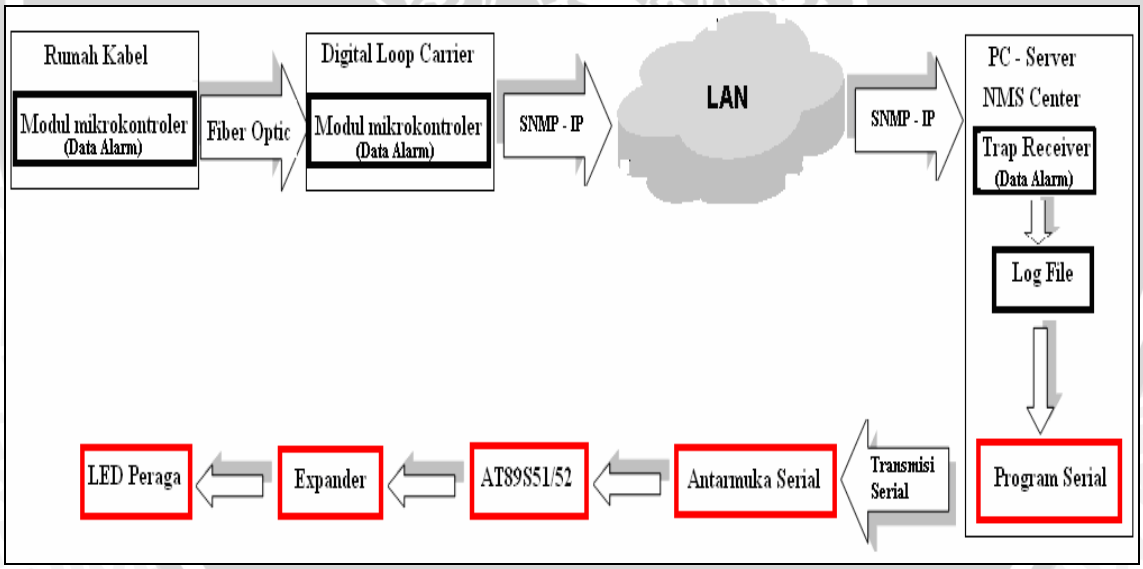
BAB IV
PERENCANAAN DAN PEMBUATAN ALAT

Bagian ini membahas mengenai perencanaan dan pembuatan alat visualisasi alarm *NMS (Network Management System)*. Pembahasan meliputi perencanaan sistem, penetapan spesifikasi alat, dan perencanaan masing-masing blok rangkaian penyusun sistem. Pembuatan sistem ini meliputi perencanaan komunikasi data, perangkat keras *(hardware)* dan perangkat lunak *(software)*.

4.1. Perencanaan Sistem

Pembuatan blok diagram rangkaian alat visualisasi alarm *NMS* ini merupakan dasar dari perencanaan sistem. Diagram blok dari sistem visualisasi alarm *NMS* terdiri dari diagram blok pengolahan data dan diagram blok sistem penerimaan data dari komputer ke lampu peraga ditunjukkan dalam Gambar 4.1. Untuk blok yang berwarna merah adalah sistem atau alat yang akan dirancang.



Gambar 4.1. Diagram blok sistem visualisasi alarm *NMS*.

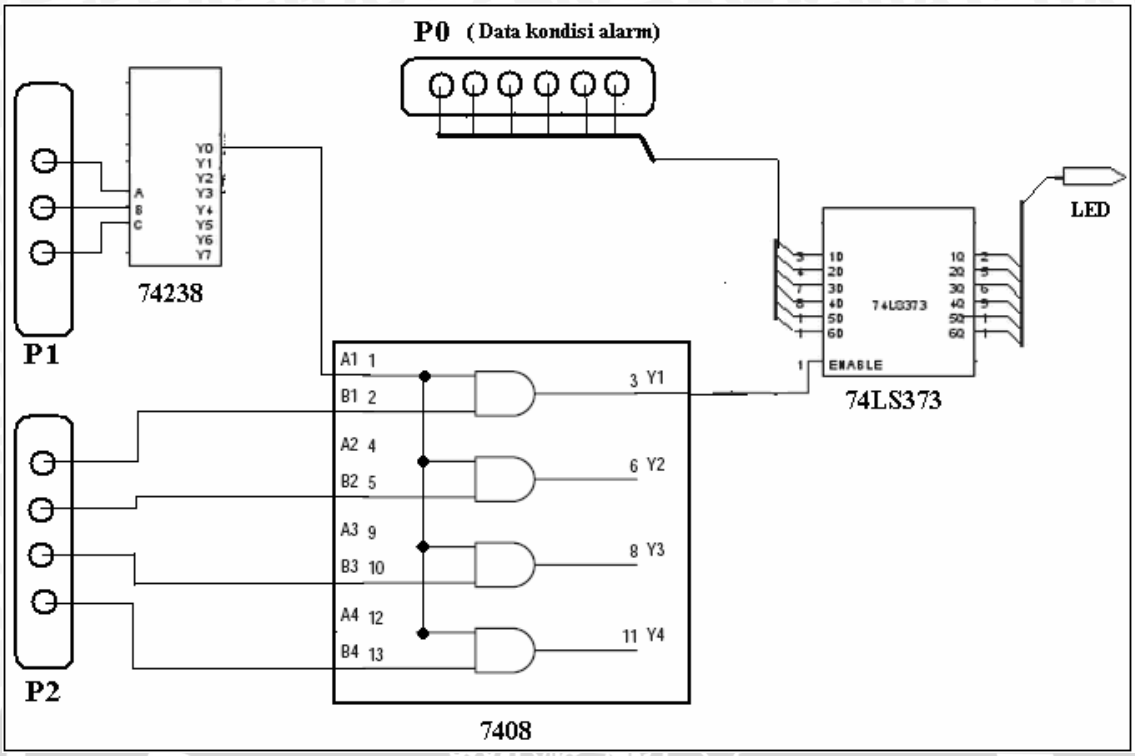
Data alarm dikirim melalui jaringan LAN (*Local Area Network*) akan diolah menjadi data yang sudah dikonversi menjadi *log file* oleh *trap receiver*. *Trap receiver* merupakan fitur dari *SNMP*. File tersebut berupa kode-kode teks yang merepresentasikan alamat dan kondisi perangkat jaringan (RK). Contoh log file dapat dilihat dalam Gambar 4.2.

```
04/06/05 - 13:33:18 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - ClrMJ
04/06/05 - 13:33:18 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - CurMJ
04/06/05 - 13:38:29 - LDY-FRB - Main Power - RST1  - None  - CurMN
04/06/05 - 13:42:06 - LDY-FRB - Main Power - RST1  - None  - ClrMN
04/06/05 - 13:42:52 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - ClrMJ
04/06/05 - 13:42:52 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - CurMJ
04/06/05 - 13:47:39 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - ClrMJ
04/06/05 - 13:47:39 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - CurMJ
04/06/05 - 13:52:25 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - ClrMJ
04/06/05 - 13:52:25 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - CurMJ
04/06/05 - 13:57:12 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - ClrMJ
04/06/05 - 13:57:12 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - CurMJ
04/07/05 - 11:46:30 - BTU FRB - Remote Comm Link Not Present - LET to RST1  - None  - CurCR

04/07/05 - 11:47:18 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - ClrMJ
04/07/05 - 11:47:19 - PDA-SAMPURNA - Loss of Frame - RST1-2-18  - RI-ISDN  - CurMJ
04/07/05 - 11:47:31 - BTU FRB - Loss of Frame - LET-1-22  - FO-XCVR  - CurMJ


04/07/05 - 11:50:05 - BTU FRB - Remote Comm Link Not Present - RST1 to LET  - None  - CurCR

04/07/05 - 11:50:09 - BTU FRB - Remote Comm Link Not Present - LET to RST1  - None  - ClrCR
04/07/05 - 11:50:12 - BTU FRB - Remote Comm Link Not Present - RST1 to LET  - None  - ClrCR
04/07/05 - 12:01:40 - SNT-FRA - Door Open - RST1  - None  - CurMJ




04/07/05 - 12:05:16 - SNT-FRA - Door Open - RST1  - None  - ClrMJ

04/07/05 - 16:45:45 - LDY-FRB - Door Open - RST1  - None  - ClrMJ

04/07/05 - 16:46:14 - LDY-FRB - Door Open - RST1  - None  - CurMJ
```

