan unjuk kerja sistem ini dapat diuraikan sebagai berikut:

1) Dalam sistem ini jarak transmisi maksimum di dalam ruangan adalah 20 meter. Oleh karena itu, dalam pengembangan selanjutnya dapat menggunakan modul RF dengan jangkauan yang lebih jauh.
2) Sistem dapat dikembangkan dengan menggunakan mikrokontroler yang memiliki unit input ouput yang lebih banyak, sehingga peralatan listrik yang diatur bisa lebih banyak.

Daftar Pustaka

Atmel. 2007. *ATMEGA8/ATMEGA8L, 8-bit AVR with 8 Kbytes in System Programable Flash*. http://www.atmel.com/dyn/resources/prod_documents/doc8159.pdf. Diakses tanggal 18 November 2009.

Dallas. 2000. *DS1307/DS1308 64 X 8 Serial Time Clock. Dallas Semiconductor.* https://www.datasheet-pdf/view/58481/DALLAS/DS1307.html. Diakses tanggal 24 Januari 2010.

Delta Electronic. 2006. *Aplikasi Pengiriman Data Tanpa Kabel.* http://www.delta-electronics.com/design/apnote/an0093.pdf. Diakses tanggal 24 Januari 2010.

Delta Electronic. 2006. *Wireless RF Communication.* http://www. delta-electronics.com/design/apnote/wirelessdst51.pdf. Diakses tanggal 24 Januari 2010.

Laipac. 2005. *RLP434A SAW Based Receiver*. Canada: Laipac Technology, Inc. http://www.laipac.com/Downloads/Easy/RLP434A.pdf. Diakses tanggal 4 Februari 2010.

Laipac. 2005. *TLP 434A-RF ASK Hybrid Modules for Radio Control.* Canada Laipac Technology, Inc. http://www.laipac.com/Downloads/Easy/tlp434a. pdf. Diakses tanggal 4 Februari 2010.

Stallings, William. 2004. *Data and Computer Communications.* New Jersey: Prentice Hall.

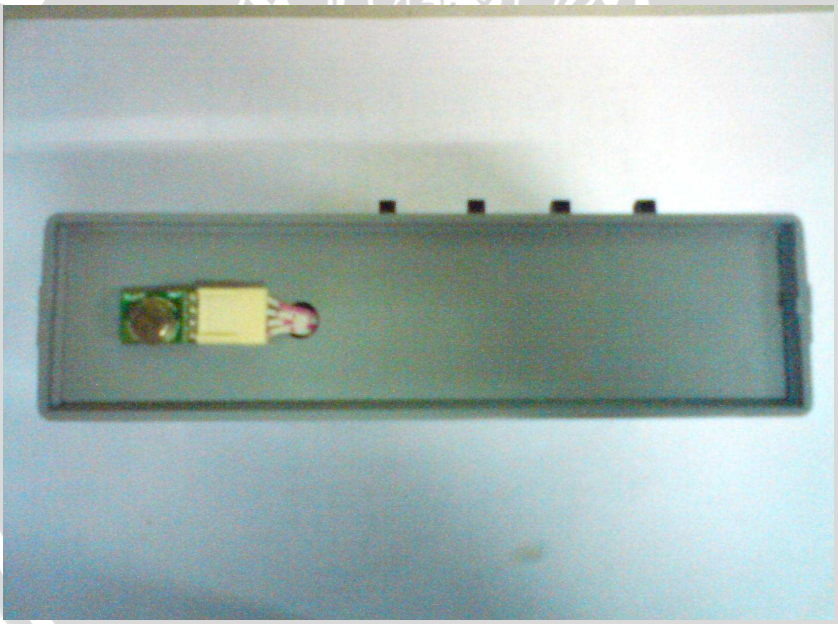
Topway. 2004. *LMB162ADC LCD Module User Manual*. Shenzen Topway Technology Co., Ltd. http://www.lcd-module.com.hk/image/TOPWAY/LMB162ADC-Manual-pdf.pdf. Diakses tanggal 10 Juli 2010.

Winch, Robert G. 1998. *Telecommunication Transmission Systems.* New York: The McGraw-Hill Companies, Inc.