Gambar 4.2. *Log file*
Sumber: PT. Telkom Kanadatek Malang

File yang berupa kode teks diterima oleh program serial. Program serial ini adalah program antarmuka *Personal Computer* (*PC*) dengan perangkat keras alat visualisasi alarm. Kode teks selanjutnya diubah menjadi karakter tertentu. Karakter ini kemudian dikirim dengan transmisi serial melalui port com 1 atau com 2. Level tegangan dari serial disesuaikan dengan level tegangan TTL. Untuk mengubah level tegangan serial (RS 232) ke level tegangan TTL digunakan MAX 232.

Data karakter yang telah mempunyai level tegangan TTL diterima oleh mikrokontroler. Data karakter ini selanjutnya diolah oleh mikrokontroler. Data karakter yang diolah oleh mikrokontroler ini berisi lokasi dan kondisi dari alarm *NMS*. Mikrokontroler nantinya akan menset port – portnya sesuai dengan kondisi alarm. Port – port mikrokontroler ini mengendalikan 16 *LED* peraga. Karena jumlah port dari mikrokontroler tidak mencukupi dengan kondisi alarm yang akan ditampilkan, maka diperlukan *expander*. *Expander* di sini juga berfungsi untuk menahan data alarm sehingga data tersebut sempat terbaca pada *LED* peraga. Dari *expander* ini hasilnya ditampilkan pada *LED* peraga. Untuk penjelasan singkat mengenai sistem *expander* ditunjukkan dalam Gambar 4.3



Gambar 4.3. Ringkasan Sistem *Expander*

4.2. Spesifikasi Alat

Hal yang perlu diperhatikan sebelum melakukan perencanaan dan pembuatan alat adalah penentuan spesifikasi sistem yang akan dibuat. Adapun spesifikasi sistem yang akan dibuat adalah sebagai berikut:

1. Alat ini merupakan prototipe.
2. Menggunakan mikrokontroler AT89S51 buatan ATMEL sebagai pengolah utama .
3. Data dari *PC* dikirim secara serial menggunakan metode UART *(Universal Asyncronous Receiver/Transmitter)*.
4. Tansmisi data serial menggunakan mode 1, *baud rate* yang digunakan 9600 Hz.
5. *LED* peraga yang digunakan adalah *LED* RGB *(Red Green Blue)* 4 kaki (1 anoda 3 katoda) dengan tipe L200CWRG1KN-4A-IL.
6. Jumlah *LED* yang digunakan adalah 16.
7. Alat ini hanya dapat menampilkan alarm yang sudah terdata saja
8. Alat ini dapat beroperasi selama 12 jam setiap hari.

4.3. Perencanaan Komunikasi Data

Port serial pada AT89S51 bersifat full-duplex, artinya port serial bisa menerima dan mengirim pada waktu bersamaan. Data dari *PC* dikirim secara serial dengan metode *UART (Universal Asyncronous Receiver/Transmitter)* mode 1. Pada mode 1

data dikirim/diterima 10 bit sekaligus, diawali dengan 1 bit start, disusul dengan 8 bit data yang dimulai dari bit yang bobotnya paling kecil (bit 0), diakhiri dengan 1 bit stop.

Kecepatan pengiriman data antara *PC* dan mikrokontroler harus disamakan atau disebut dengan pengaturan *baudrate.* Dalam perencanaan ini *baudrate* yang digunakan adalah 9600 baud. Untuk antar muka tegangan RS232 dengan TTL digunakan MAX232.

Pada perencanaan ini 16 *LED* yang ditampilkan disusun menjadi 4 kolom dan 4 baris. Dan *LED-LED* tersebut merupakan keluaran dari *latch.* Masing-masing *latch* mengendalikan 2 *LED*. Kolom menunjukkan macam kerusakan rumah kabel, baris menunjukkan lokasi rumah kabel. Dari masing-masing *LED* ini mempunyai 3 kondisi. 3 kondisi ini dilambangkan dengan 3 warna dari *LED* (hijau untuk kondisi normal, merah untuk kondisi mayor, biru untuk kondisi minor). Dari 3 warna *LED* tersebut akan nyala satu persatu.

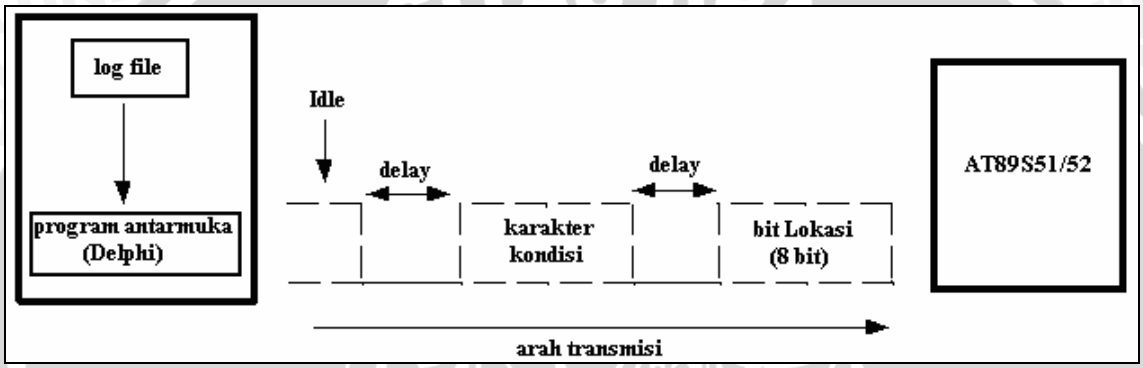
Pertama mikrokontroler menset semua *LED* dalam kondisi normal (hijau). Kemudian mikrokontroler menunggu karakter yang dikirim dari program antarmuka melalui MAX232. Di dalam program mikrokontroler sudah disiapkan kombinasi port untuk masing-masing karakter. Ada dua macam kombinasi port. Pertama kombinasi port lokasi, yang berfungsi untuk menentukan koordinat kolom dan baris pada *LED* peraga. Kedua kombinasi port kondisi, yang berfungsi menentukan kondisi alarm (normal, mayor, minor) yang akan ditampilkan pada *LED* peraga.

4.3.1. Protokol Komunikasi Untuk Pengiriman Karakter Pada RS-232

Protokol pengiriman karakter yang digunakan dalam perancangan ini cukup sederhana. Pada program antarmuka kode-kode teks alarm yang masing-masing kode berisi lokasi dan kondisi alarm yang diterima dari *log file* dipisahkan menurut kelompok lokasi dan kondisi alarm. Dari masing-masing kelompok ini akan diinisialisasikan menjadi karakter – karakter yang identik dengan karakter – karakter yang ada di program mikrokontroler. Karakter yang dikirim oleh program antarmuka ditunjukkan pada Tabel 4.1. dan 4.2. Di sini program antarmuka akan membandingkan kode-kode teks alarm yang diterima sekarang dengan yang diterima sebelumnya apakah terjadi perubahan kode – kode teks alarm. Jika ada perubahan data, maka kode-kode teks alarm yang sudah diubah menjadi karakter seperti pada Tabel 4.1 dan 4.2 akan dikirimkan. Jika tidak ada perubahan kode-kode teks alarm yang diterima dari *log file* maka karakter tidak dikirim. Saat program antarmuka mengirim karakter, mikrokontroler akan menerima sebagai interupsi. Di mikrokontroler karakter yang diterima akan dicocokkan dengan kombinasi port yang menangani karakter tersebut,

kemudian ditampilkan sesuai dengan koordinat baris dan kolom serta kondisi alarm yang dimaksud. Selama tidak ada perubahan pada kombinasi port yang menangani baris dan kolom tersebut maka data akan ditahan.

Data pertama dikirim oleh program antarmuka adalah karakter yang menentukan lokasi alarm. Begitu karakter untuk lokasi alarm dikirim maka mikrokontroler mengaktifkan kombinasi port lokasi yang menangani karakter tersebut. Setelah itu program anatarmuka mengirim karakter untuk kondisi. Kemudian mikrokontroler mengaktifkan kombinasi port kondisi yang menangani karakter tersebut. Untuk lebih jelasnya mengenai format pengiriman data port lokasi yang dikirim oleh program anatarmuka ke mikrokontroler dapat dilihat dalam Gambar 4.3.b.



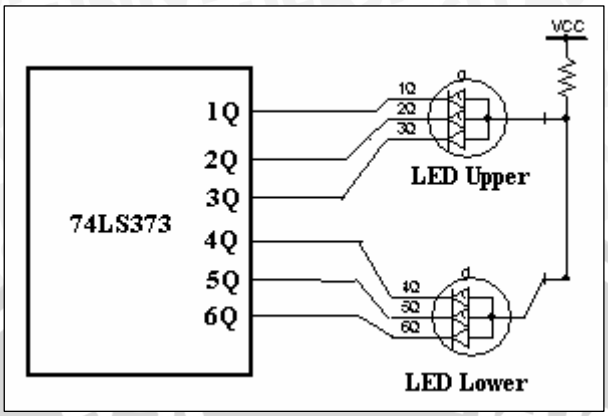
Gambar 4.3.b. Format Pengiriman Karakter

Untuk *delay* mempunyai nilai maksimal 0,01 detik. Sedangkan untuk *idle* mempunyai nilai minimal 0,1 detik.

Tabel 4.1. Format data port lokasi

Kombinasi *Port* lokasi (*Port* 1 dan *Port* 2)							***LED* yang dikontrol**
P2.3	**P2.2**	**P2.1**	**P2.0**	**P1.2**	**P1.1**	**P1.0**	
0	0	0	1	0	0	0	*LED* kolom 1 baris 1 dan 2
0	0	1	0	0	0	0	*LED* kolom 2 baris 1 dan 2
0	1	0	0	0	0	0	*LED* kolom 3 baris 1 dan 2
1	0	0	0	0	0	0	*LED* kolom 4 baris 1 dan 2
0	0	0	1	0	0	1	*LED* kolom 1 baris 3 dan 4
0	0	1	0	0	0	1	*LED* kolom 2 baris 3 dan 4
0	1	0	0	0	0	1	*LED* kolom 3 baris 3 dan 4
1	0	0	0	0	0	1	*LED* kolom 4 baris 3 dan 4