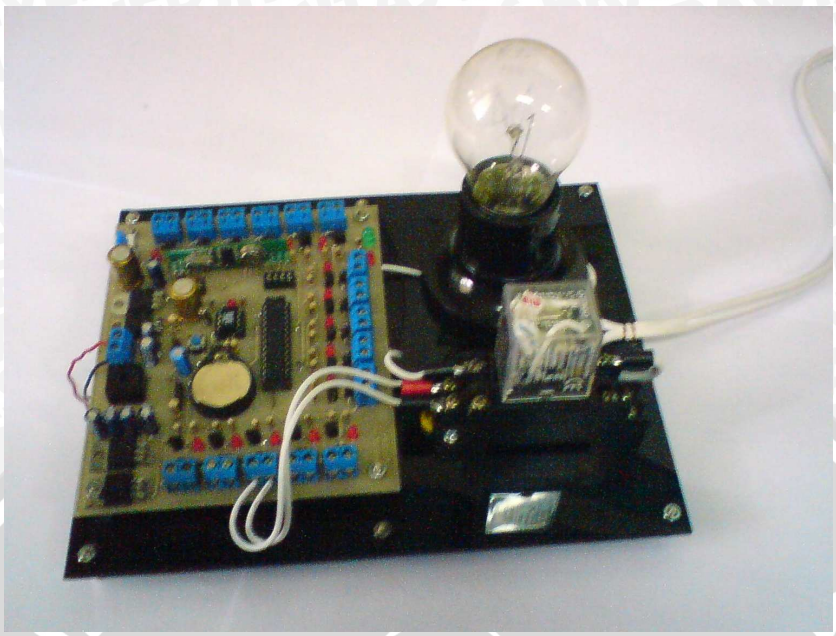
LAMPIRAN I

FOTO ALAT

Gambar 1. *Remote Control*



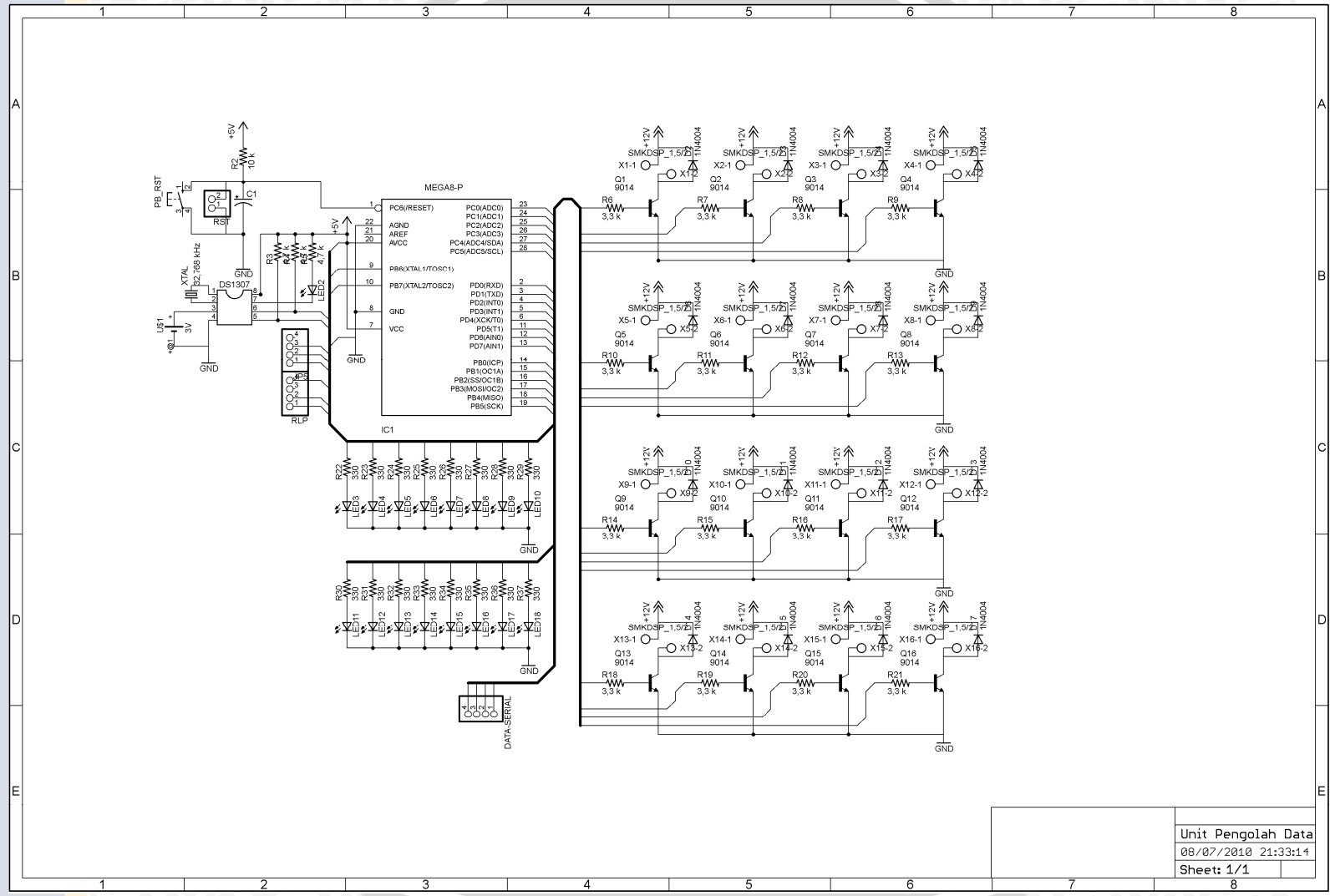
Gambar 2. Modul RF yang digunakan *remote control*

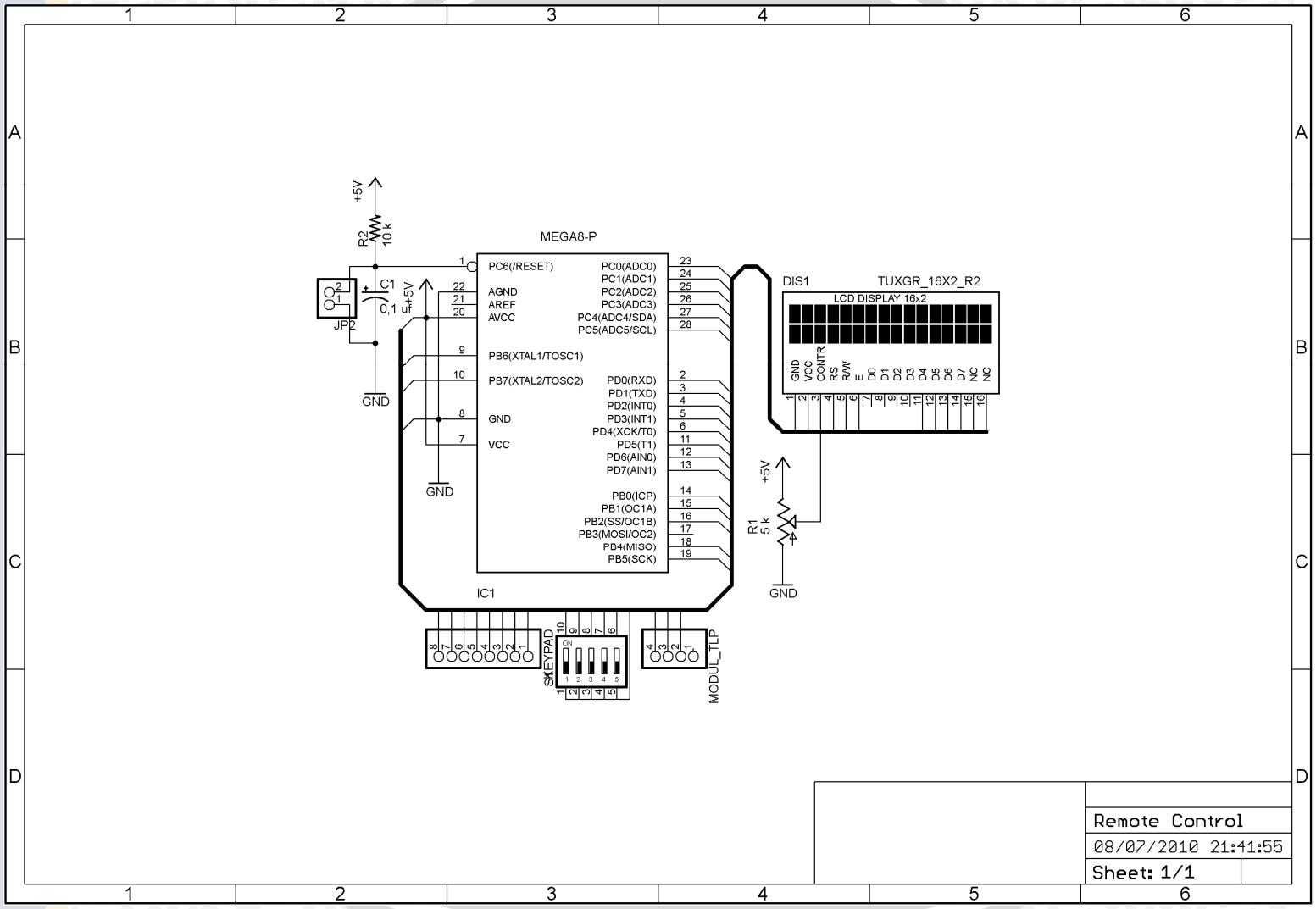


Gambar 3. Unit pengolah data dan rangkaian *driver relay*

LAMPIRAN II

GAMBAR RANGKAIAN


MEGA8-P
IC1
DS1307
RLP
DATA-SERIAL
Unit Pengolah Data
08/07/2010 21:33:14
Sheet: 1/1


MEGA8-P
PC6(/RESET)
AGND
AREF
AVCC
PB6(XTAL1/TOSC1)
PB7(XTAL2/TOSC2)
GND
VCC
PC0(ADC0)
PC1(ADC1)
PC2(ADC2)
PC3(ADC3)
PC4(ADC4/SDA)
PC5(ADC5/SCL)
PD0(RXD)
PD1(TXD)
PD2(INT0)
PD3(INT1)
PD4(XCK/T0)
PD5(T1)
PD6(AIN0)
PD7(AIN1)
PB0(ICP)
PB1(OC1A)
PB2(SS/OC1B)
PB3(MOSI/OC2)
PB4(MISO)
PB5(SCK)
IC1
+5V
R2
10 k
C1
0,1
JP2
R1
5 k
DIS1
TUXGR_16X2_R2
LCD DISPLAY 16x2
GND
VCC
CONTR
RS
R/W
E
D0
D1
D2
D3
D4
D5
D6
D7
NC
NC
SKEYPAD
MODUL_TLP
Remote Control
08/07/2010 21:41:55
Sheet: 1/1