Masing-masing *Latch* mengendalikan dua *LED* seperti dalam Gambar 4.4. Untuk *LED* dari 1Q, 2Q, 3Q (keluaran *Latch*) dinamakan *LED upper*. Untuk *LED* dari 4Q, 5Q, 6Q (keluaran *Latch*) dinamakan *LED lower.*



Gambar 4.4. Keterangan *LED Upper* dan *LED Lower*

Tabel 4.2. Format data port kondisi

Karakter	Keluaran Mikrokontroller (Port 0)					
	Untuk *LED upper*			Untuk *LED lower*		
	Merah	Hijau	Biru	Merah	Hijau	Biru
	P0.0	P0.1	P0.2	P0.3	P0.4	P0.5
r	0	1	1	0	1	1
s	1	0	1	0	1	1
t	1	1	0	0	1	1
u	0	1	1	1	0	1
v	1	0	1	1	0	1
w	1	1	0	1	0	1
x	0	1	1	1	1	0
y	1	0	1	1	1	0
z	1	1	0	1	1	0

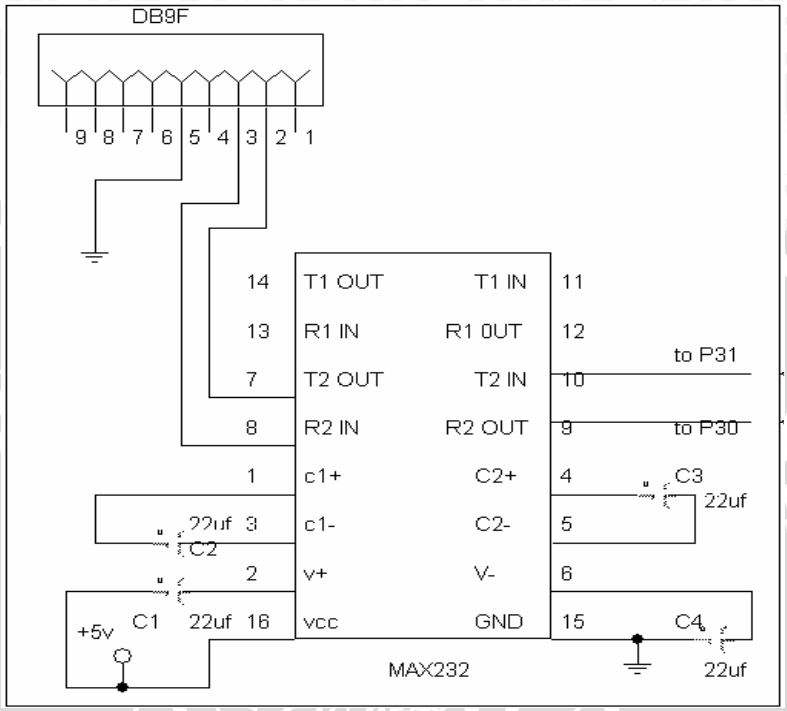
Port P0.0, P0.1, P0.2 mengendalikan *LED upper*. Port P0.3, P0.4, P0.5 mengendalikan *LED lower.* Warna *LED* ditentukan oleh salah satu kaki *LED* aktif *low.*

4.4. Perencanaan Perangkat Keras (Hardware)

4.4.1 Rangkaian Antarmuka Serial

Antarmuka serial digunakan untuk menyesuaikan level tegangan RS 232 dengan TTL dan sebaliknya. Dalam perencanaan ini digunakan IC MAX232. MAX 232 ini

mempunyai dua charge pump internal untuk mengubah tegangan +5 V ke +10 V yang sesuai untuk operasi RS 232. Converter pertama (Voltage doubler) untuk menggandakan tegangan +5 V menjadi –10 V. Pengaturan komunikasi datanya diatur secara perangkat lunak. Rangkaian antarmuka serial yang digunakan dapat dilihat dalam Gambar 4.5.



Gambar 4.5.. Rangkaian antarmuka serial

Sumber: Putra, 2003

4.4.2 Rangkaian Sistem Mikrokontroler

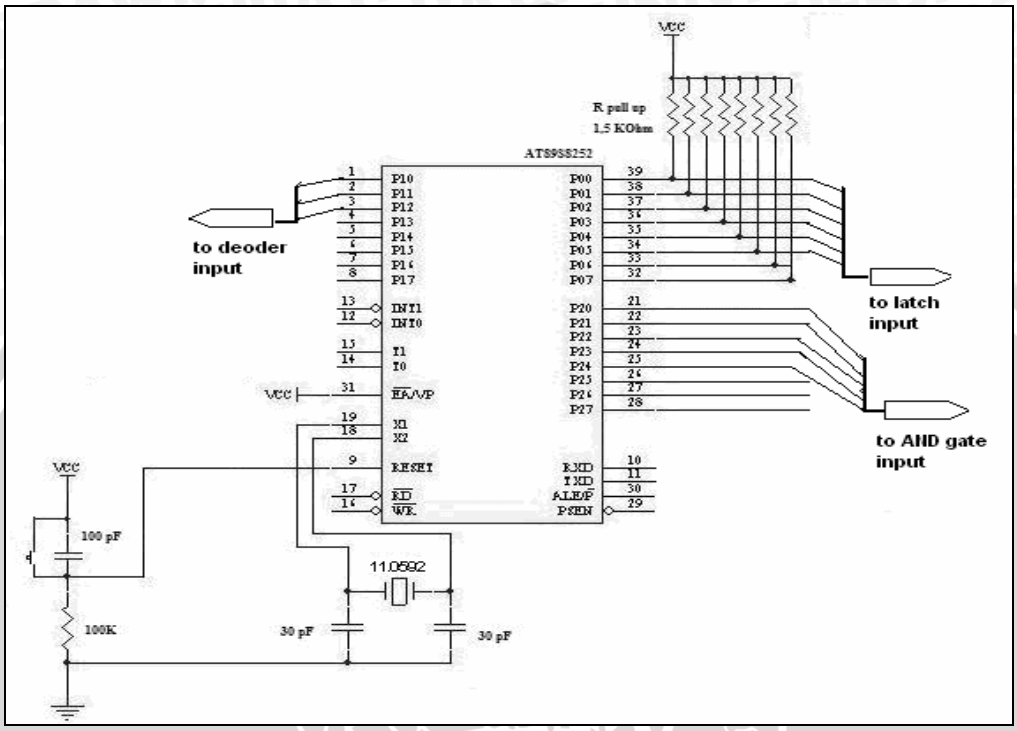
Pada alat ini digunakan mikrokontroler AT89S51 sebagai pusat dari pengolah data dan pengendali peralatan lainnya. Gambar 4.6. adalah diagram skematik dari mikrokontroler AT89S51. Dari *datasheet* AT89S51 diketahui bahwa besarnya I_{OL} secara tipikal untuk setiap pin pada port 0 adalah 3,2 mA dan besar I_{OH} -0,8 mA. Sedangkan untuk port 1, 2, dan 3 besarnya I_{OL} adalah 1,6 mA, I_{OH} nya -0,06 mA.

Pada rangkaian di atas, karena port 0 bersifat *open drain*, maka ketika mengeluarkan logika 0 maka akan terbaca sebagai logika ambang. Untuk itu dibutuhkan resistor *pull up* agar bisa dipastikan memberikan logika 0. Ketika memberikan logika 0 maka tegangan keluaran (V_{OL}) seharusnya adalah 0,5 V. Karena arus yang dibutuhkan ketika logika 0 (I_{OL}) adalah 3,2 mA dan catu daya yang digunakan (Vcc) adalah 5V, maka nilai R *pull up* yang dibutuhkan adalah:

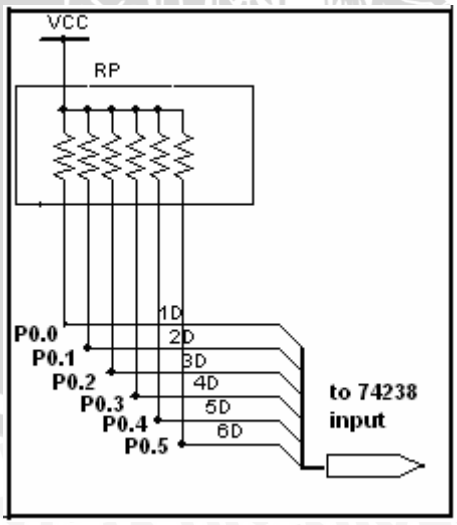
$$R_{pull\_up} = \frac{Vcc - V_{OL}}{I_{OL}}$$

$$R_{pull\_up} = \frac{5V - 0{,}5V}{3{,}2mA}$$

$$R_{pull\_up} = 1406{,}25\ \Omega \approx 1{,}5k\Omega$$



Gambar 4.6. Diagram Skematik Mikrokontroler AT89S51



Gambar 4.7. Diagram Skematik rangkaian *pullup*

Mikrokontroler AT89S51 memiliki 4 port dengan 32 jalur/pin yang dapat diprogram sebagai masukan maupun keluaran, pada perancangan ini pin-pin yang digunakan dapat dilihat dalam Tabel 4.3.

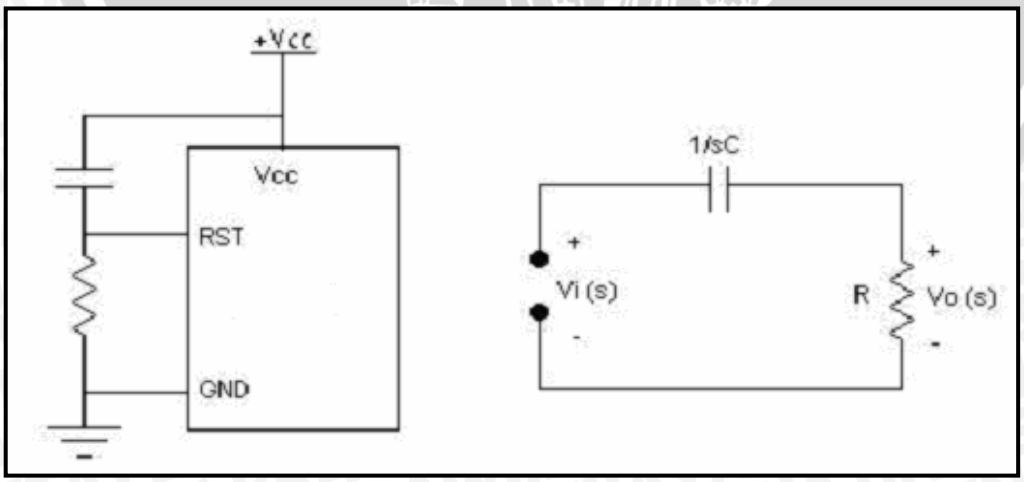
Tabel 4.3. Pin-pin mikrokontroler yang difungsikan

Port 0	sebagai jalur data pada semua IC 74LS373
Pin 1.0	dihubungkan dengan pin A pada IC 74238
Pin 1.1	dihubungkan dengan pin B pada IC 74238
Pin 1.2	dihubungkan dengan pin C pada IC 74238
Pin 2.0	dihubungkan dengan pin B1 pada semua IC 7408
Pin 2.1	dihubungkan dengan pin B2 pada semua IC 7408
Pin 2.2	dihubungkan dengan pin B3 pada semua IC 7408
Pin 2.3	dihubungkan dengan pin B4 pada semua IC 7408
Pin 3.0	dihubungkan dengan R2 out IC MAX232 (komunikasi serial)
Pin 3.1	dihubungkan dengan T2 in IC MAX232 (komunikasi serial)

4.2.1.1. Rangkaian Power – ON Reset

Rangkaian reset digunakan untuk mereset mikrokontroler AT89S51 ketika mikrokontroler pertama kali diaktifkan maupun ketika tombol *reset* ditekan. Pada saat pertama kali mikrokontroler diaktifkan, maka kapasitor C akan terhubung singkat. Arus mengalir dari V_{CC} langsung ke pin Reset, sehingga pin tersebut berlogika 1 (*high*). Kapasitor akan terisi sampai tegangan pada kapasitor mencapai Vcc, otomatis tegangan pada RST akan turun menjadi logika 0 dan proses *reset* selesai. *Pin Reset* diberi logika 1 (*high*) selama sekurangnya dua siklus mesin (24 periode osilator).

Dalam Gambar 4.8. dapat diketahui bahwa untuk mengaktifkan sinyal reset ini, maka kapasitor dihubungkan dengan V_{CC} dan sebuah resistor yang dihubungkan ke *ground*.