LAMPIRAN III

LISTING PROGRAM MIKROKONTROLER

ATMega8 *REMOTE CONTROL*

```
#include <mega8.h>
#include <delay.h>
#asm
  .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>
#include <stdio.h>

//-----------------------//
//inisialisasi keypad    //
//-----------------------//
#define baris1  PORTD.2  //
#define baris2  PORTD.3  //
#define baris3  PORTC.5  //
#define baris4  PORTC.4  //
#define kolom1  PINC.3   //
#define kolom2  PINC.2   //
#define kolom3  PINC.1   //
#define kolom4  PINC.0   //

unsigned char buff[33];
char tombol,beban;
int i;
void scan_keypad()
{
 tombol=0;
 if (PIND.4==1)
 {
 lcd_clear();
 lcd_putsf("mode otomatis");
 PORTC=0x3F;
 DDRC=0xF0;
 PORTD=0xFF;
 DDRD=0b00001110;
 baris1=0;      //cek baris1
 delay_us(1);
 if(kolom1==0)   tombol=0;     else
 if(kolom2==0)   tombol=0;     else
 if(kolom3==0)   tombol=0;     else
```

```
if(kolom4==0)   tombol=0;     else
  {
  baris1=1;
  baris2=0;    //cek baris2
  delay_us(1);
  if(kolom1==0)   tombol=0;     else
  if(kolom2==0)   tombol=0;     else
  if(kolom3==0)   tombol=0;     else
  if(kolom4==0)   tombol=0;     else
     {
     baris2=1;
     baris3=0;      //cek baris3
     delay_us(1);
     if(kolom1==0)   tombol=0;     else
     if(kolom2==0)   tombol=0;     else
     if(kolom3==0)   tombol=0;     else
     if(kolom4==0)   tombol=0;     else
       {
       baris3=1;
       baris4=0;    //cek baris4
       delay_us(1);
       if(kolom1==0)   tombol=82;     else
       if(kolom2==0)   tombol=0;     else
       if(kolom3==0)   tombol=0;     else
       if(kolom4==0)   tombol=0;     else
                tombol=0;
       };
     };
  };
}

else if (PIND.5==0)
{
lcd_clear();
lcd_putsf("mati");
PORTC=0x3F;
DDRC=0xF0;
PORTD=0xFF;
DDRD=0b00001110;
baris1=0;      //cek baris1
```

```
delay_us(1);
if(kolom1==0)  {tombol=1;  beban=1;}  else
if(kolom2==0)  {tombol=2;  beban=2;}  else
if(kolom3==0)  {tombol=3;  beban=3;}  else
if(kolom4==0)  {tombol=81; beban=13;}  else
  {
  baris1=1;
  baris2=0;   //cek baris2
  delay_us(1);
  if(kolom1==0)  {tombol=4;  beban=4;}  else
  if(kolom2==0)  {tombol=5;  beban=5;}  else
  if(kolom3==0)  {tombol=6;  beban=6;}  else
  if(kolom4==0)  {tombol=14;  beban=14;}  else
     {
     baris2=1;
     baris3=0;      //cek baris3
     delay_us(1);
     if(kolom1==0)  {tombol=7;  beban=7;}  else
     if(kolom2==0)  {tombol=8;  beban=8;}  else
     if(kolom3==0)  {tombol=9;  beban=9;}  else
     if(kolom4==0)  {tombol=15;  beban=15;}  else
       {
       baris3=1;
       baris4=0;   //cek baris4
       delay_us(1);
       if(kolom1==0)  {tombol=11;  beban=10;}  else
       if(kolom2==0)  {tombol=10;  beban=11;}  else
       if(kolom3==0)  {tombol=12;  beban=12;}  else
       if(kolom4==0)  {tombol=16;  beban=16;}  else
                      tombol=0;
       };
     };
  };
}

else if ((PIND.6==0)&&(PIND.7==0))
{
lcd_clear();
lcd_putsf("hidup");
PORTC=0x3F;
```

```
DDRC=0xF0;
PORTD=0xFF;
DDRD=0b00001110;
baris1=0;     //cek baris1
delay_us(1);
if(kolom1==0)   {tombol=17; beban=1;}    else
if(kolom2==0)   {tombol=18; beban=2;}    else
if(kolom3==0)   {tombol=19; beban=3;}    else
if(kolom4==0)   {tombol=29; beban=13;}   else
  {
  baris1=1;
  baris2=0;    //cek baris2
  delay_us(1);
  if(kolom1==0)   {tombol=20;  beban=4;}     else
  if(kolom2==0)   {tombol=21;  beban=5;}     else
  if(kolom3==0)   {tombol=22;  beban=6;}     else
  if(kolom4==0)   {tombol=30;  beban=14;}    else
     {
     baris2=1;
     baris3=0;      //cek baris3
     delay_us(1);
     if(kolom1==0)   {tombol=23;  beban=7;}     else
     if(kolom2==0)   {tombol=24;  beban=8;}     else
     if(kolom3==0)   {tombol=25;  beban=9;}     else
     if(kolom4==0)   {tombol=31;  beban=15;}    else
       {
       baris3=1;
       baris4=0;    //cek baris4
       delay_us(1);
       if(kolom1==0)   {tombol=27;  beban=10;}     else
       if(kolom2==0)   {tombol=26;  beban=11;}     else
       if(kolom3==0)   {tombol=28;  beban=12;}     else
       if(kolom4==0)   {tombol=32;  beban=16;}     else
             tombol=0;
       };
     };
  }
```

```
else if ((PIND.6==1)&&(PIND.7==0))
 {
 lcd_clear();
 lcd_putsf("hidup 30 menit");
 PORTC=0x3F;
 DDRC=0xF0;
 PORTD=0xFF;
 DDRD=0b00001110;
 baris1=0;     //cek baris1
 delay_us(1);
 if(kolom1==0)   {tombol=33; beban=1;}    else
 if(kolom2==0)   {tombol=34; beban=2;}    else
 if(kolom3==0)   {tombol=35; beban=3;}    else
 if(kolom4==0)   {tombol=45; beban=13;}   else
   {
   baris1=1;
   baris2=0;    //cek baris2
   delay_us(1);
   if(kolom1==0)   {tombol=36;  beban=4;}    else
   if(kolom2==0)   {tombol=37;  beban=5;}    else
   if(kolom3==0)   {tombol=38;  beban=6;}    else
   if(kolom4==0)   {tombol=46;  beban=14;}   else
     {
     baris2=1;
     baris3=0;     //cek baris3
     delay_us(1);
     if(kolom1==0)   {tombol=39;  beban=7;}    else
     if(kolom2==0)   {tombol=40;  beban=8;}    else
     if(kolom3==0)   {tombol=41;  beban=9;}    else
     if(kolom4==0)   {tombol=47;  beban=15;}   else
       {
       baris3=1;
       baris4=0;    //cek baris4
       delay_us(1);
       if(kolom1==0)   {tombol=43;   beban=10;}    else
       if(kolom2==0)   {tombol=42;   beban=11;}    else
       if(kolom3==0)   {tombol=44;   beban=12;}    else
       if(kolom4==0)   {tombol=48;   beban=16;}    else
                 tombol=0;
       };
```

```
      };
   };
}

else if ((PIND.6==0)&&(PIND.7==1))
 {
 lcd_clear();
 lcd_putsf("hidup 1 jam");
 PORTC=0x3F;
 DDRC=0xF0;
 PORTD=0xFF;
 DDRD=0b00001110;
 baris1=0;     //cek baris1
 delay_us(1);
 if(kolom1==0)   {tombol=49; beban=1;}    else
 if(kolom2==0)   {tombol=50; beban=2;}    else
 if(kolom3==0)   {tombol=51; beban=3;}    else
 if(kolom4==0)   {tombol=61; beban=13;}    else
   {
   baris1=1;
   baris2=0;    //cek baris2
   delay_us(1);
   if(kolom1==0)   {tombol=52;  beban=4;}   else
   if(kolom2==0)   {tombol=53;  beban=5;}   else
   if(kolom3==0)   {tombol=54;  beban=6;}   else
   if(kolom4==0)   {tombol=62;  beban=14;}   else
     {
     baris2=1;
     baris3=0;      //cek baris3
     delay_us(1);
     if(kolom1==0)   {tombol=55;  beban=7;}   else
     if(kolom2==0)   {tombol=56;  beban=8;}   else
     if(kolom3==0)   {tombol=57;  beban=9;}   else
     if(kolom4==0)   {tombol=63;  beban=15;}   else
      {
      baris3=1;
      baris4=0;    //cek baris4
      delay_us(1);
       if(kolom1==0)   {tombol=59;   beban=10;}  else
       if(kolom2==0)   {tombol=58;   beban=11;}  else
```

```
        if(kolom3==0)   {tombol=60;   beban=12;}  else
        if(kolom4==0)   {tombol=64;   beban=16;}  else
                    tombol=0;
        };
      };
   };
}

else if ((PIND.6==1)&&(PIND.7==1))
 {
 lcd_clear();
 lcd_putsf("hidup 2 jam");
 PORTC=0x3F;
 DDRC=0xF0;
 PORTD=0xFF;
 DDRD=0b00001110;
 baris1=0;     //cek baris1
 delay_us(1);
 if(kolom1==0)   {tombol=65; beban=1;}    else
 if(kolom2==0)   {tombol=66; beban=2;}    else
 if(kolom3==0)   {tombol=67; beban=3;}    else
 if(kolom4==0)   {tombol=77; beban=13;}    else
   {
  baris1=1;
   baris2=0;    //cek baris2
   delay_us(1);
   if(kolom1==0)   {tombol=68;  beban=4;}   else
   if(kolom2==0)   {tombol=69;  beban=5;}   else
   if(kolom3==0)   {tombol=70;  beban=6;}   else
   if(kolom4==0)   {tombol=78;  beban=14;}   else
    {
    baris2=1;
    baris3=0;      //cek baris3
    delay_us(1);
    if(kolom1==0)   {tombol=71;  beban=7;}   else
    if(kolom2==0)   {tombol=72;  beban=8;}   else
    if(kolom3==0)   {tombol=73;  beban=9;}   else
    if(kolom4==0)   {tombol=79;  beban=15;}   else
      {
      baris3=1;
```

```
        baris4=0;    //cek baris4
        delay_us(1);
        if(kolom1==0)   {tombol=75;   beban=10;}  else
        if(kolom2==0)   {tombol=74;   beban=11;}  else
        if(kolom3==0)   {tombol=76;   beban=12;}  else
        if(kolom4==0)   {tombol=80;   beban=16;}  else
                    tombol=0;
        };
      };
   };
}
}

void main(void)
{PORTB=0x00;
DDRB=0x00;

PORTC=0x00;
DDRC=0x00;

PORTD=0xFF;
DDRD=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;

ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);
```

```
while (1)
    {
    scan_keypad();
    if (tombol!=0)
     {
     printf ("%c%c%c%c",85,90,95,tombol);
     lcd_gotoxy(0,1);
     sprintf(buff,"Lampu %i ", beban);
     lcd_puts(buff);
     }
    };
}
```