Gambar 4.8. Rangkaian *Reset* dan Rangkaian Setaranya

Sumber: Putra, 2003

Karena kristal yang digunakan mempunyai frekuensi sebesar 11,0592 MHz, maka satu periode membutuhkan waktu sebesar:

$$T = \frac{1}{f_{XTAL}} = \frac{1}{11{,}0592\,MHz}\,s = 9{,}04 \text{ x } 10^{-8} \quad \text{s}$$

Sehingga waktu minimal logika tinggi yang dibutuhkan untuk mereset mikrokontroler adalah:

$t_{reset(min)}$ = T * periode yang dibutuhkan

= $9{,}04 \text{ x } 10^{-8}$ * 24 = 2 µs

Jadi mikrokontroler membutuhkan waktu minimal 2 µs untuk mereset. Waktu minimal inilah yang dijadikan pedoman untuk menentukan nilai R dan C. Dari rangkaian setara yang terdapat dalam Gambar 4.8. diperoleh:

$$\text{Vo} = \frac{R}{R + \frac{1}{sC}} * Vi$$

$$\text{Vo} = \frac{sCR}{sCR + 1} * Vi$$

dengan tegangan Vi adalah Vcc yaitu 5 V, dalam fungsi Laplace adalah $\frac{5}{s}$ sehingga:

$$\text{Vo} = \frac{5}{s} * \frac{sCR}{sCR+1} = 5 * \frac{CR}{sCR+1} = 5 * \left[\frac{1}{s + 1/RC}\right]$$

$$\text{Vo} = 5 * e^{\frac{-t}{RC}}$$

$$\frac{5}{Vo} = e^{\frac{t}{RC}}$$

$$\ln \frac{5}{Vo} = \frac{t}{RC}$$

$$\text{t} = \text{RC}\left[\ln \frac{5}{Vo}\right]$$

Nilai Vo adalah tegangan logika nominal yang diijinkan oleh pin RST (*Datasheet* Atmel AT89S51), nilai Vo didapat dari:

$\text{Vo} = 0{,}7 \text{ x } V_{CC}$

$\text{Vo} = 0{,}7 \text{ x } 5 \text{ V} = 3{,}5 \text{ V}$

Maka: $\text{t} = \text{RC} * \left[\ln \frac{5}{3{,}5}\right]$

$$t = RC*0{,}357$$

$$t = RC * \ln\frac{5}{V_O}$$

Nilai RC minimal yang dijadikan pedoman adalah:

$$t = RC * \ln\frac{5}{3{,}5}$$

$$2\mu s = RC * 0{,}357$$

$$RC = 5{,}603 \cdot 10^{-6}$$

Jika diambil nilai R sebesar 10 kΩ dan C sebesar 10 µF, maka:

$$t = 0{,}357 * RC$$

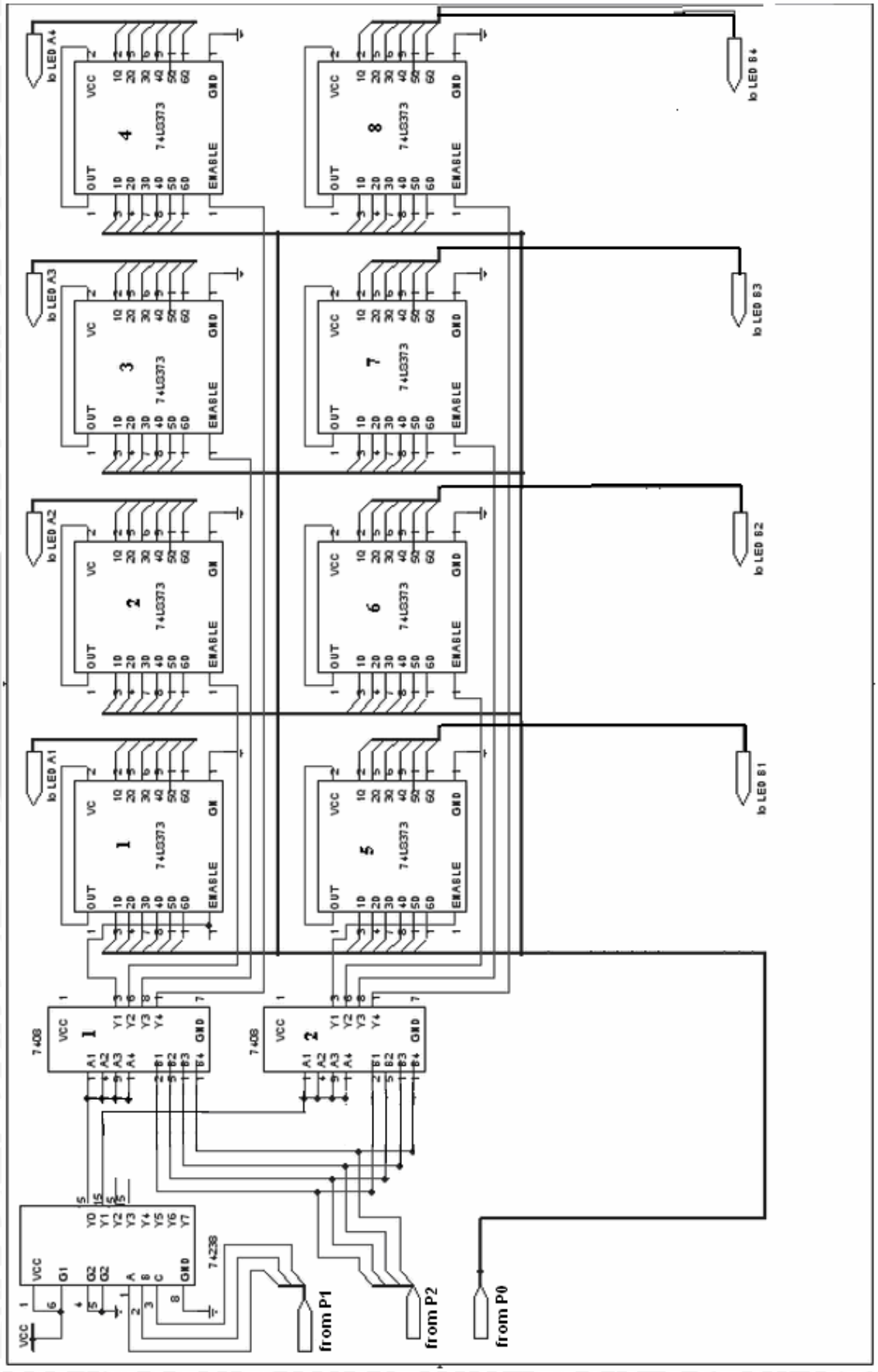
$$t = 0{,}357 * 10^{-5} * 10^{4}$$

$$t = 35{,}7ms$$

Jadi dengan nilai komponen R = 10 kΩ, dan C = 10 µF sudah dapat memenuhi syarat minimal waktu yang dibutuhkan oleh mikrokontroler