LAMPIRAN IV

LISTING PROGRAM MIKROKONTROLER ATMega8 UNIT PENGOLAH DATA

```
#include <mega8.h>
#include <stdio.h>
#include <delay.h>

//I2C Bus functions
#asm
   .equ __i2c_port=0x12 ;PORTD
   .equ __sda_bit=3
   .equ __scl_bit=2
#endasm
#include <i2c.h>


#define ADDA_ADDR 0×90
#define EEPROM_ADDR 0xA0
#define RTC_ADDR 0xD0

char status,data,kepala_1, kepala_2, kepala_3, data_fix, penanda_1,
penanda_2, penanda_3, penanda_4, penanda_5, penanda_6,
penanda_7,
penanda_8, penanda_9, penanda_10, penanda_11, penanda_12,
penanda_13, penanda_14, penanda_15, penanda_16,
penanda_otomatis;


unsigned char data_rtc[8], detik, menit, jam,
jam_1_on, menit_1_on, detik_1_on, jam_1_off, menit_1_off,
detik_1_off,
jam_2_on, menit_2_on, detik_2_on, jam_2_off, menit_2_off,
detik_2_off,
jam_3_on, menit_3_on, detik_3_on, jam_3_off, menit_3_off,
detik_3_off,
jam_4_on, menit_4_on, detik_4_on, jam_4_off, menit_4_off,
detik_4_off,
jam_5_on, menit_5_on, detik_5_on, jam_5_off, menit_5_off,
detik_5_off,
jam_6_on, menit_6_on, detik_6_on, jam_6_off, menit_6_off,
detik_6_off,
jam_7_on, menit_7_on, detik_7_on, jam_7_off, menit_7_off,
detik_7_off,
jam_8_on, menit_8_on, detik_8_on, jam_8_off, menit_8_off,
detik_8_off,
jam_9_on, menit_9_on, detik_9_on, jam_9_off, menit_9_off,
detik_9_off,
jam_10_on, menit_10_on, detik_10_on, jam_10_off, menit_10_off,
detik_10_off,
jam_11_on, menit_11_on, detik_11_on, jam_11_off, menit_11_off,
detik_11_off,
jam_12_on, menit_12_on, detik_12_on, jam_12_off, menit_12_off,
detik_12_off,
jam_13_on, menit_13_on, detik_13_on, jam_13_off, menit_13_off,
detik_13_off,
jam_14_on, menit_14_on, detik_14_on, jam_14_off, menit_14_off,
detik_14_off,
jam_15_on, menit_15_on, detik_15_on, jam_15_off, menit_15_off,
detik_15_off,
jam_16_on, menit_16_on, detik_16_on, jam_16_off, menit_16_off,
detik_16_off;
```

```
void read_rtc(void);
void write_rtc(unsigned char alamat, unsigned char data);
void rtc_init(unsigned char rs,unsigned char sqwe,unsigned char
out);
unsigned char dec2bcd(unsigned char input);
unsigned char bcd2dec(unsigned char input);


//--------definisi beban----------//
#define lampu_1 PORTC.5
#define lampu_2 PORTC.4
#define lampu_3 PORTC.3
#define lampu_4 PORTC.2
#define lampu_5 PORTC.1
#define lampu_6 PORTC.0
#define lampu_7 PORTB.5
#define lampu_8 PORTB.4
#define lampu_9 PORTB.3
#define lampu_10 PORTB.2
#define lampu_11 PORTB.1
#define lampu_12 PORTB.0
#define lampu_13 PORTD.7
#define lampu_14 PORTD.6
#define lampu_15 PORTD.5
#define lampu_16 PORTB.7

#define hidup 1
#define mati 0
```

```
// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
status=UCSRA;
data=UDR;

if ((kepala_1==85)&&(kepala_2==90)&&(kepala_3==95))
   {
   data_fix=data;
   if ((data_fix==33)||(data_fix==49)||(data_fix==65)) penanda_1=1;
   else if ((data_fix==34)||(data_fix==50)||(data_fix==66))
penanda_2=1;
   else if ((data_fix==35)||(data_fix==51)||(data_fix==67))
penanda_3=1;
   else if ((data_fix==36)||(data_fix==52)||(data_fix==68))
penanda_4=1;
   else if ((data_fix==37)||(data_fix==53)||(data_fix==69))
penanda_5=1;
   else if ((data_fix==38)||(data_fix==54)||(data_fix==70))
penanda_6=1;
   else if ((data_fix==39)||(data_fix==55)||(data_fix==71))
penanda_7=1;
   else if ((data_fix==40)||(data_fix==56)||(data_fix==72))
penanda_8=1;
   else if ((data_fix==41)||(data_fix==57)||(data_fix==73))
penanda_9=1;
   else if ((data_fix==42)||(data_fix==58)||(data_fix==74))
penanda_10=1;
```

```
   else        if      ((data_fix==43)||(data_fix==59)||(data_fix==75))
penanda_11=1;
   else        if      ((data_fix==44)||(data_fix==60)||(data_fix==76))
penanda_12=1;
   else        if      ((data_fix==45)||(data_fix==61)||(data_fix==77))
penanda_13=1;
   else        if      ((data_fix==46)||(data_fix==62)||(data_fix==78))
penanda_14=1;
   else        if      ((data_fix==47)||(data_fix==63)||(data_fix==79))
penanda_15=1;
   else        if      ((data_fix==48)||(data_fix==64)||(data_fix==80))
penanda_16=1;

      }
else data_fix=0;
if (data==85)
   {
   kepala_1=data;
   }
else if ((kepala_1==85)&&(data==90))
   {
   kepala_2=data;
   }
else if ((kepala_1==85)&&(kepala_2==90)&&(data==95))
   {
   kepala_3=data;
   }
else
   {
   kepala_1=0;
   kepala_2=0;
   kepala_3=0;
   }

}

void main(void)
{
PORTB=0x00;
DDRB=0b10111111;

PORTC=0x00;
DDRC=0b0111111;

PORTD=0x00;
DDRD=0b11100000;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x33;

// Analog Comparator initialization
```

```
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;


// I2C Bus initialization
i2c_init();
rtc_init(0,1,0);

// Global enable interrupts
#asm("sei")

while (1)
      {
      read_rtc();
      jam=data_rtc[2];
      menit=data_rtc[1];
      detik=data_rtc[0];

      //-------mode mati---------//
      if (data_fix==1)  lampu_1=0;
      if (data_fix==2)  lampu_2=0;
      if (data_fix==3)  lampu_3=0;
      if (data_fix==4)  lampu_4=0;
      if (data_fix==5)  lampu_5=0;
      if (data_fix==6)  lampu_6=0;
      if (data_fix==7)  lampu_7=0;
      if (data_fix==8)  lampu_8=0;
      if (data_fix==9)  lampu_9=0;
      if (data_fix==10) lampu_10=0;
      if (data_fix==11) lampu_11=0;
      if (data_fix==12) lampu_12=0;
      if (data_fix==81) lampu_13=0;
      if (data_fix==14) lampu_14=0;
      if (data_fix==15) lampu_15=0;
      if (data_fix==16) lampu_16=0;

      //-------mode hidup---------//
      if (data_fix==17) lampu_1=1;
      if (data_fix==18) lampu_2=1;
      if (data_fix==19) lampu_3=1;
      if (data_fix==20) lampu_4=1;
      if (data_fix==21) lampu_5=1;
      if (data_fix==22) lampu_6=1;
      if (data_fix==23) lampu_7=1;
      if (data_fix==24) lampu_8=1;
      if (data_fix==25) lampu_9=1;
      if (data_fix==26) lampu_10=1;
      if (data_fix==27) lampu_11=1;
      if (data_fix==28) lampu_12=1;
      if (data_fix==29) lampu_13=1;
      if (data_fix==30) lampu_14=1;
      if (data_fix==31) lampu_15=1;
      if (data_fix==32) lampu_16=1;

      //-----mode hidup 30 menit-------//
      if (data_fix==33)
        {
        if (penanda_1==1)
```

```
    {
    lampu_1=1;

    jam_1_on=jam;
    menit_1_on=menit;
    detik_1_on=detik;

    detik_1_off=detik_1_on;
    menit_1_off=menit_1_on+30;
    if (menit_1_off>59)
       {
       menit_1_off=menit_1_off-60;
       jam_1_off=jam_1_on+1;
       }
    else
       jam_1_off=jam_1_on;

    penanda_1=0;
    }
  }

if (data_fix==34)
  {
  if (penanda_2==1)
    {
    lampu_2=1;

    jam_2_on=jam;
    menit_2_on=menit;
    detik_2_on=detik;

    detik_2_off=detik_2_on;
    menit_2_off=menit_2_on+30;
    if (menit_2_off>59)
       {
       menit_2_off=menit_2_off-60;
       jam_2_off=jam_2_on+1;
       }
    else
       jam_2_off=jam_2_on;

    penanda_2=0;
    }
  }

if (data_fix==35)
  {
  if (penanda_3==1)
    {
    lampu_3=1;

    jam_3_on=jam;
    menit_3_on=menit;
    detik_3_on=detik;

    detik_3_off=detik_3_on;
    menit_3_off=menit_3_on+30;
    if (menit_3_off>59)
       {
       menit_3_off=menit_3_off-60;
```

```
      jam_3_off=jam_3_on+1;
      }
    else
      jam_3_off=jam_3_on;

    penanda_3=0;
    }
  }

if (data_fix==36)
  {
  if (penanda_4==1)
    {
    lampu_4=1;

    jam_4_on=jam;
    menit_4_on=menit;
    detik_4_on=detik;

    detik_4_off=detik_4_on;
    menit_4_off=menit_4_on+30;
    if (menit_4_off>59)
      {
      menit_4_off=menit_4_off-60;
      jam_4_off=jam_4_on+1;
      }
    else
      jam_4_off=jam_4_on;

    penanda_4=0;
      }
    }

if (data_fix==37)
  {
  if (penanda_5==1)
    {
    lampu_5=1;

    jam_5_on=jam;
    menit_5_on=menit;
    detik_5_on=detik;

    detik_5_off=detik_5_on;
    menit_5_off=menit_5_on+30;
    if (menit_5_off>59)
      {
      menit_5_off=menit_5_off-60;
      jam_5_off=jam_5_on+1;
      }
    else
      jam_5_off=jam_5_on;

    penanda_5=0;
    }
  }

if (data_fix==38)
  {
  if (penanda_6==1)
```

```
    {
    lampu_6=1;

    jam_6_on=jam;
    menit_6_on=menit;
    detik_6_on=detik;

    detik_6_off=detik_6_on;
    menit_6_off=menit_6_on+30;
    if (menit_6_off>59)
       {
       menit_6_off=menit_6_off-60;
       jam_6_off=jam_6_on+1;
       }
    else
       jam_6_off=jam_6_on;

    penanda_6=0;
    }
  }

if (data_fix==39)
  {
  if (penanda_7==1)
    {
    lampu_7=1;

    jam_7_on=jam;
    menit_7_on=menit;
    detik_7_on=detik;

    detik_7_off=detik_7_on;
    menit_7_off=menit_7_on+30;
    if (menit_7_off>59)
       {
       menit_7_off=menit_7_off-60;
       jam_7_off=jam_7_on+1;
       }
    else
       jam_7_off=jam_7_on;

    penanda_7=0;
    }
  }

if (data_fix==40)
  {
  if (penanda_8==1)
    {
    lampu_8=1;

    jam_8_on=jam;
    menit_8_on=menit;
    detik_8_on=detik;

    detik_8_off=detik_8_on;
    menit_8_off=menit_8_on+30;
    if (menit_8_off>59)
       {
       menit_8_off=menit_8_off-60;
```

```
      jam_8_off=jam_8_on+1;
      }
    else
      jam_8_off=jam_8_on;

    penanda_8=0;
    }
  }

if (data_fix==41)
  {
  if (penanda_9==1)
    {
    lampu_9=1;

    jam_9_on=jam;
    menit_9_on=menit;
    detik_9_on=detik;

    detik_9_off=detik_9_on;
    menit_9_off=menit_9_on+30;
    if (menit_9_off>59)
      {
      menit_9_off=menit_9_off-60;
      jam_9_off=jam_9_on+1;
      }
    else
      jam_9_off=jam_9_on;

    penanda_9=0;
      }
    }

if (data_fix==42)
  {
  if (penanda_10==1)
    {
    lampu_10=1;

    jam_10_on=jam;
    menit_10_on=menit;
    detik_10_on=detik;

    detik_10_off=detik_10_on;
    menit_10_off=menit_10_on+30;
    if (menit_10_off>59)
      {
      menit_10_off=menit_10_off-60;
      jam_10_off=jam_10_on+1;
      }
    else
      jam_10_off=jam_10_on;

    penanda_10=0;
    }
  }

if (data_fix==43)
  {
  if (penanda_11==1)
```

```
    {
    lampu_11=1;

    jam_11_on=jam;
    menit_11_on=menit;
    detik_11_on=detik;

    detik_11_off=detik_11_on;
    menit_11_off=menit_11_on+30;
    if (menit_11_off>59)
       {
       menit_11_off=menit_11_off-60;
       jam_11_off=jam_11_on+1;
       }
    else
       jam_11_off=jam_11_on;

    penanda_11=0;
    }
  }

if (data_fix==44)
  {
  if (penanda_12==1)
    {
    lampu_12=1;

    jam_12_on=jam;
    menit_12_on=menit;
    detik_12_on=detik;

    detik_12_off=detik_12_on;
    menit_12_off=menit_12_on+30;
    if (menit_12_off>59)
       {
       menit_12_off=menit_12_off-60;
       jam_12_off=jam_12_on+1;
       }
    else
       jam_12_off=jam_12_on;

    penanda_12=0;
    }
  }

if (data_fix==45)
  {
  if (penanda_13==1)
    {
    lampu_13=1;

    jam_13_on=jam;
    menit_13_on=menit;
    detik_13_on=detik;

    detik_13_off=detik_13_on;
    menit_13_off=menit_13_on+30;
    if (menit_13_off>59)
       {
       menit_13_off=menit_13_off-60;
```

```
      jam_13_off=jam_13_on+1;
      }
    else
      jam_13_off=jam_13_on;

    penanda_13=0;
    }
  }