4.4.3 Rangkaian Sistem *Expander*

Sistem *expander* pada perencanaan ini digunakan untuk mengimbangi antara jumlah port dari mikrokontroler dengan jumlah LED yang diinginkan, sekaligus berfungsi sebagai penahan data. Sistem *expander* terdiri dari *decoder,* gerbang AND, dan *Latch. Decoder* yang digunakan adalah 74238. Gerbang AND yang digunakan adalah 7408. *Latch* yang digunakan 74LS373. *Decoder* dan gerbang AND di sini menentukan kombinasi lokasi alarm. Sedangkan latch menentukan kombinasi kondisi sekaligus penahan data kondisi alarm. Pada blok *expander* di perencanaan ini yang perlu diperhatikan adalah besarnya I_{IL} dan I_{IH} dari IC 74LS373. Untuk masing – masing IC 74LS373 besarnya I_{IL} maksimal adalah -0,4 mA, I_{IH} maksimalnya 20 µA. Skema sistem *expander* dapat dilihat dalam Gambar 4.9.



Gambar 4.9. Skema sistem *expander*

4.4.3.1 Konfigurasi Pin-Pin pada Rangkaian *Expander*

Pada perencanaan ini pin – pin masing komponen yang digunakan pada sistem *expander* untuk IC 74238 dapat dilihat pada Tabel 4.4, untuk IC 7408 dapat dilihat pada Tabel 4.5.

Tabel 4.4. Hubungan Pin-Pin IC 74238

pin A	dihubungkan dengan pin 1.0 AT89S51
pin B	dihubungkan dengan pin 1.1 AT89S51
pin C	dihubungkan dengan pin 1.2 AT89S51
pin Y0	dihubungkan dengan pin A1,A2, A3, A4 IC 7408 ke 1
pin Y1	dihubungkan dengan pin A1,A2, A3, A4 IC 7408 ke 2
pin G1	dihubungkan ke VCC
pin G2A	dihubungkan ke *ground*
pin G2B	dihubungkan ke *ground*

Tabel 4.5. Hubungan Pin-Pin IC 7408

pin Y1	dihubungkan ke pin *enable* IC 74LS373 ke 1
pin Y2	dihubungkan ke pin *enable* IC 74LS373 ke 2
pin Y3	dihubungkan ke pin *enable* IC 74LS373 ke 3
pin Y4	dihubungkan ke pin *enable* IC 74LS373 ke 4

*IC 7408 lainnya sama dengan Tabel 4.4.

Untuk semua IC 74LS373 pin-pin keluaran (1Q sampai 6Q) dihubungkan dengan rangkaian *LED* peraga.

Kemudian konfigurasi alamat dekoder dari rangkaian *expader* yang nantinya menentukan kombinasi lokasi dari alarm *NMS* dapat dilihat pada Tabel 4.6.

Tabel 4.6. Alamat Dekoder dari Rangkaian *Expander*

IC 74238 (masukan)			**IC 7408** (masukan)				**IC 74LS373** (kondisi *enable*)							
A	B	C	B1	B2	B3	B4	E1	E2	E3	E4	E5	E6	E7	E8
1	0	0	**1**	0	0	0	**1**	0	0	0	0	0	0	0
1	0	0	0	**1**	0	0	0	**1**	0	0	0	0	0	0
1	0	0	0	0	**1**	0	0	0	**1**	0	0	0	0	0
1	0	0	0	0	0	**1**	0	0	0	**1**	0	0	0	0
0	**1**	0	**1**	0	0	0	0	0	0	0	**1**	0	0	0
0	**1**	0	0	**1**	0	0	0	0	0	0	0	**1**	0	0
0	**1**	0	0	0	**1**	0	0	0	0	0	0	0	**1**	0
0	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	**1**

*A, B, C : pin – pin masukan dari IC 74238

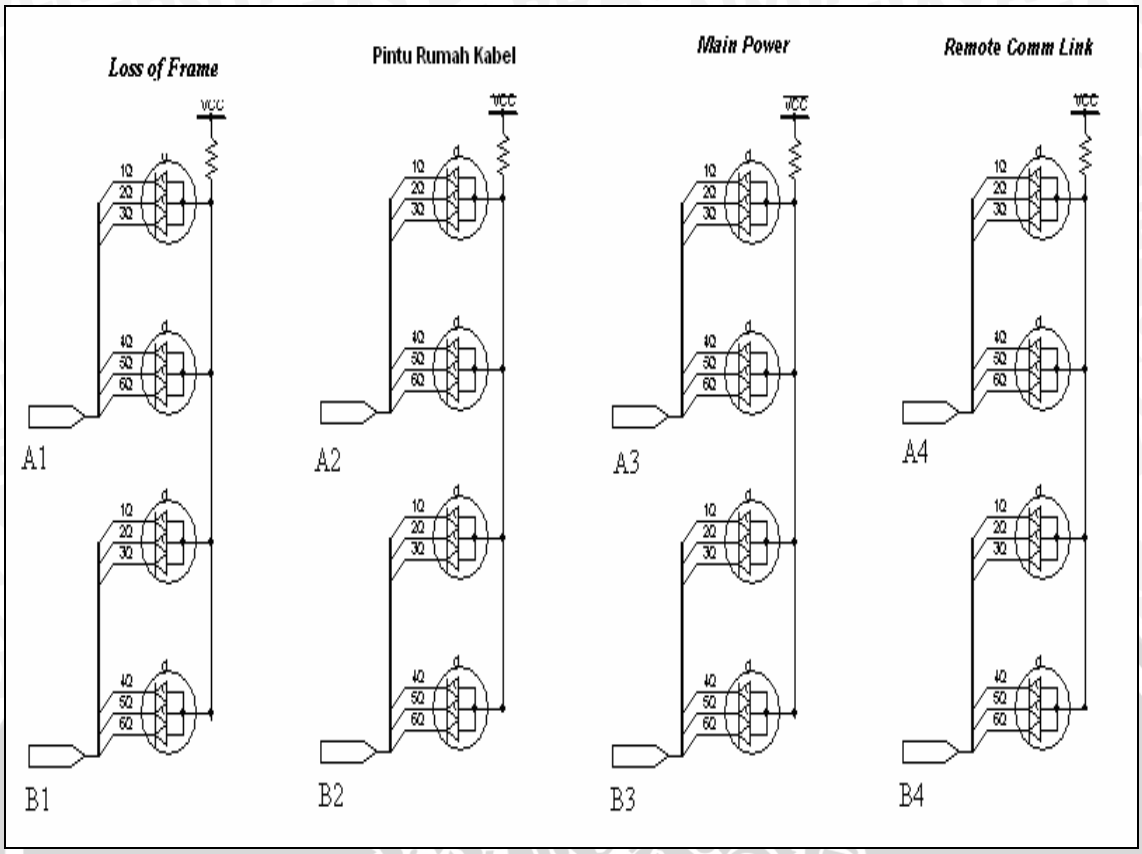
**B1 sampai B4 : pin – pin masukan dari IC 7408