if (data_fix==46)
  {
  if (penanda_14==1)
    {
    lampu_14=1;

    jam_14_on=jam;
    menit_14_on=menit;
    detik_14_on=detik;

    detik_14_off=detik_14_on;
    menit_14_off=menit_14_on+30;
    if (menit_14_off>59)
      {
      menit_14_off=menit_14_off-60;
      jam_14_off=jam_14_on+1;
      }
    else
      jam_14_off=jam_1_on;

    penanda_14=0;
      }
  }

if (data_fix==47)
  {
  if (penanda_15==1)
    {
    lampu_15=1;

    jam_15_on=jam;
    menit_15_on=menit;
    detik_15_on=detik;

    detik_15_off=detik_15_on;
    menit_15_off=menit_15_on+30;
    if (menit_15_off>59)
      {
      menit_15_off=menit_15_off-60;
      jam_15_off=jam_15_on+1;
      }
    else
      jam_15_off=jam_15_on;

    penanda_15=0;
    }
  }

if (data_fix==48)
  {
  if (penanda_16==1)
```

```
    {
    lampu_16=1;

    jam_16_on=jam;
    menit_16_on=menit;
    detik_16_on=detik;

    detik_16_off=detik_16_on;
    menit_16_off=menit_16_on+30;
    if (menit_16_off>59)
       {
       menit_16_off=menit_16_off-60;
       jam_16_off=jam_16_on+1;
       }
    else
       jam_16_off=jam_16_on;

    penanda_16=0;
    }
  }

//-----mode hidup 1 jam-------//
if (data_fix==49)
  {
  if (penanda_1==1)
     {
     lampu_1=1;

     jam_1_on=jam;
     menit_1_on=menit;
     detik_1_on=detik;

     detik_1_off=detik_1_on;
     menit_1_off=menit_1_on;
     jam_1_off=jam_1_on+1;
     if (jam_1_off>23)
        {
        jam_1_off=jam_1_off-24;
        }
     penanda_1=0;
     }
  }

if (data_fix==50)
  {
  if (penanda_2==1)
     {
     lampu_2=1;

     jam_2_on=jam;
     menit_2_on=menit;
     detik_2_on=detik;

     detik_2_off=detik_2_on;
     menit_2_off=menit_2_on;
     jam_2_off=jam_2_on+1;
     if (jam_2_off>23)
        {
        jam_2_off=jam_2_off-24;
        }
```

```
    penanda_2=0;
    }
  }

if (data_fix==51)
  {
  if (penanda_3==1)
    {
    lampu_3=1;

    jam_3_on=jam;
    menit_3_on=menit;
    detik_3_on=detik;

    detik_3_off=detik_3_on;
    menit_3_off=menit_3_on;
    jam_3_off=jam_3_on+1;
    if (jam_3_off>23)
       {
       jam_3_off=jam_3_off-24;
       }
    penanda_3=0;
    }
  }
if (data_fix==52)
  {
  if (penanda_4==1)
    {
    lampu_4=1;

    jam_4_on=jam;
    menit_4_on=menit;
    detik_4_on=detik;

    detik_4_off=detik_4_on;
    menit_4_off=menit_4_on;
    jam_4_off=jam_4_on+1;
    if (jam_4_off>23)
       {
       jam_4_off=jam_4_off-24;
       }
    penanda_4=0;
    }
  }
if (data_fix==53)
  {
  if (penanda_5==1)
    {
    lampu_5=1;

    jam_5_on=jam;
    menit_5_on=menit;
    detik_5_on=detik;

    detik_5_off=detik_5_on;
    menit_5_off=menit_5_on;
    jam_5_off=jam_5_on+1;
    if (jam_5_off>23)
       {
       jam_5_off=jam_5_off-24;
```

```
      }
    penanda_5=0;
    }
  }
if (data_fix==54)
  {
  if (penanda_6==1)
    {
    lampu_6=1;

    jam_6_on=jam;
    menit_6_on=menit;
    detik_6_on=detik;

    detik_6_off=detik_6_on;
    menit_6_off=menit_6_on;
    jam_6_off=jam_6_on+1;
    if (jam_6_off>23)
      {
      jam_6_off=jam_6_off-24;
      }
    penanda_6=0;
    }
  }
if (data_fix==55)
  {
  if (penanda_7==1)
    {
    lampu_7=1;

    jam_7_on=jam;
    menit_7_on=menit;
    detik_7_on=detik;

    detik_7_off=detik_7_on;
    menit_7_off=menit_7_on;
    jam_7_off=jam_7_on+1;
    if (jam_7_off>23)
      {
      jam_7_off=jam_7_off-24;
      }
    penanda_7=0;
    }
  }
if (data_fix==56)
  {
  if (penanda_8==1)
    {
    lampu_8=1;

    jam_8_on=jam;
    menit_8_on=menit;
    detik_8_on=detik;

    detik_8_off=detik_8_on;
    menit_8_off=menit_8_on;
    jam_8_off=jam_8_on+1;
    if (jam_8_off>23)
      {
      jam_8_off=jam_8_off-24;
```

```
      }
    penanda_8=0;
    }
  }
if (data_fix==57)
  {
  if (penanda_9==1)
    {
    lampu_9=1;

    jam_9_on=jam;
    menit_9_on=menit;
    detik_9_on=detik;

    detik_9_off=detik_9_on;
    menit_9_off=menit_9_on;
    jam_9_off=jam_9_on+1;
    if (jam_9_off>23)
      {
      jam_9_off=jam_9_off-24;
      }
    penanda_9=0;
    }
  }
if (data_fix==58)
  {
  if (penanda_10==1)
    {
    lampu_10=1;

    jam_10_on=jam;
    menit_10_on=menit;
    detik_10_on=detik;

    detik_10_off=detik_10_on;
    menit_10_off=menit_10_on;
    jam_10_off=jam_10_on+1;
    if (jam_10_off>23)
      {
      jam_10_off=jam_10_off-24;
      }
    penanda_10=0;
    }
  }
if (data_fix==59)
  {
  if (penanda_11==1)
    {
    lampu_11=1;

    jam_11_on=jam;
    menit_11_on=menit;
    detik_11_on=detik;

    detik_11_off=detik_11_on;
    menit_11_off=menit_11_on;
    jam_11_off=jam_11_on+1;
    if (jam_11_off>23)
      {
      jam_11_off=jam_11_off-24;
```

```
      }
    penanda_11=0;
    }
  }
if (data_fix==60)
  {
  if (penanda_12==1)
    {
    lampu_12=1;

    jam_12_on=jam;
    menit_12_on=menit;
    detik_12_on=detik;

    detik_12_off=detik_12_on;
    menit_12_off=menit_12_on;
    jam_12_off=jam_12_on+1;
    if (jam_12_off>23)
      {
      jam_12_off=jam_12_off-24;
      }
    penanda_12=0;
    }
  }
if (data_fix==61)
  {
  if (penanda_13==1)
    {
    lampu_13=1;

    jam_13_on=jam;
    menit_13_on=menit;
    detik_13_on=detik;

    detik_13_off=detik_13_on;
    menit_13_off=menit_13_on;
    jam_13_off=jam_13_on+1;
    if (jam_13_off>23)
      {
      jam_13_off=jam_13_off-24;
      }
    penanda_13=0;
    }
  }
if (data_fix==62)
  {
  if (penanda_14==1)
    {
    lampu_14=1;

    jam_14_on=jam;
    menit_14_on=menit;
    detik_14_on=detik;

    detik_14_off=detik_14_on;
    menit_14_off=menit_14_on;
    jam_14_off=jam_14_on+1;
    if (jam_14_off>23)
      {
      jam_14_off=jam_14_off-24;
```

```
      }
    penanda_14=0;
    }
  }
if (data_fix==63)
  {
  if (penanda_15==1)
    {
    lampu_15=1;

    jam_15_on=jam;
    menit_15_on=menit;
    detik_15_on=detik;

    detik_15_off=detik_15_on;
    menit_15_off=menit_15_on;
    jam_15_off=jam_15_on+1;
    if (jam_15_off>23)
      {
      jam_15_off=jam_15_off-24;
      }
    penanda_15=0;
    }
  }
if (data_fix==64)
  {
  if (penanda_16==1)
    {
    lampu_16=1;

    jam_16_on=jam;
    menit_16_on=menit;
    detik_16_on=detik;

    detik_16_off=detik_16_on;
    menit_16_off=menit_16_on;
    jam_16_off=jam_16_on+1;
    if (jam_16_off>23)
      {
      jam_16_off=jam_16_off-24;
      }
    penanda_16=0;
    }
  }

//-----mode hidup 2 jam-------//
if (data_fix==65)
  {
  if (penanda_1==1)
    {
    lampu_1=1;

    jam_1_on=jam;
    menit_1_on=menit;
    detik_1_on=detik;

    detik_1_off=detik_1_on;
    menit_1_off=menit_1_on;
    jam_1_off=jam_1_on+2;
    if (jam_1_off>23)
```

```
      {
      jam_1_off=jam_1_off-24;
      }
    penanda_1=0;
    }
  }

if (data_fix==66)
  {
  if (penanda_2==1)
    {
    lampu_2=1;

    jam_2_on=jam;
    menit_2_on=menit;
    detik_2_on=detik;

    detik_2_off=detik_2_on;
    menit_2_off=menit_2_on;
    jam_2_off=jam_2_on+2;
    if (jam_2_off>23)
      {
      jam_2_off=jam_2_off-24;
      }
    penanda_2=0;
    }
  }

if (data_fix==67)
  {
  if (penanda_3==1)
    {
    lampu_3=1;

    jam_3_on=jam;
    menit_3_on=menit;
    detik_3_on=detik;

    detik_3_off=detik_3_on;
    menit_3_off=menit_3_on;
    jam_3_off=jam_3_on+2;
    if (jam_3_off>23)
      {
      jam_3_off=jam_3_off-24;
      }
    penanda_3=0;
    }
  }

if (data_fix==68)
  {
  if (penanda_4==1)
    {
    lampu_4=1;

    jam_4_on=jam;
    menit_4_on=menit;
    detik_4_on=detik;

    detik_4_off=detik_4_on;
```

```
    menit_4_off=menit_4_on;
    jam_4_off=jam_4_on+2;
    if (jam_4_off>23)
       {
       jam_4_off=jam_4_off-24;
       }
    penanda_4=0;
    }
  }

if (data_fix==69)
  {
  if (penanda_5==1)
    {
    lampu_5=1;

    jam_5_on=jam;
    menit_5_on=menit;
    detik_5_on=detik;

    detik_5_off=detik_5_on;
    menit_5_off=menit_5_on;
    jam_5_off=jam_5_on+2;
    if (jam_5_off>23)
       {
       jam_5_off=jam_5_off-24;
       }
    penanda_5=0;
    }
  }

if (data_fix==70)
  {
  if (penanda_6==1)
    {
    lampu_6=1;

    jam_6_on=jam;
    menit_6_on=menit;
    detik_6_on=detik;

    detik_6_off=detik_6_on;
    menit_6_off=menit_6_on;
    jam_6_off=jam_6_on+2;
    if (jam_6_off>23)
       {
       jam_6_off=jam_6_off-24;
       }
    penanda_6=0;
    }
  }

if (data_fix==71)
  {
  if (penanda_7==1)
    {
    lampu_7=1;

    jam_7_on=jam;
    menit_7_on=menit;
```

```
    detik_7_on=detik;

    detik_7_off=detik_7_on;
    menit_7_off=menit_7_on;
    jam_7_off=jam_7_on+2;
    if (jam_7_off>23)
       {
       jam_7_off=jam_7_off-24;
       }
    penanda_7=0;
    }
  }

if (data_fix==72)
  {
  if (penanda_8==1)
    {
    lampu_8=1;

    jam_8_on=jam;
    menit_8_on=menit;
    detik_8_on=detik;

    detik_8_off=detik_8_on;
    menit_8_off=menit_8_on;
    jam_8_off=jam_8_on+2;
    if (jam_8_off>23)
       {
       jam_8_off=jam_8_off-24;
       }
       penanda_8=0;
       }
    }

if (data_fix==73)
  {
  if (penanda_9==1)
    {
    lampu_9=1;

    jam_9_on=jam;
    menit_9_on=menit;
    detik_9_on=detik;

    detik_9_off=detik_9_on;
    menit_9_off=menit_9_on;
    jam_9_off=jam_9_on+2;
    if (jam_9_off>23)
       {
       jam_9_off=jam_9_off-24;
       }
    penanda_9=0;
    }
  }

if (data_fix==74)
  {
  if (penanda_10==1)
    {
    lampu_10=1;
```

```
    jam_10_on=jam;
    menit_10_on=menit;
    detik_10_on=detik;

    detik_10_off=detik_10_on;
    menit_10_off=menit_10_on;
    jam_10_off=jam_10_on+2;
    if (jam_10_off>23)
       {
       jam_10_off=jam_10_off-24;
       }
    penanda_10=0;
    }
  }

if (data_fix==75)
  {
  if (penanda_11==1)
    {
    lampu_11=1;

    jam_11_on=jam;
    menit_11_on=menit;
    detik_11_on=detik;

    detik_11_off=detik_11_on;
    menit_11_off=menit_11_on;
    jam_11_off=jam_11_on+2;
    if (jam_11_off>23)
       {
       jam_11_off=jam_11_off-24;
       }
    penanda_11=0;
    }
  }

if (data_fix==76)
  {
  if (penanda_12==1)
    {
    lampu_12=1;

    jam_12_on=jam;
    menit_12_on=menit;
    detik_12_on=detik;

    detik_12_off=detik_12_on;
    menit_12_off=menit_12_on;
    jam_12_off=jam_12_on+2;
    if (jam_12_off>23)
       {
       jam_12_off=jam_12_off-24;
       }
    penanda_12=0;
    }
  }

if (data_fix==77)
  {
```

```
 if (penanda_13==1)
    {
    lampu_13=1;

    jam_13_on=jam;
    menit_13_on=menit;
    detik_13_on=detik;

    detik_13_off=detik_13_on;
    menit_13_off=menit_13_on;
    jam_13_off=jam_13_on+2;
    if (jam_13_off>23)
       {
       jam_13_off=jam_13_off-24;
       }
    penanda_13=0;
    }
 }

if (data_fix==78)
 {
 if (penanda_14==1)
    {
    lampu_14=1;

    jam_14_on=jam;
    menit_14_on=menit;
    detik_14_on=detik;

    detik_14_off=detik_14_on;
    menit_14_off=menit_14_on;
    jam_14_off=jam_14_on+2;
    if (jam_14_off>23)
       {
       jam_14_off=jam_14_off-24;
       }
    penanda_14=0;
    }
 }

if (data_fix==79)
 {
 if (penanda_15==1)
    {
    lampu_15=1;

    jam_15_on=jam;
    menit_15_on=menit;
    detik_15_on=detik;

    detik_15_off=detik_15_on;
    menit_15_off=menit_15_on;
    jam_15_off=jam_15_on+2;
    if (jam_15_off>23)
       {
       jam_15_off=jam_15_off-24;
       }
    penanda_15=0;
    }
 }
```

```
if (data_fix==80)
  {
  if (penanda_16==1)
     {
     lampu_16=1;

     jam_16_on=jam;
     menit_16_on=menit;
     detik_16_on=detik;

     detik_16_off=detik_16_on;
     menit_16_off=menit_16_on;
     jam_16_off=jam_16_on+2;
     if (jam_16_off>23)
        {
        jam_16_off=jam_16_off-24;
        }
     penanda_16=0;
     }
  }

//-----mode otomatis-------//
if (data_fix==82)
  {
  penanda_otomatis=1;
  }

//---------mematikan lampu----------------//
if (penanda_otomatis==1)
                                          {
                                          if ((jam>=11)||(menit<=6))
                                             {
                                             lampu_1=1;
                                             lampu_2=1;
                                             lampu_15=1;
                                             }
                                          else
                                             {
                                             lampu_1=0;
                                             lampu_2=0;
                                             lampu_15=0;

                                             }
                                          }
                                       else
                                          {
                                          if
                                    ((jam==jam_1_off)&&(menit==menit_1_off)&&(detik==detik_1_o
                                    ff))
                                             {
                                             lampu_1=0;
                                             penanda_1=1;
                                             }
                                          if
                                    ((jam==jam_2_off)&&(menit==menit_2_off)&&(detik==detik_2_o
                                    ff))
                                             {
                                             lampu_2=0;
```

```
        penanda_2=1;
        }
      if
((jam==jam_3_off)&&(menit==menit_3_off)&&(detik==detik_3_o
ff))
        {
        lampu_3=0;
        penanda_3=1;
        }
       if
((jam==jam_4_off)&&(menit==menit_4_off)&&(detik==detik_4_o
ff))
        {
        lampu_4=0;
        penanda_4=1;
        }
      if
((jam==jam_5_off)&&(menit==menit_5_off)&&(detik==detik_5_o
ff))
        {
        lampu_5=0;
        penanda_5=1;
        }
      if
((jam==jam_6_off)&&(menit==menit_6_off)&&(detik==detik_6_o
ff))
        {
        lampu_6=0;
        penanda_6=1;
        }
```

```
        if
((jam==jam_7_off)&&(menit==menit_7_off)&&(detik==detik_7_o
ff))
          {
          lampu_7=0;
          penanda_7=1;
          }
        if
((jam==jam_8_off)&&(menit==menit_8_off)&&(detik==detik_8_o
ff))
          {
          lampu_8=0;
          penanda_8=1;
          }
        if
((jam==jam_9_off)&&(menit==menit_9_off)&&(detik==detik_9_o
ff))
          {
          lampu_9=0;
          penanda_9=1;
          }
         if
((jam==jam_10_off)&&(menit==menit_10_off)&&(detik==detik_1
0_off))
          {
          lampu_10=0;
          penanda_10=1;
          }
```

```
    if ((jam==jam_11_off)&&(menit==menit_11_off)&&(detik==detik_11_off))
        {
        lampu_11=0;
        penanda_11=1;
        }
    if ((jam==jam_12_off)&&(menit==menit_12_off)&&(detik==detik_12_off))
        {
        lampu_12=0;
        penanda_12=1;
        }
    if ((jam==jam_13_off)&&(menit==menit_13_off)&&(detik==detik_13_off))
        {
        lampu_13=0;
        penanda_13=1;
        }
    if ((jam==jam_14_off)&&(menit==menit_14_off)&&(detik==detik_14_off))
        {
        lampu_14=0;
        penanda_14=1;
        }
    if ((jam==jam_15_off)&&(menit==menit_15_off)&&(detik==detik_15_off))
        {
        lampu_15=0;
        penanda_15=1;
        }
    if ((jam==jam_16_off)&&(menit==menit_16_off)&&(detik==detik_16_off))
        {
        lampu_16=0;
        penanda_16=1;
        }
    }
  };
}

void read_rtc(void){
unsigned char i, tmp_data;

i2c_start();
i2c_write(RTC_ADDR);
i2c_write(0);
i2c_stop();
i2c_start();
i2c_write(RTC_ADDR | 1);
for (i=0; i<2; i++){
```

```
tmp_data = bcd2dec(i2c_read(1));
data_rtc[i]=tmp_data;
}
tmp_data = bcd2dec(i2c_read(0));
data_rtc[2]=tmp_data;
i2c_stop();
}
void write_rtc(unsigned char alamat, unsigned char data){
i2c_start();
i2c_write(RTC_ADDR);
i2c_write(alamat);
if (alamat < 7)
i2c_write(dec2bcd(data));
else
i2c_write(data);
i2c_stop();
}

void rtc_init(unsigned char rs,unsigned char sqwe,unsigned char out)
{
rs&=3;
if (sqwe) rs|=0x10;
if (out) rs|=0x80;
i2c_start();
i2c_write(0xd0);
i2c_write(7);
i2c_write(rs);
i2c_stop();
}
```

```
unsigned char dec2bcd(unsigned char input){
unsigned char tmp_data;

if (input > 9) tmp_data = ((input / 10)*16) + (input % 10);
else
tmp_data = input;
return tmp_data;
}

unsigned char bcd2dec(unsigned char input){
unsigned char tmp_data, tmp1;

tmp_data = input;
tmp1 = tmp_data % 16;
if (tmp_data > 15) tmp_data = tmp_data / 16;
else tmp_data = 0;
tmp_data = (tmp_data * 10)+tmp1;
return tmp_data;
}
```