***E1 sampai E8 : pin – pin *enable* dari IC 74LS373

Pada perencanaan ini menurut Tabel 4.4, A1 sampai A4 ditentukan dari keluaran IC 7408. Sedangkan masukan IC 7408 dikendalikan oleh AT89SS51 melalui port 1. B1 sampai B4 dikendalikan oleh AT89SS51 melalui port 2. Kemudian untuk *enable* (E1 sampai E8 jika berlogika 1 maka data masukan dari IC 74LS373 akan dikeluarakan. Perubahan logika *enable* IC 74LS373 akan menentukan koordinat LED yang mengalami perubahan data.

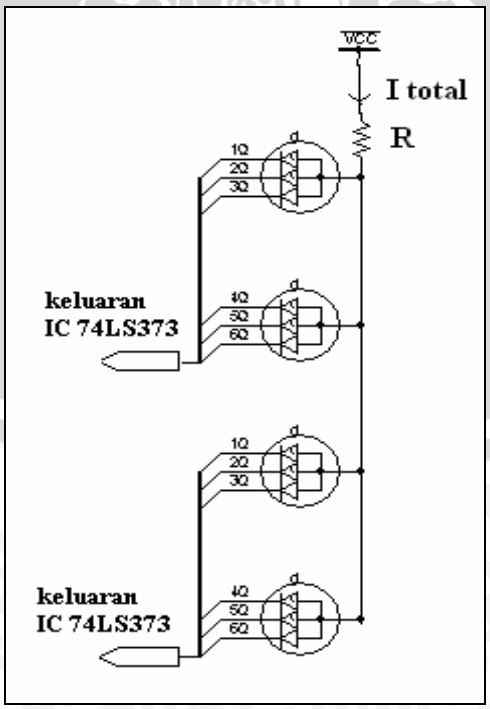
4.4.3 Rangkaian LED Peraga

LED yang digunakan dalam perencanaan ini adalah *LED RGB* 4 kaki (1 anoda 3 katoda) dengan tipe L200CWRG1KN-4A-IL. *LED* disusun menjadi 4 kolom dan 6 baris. Kolom menunjukkan kondisi rumah kabel (RK), baris menunjukkan lokasi RK. Nyala *LED* hijau *(Green)* menunjukkan RK dalam keadaan normal. Nyala *LED* merah *(Red)* menunjukkan RK dalam keadaan mayor. Nyala *LED* biru *(Blue)* menunjukkan RK dalam keadaan minor. Skema penyusunan *LED RGB* dapat dilihat dalam Gambar 4.10.



Gambar 4.10. Skema LED Peraga

Untuk menentukan resistor pembatas arus yang perlu diperhatikan adalah arus maju *(I_F)* dari *LED RGB*. Menurut *datasheet* I_F dari *LED RGB* sebesar 20 mA. Perencanaan pembatas arus dapat dilihat dalam Gambar 4.11.



Gambar 4.11. Skema Pembatas Arus.

Nilai dari R untuk Gambar 4.8 adalah :

I_F x 4 = arus maju minimal yang dibutuhkan satu kolom *LED RGB*

20 mA x 4 = 80 mA (I total, dengan kondisi *LED* nyala salah satu warna).

$$R = \frac{5volt}{80mA}$$

$$R = 62{,}5\Omega$$

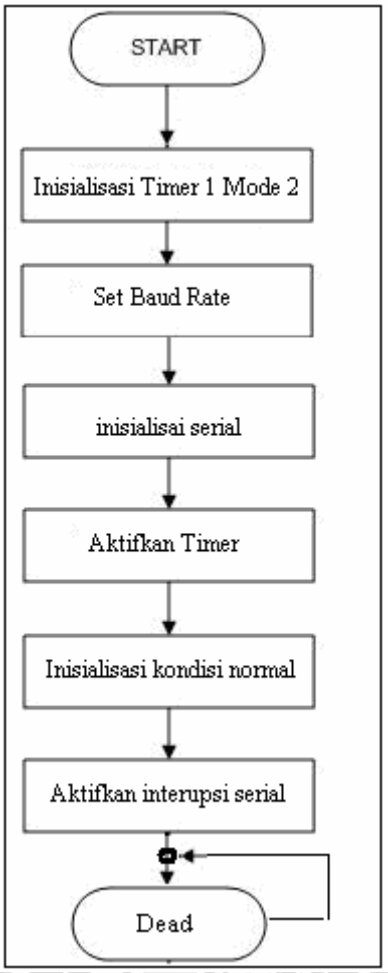
4.5. Perencanaan Perangkat Lunak (*Software*)

Ada dua macam. perangkat lunak (*Software*) pada perencanaan ini. Yang pertama perencanaan program mikrokontroler menggunakan bahasa *assembly*, yang ke dua program serial menggunakan bahasa Delphi.

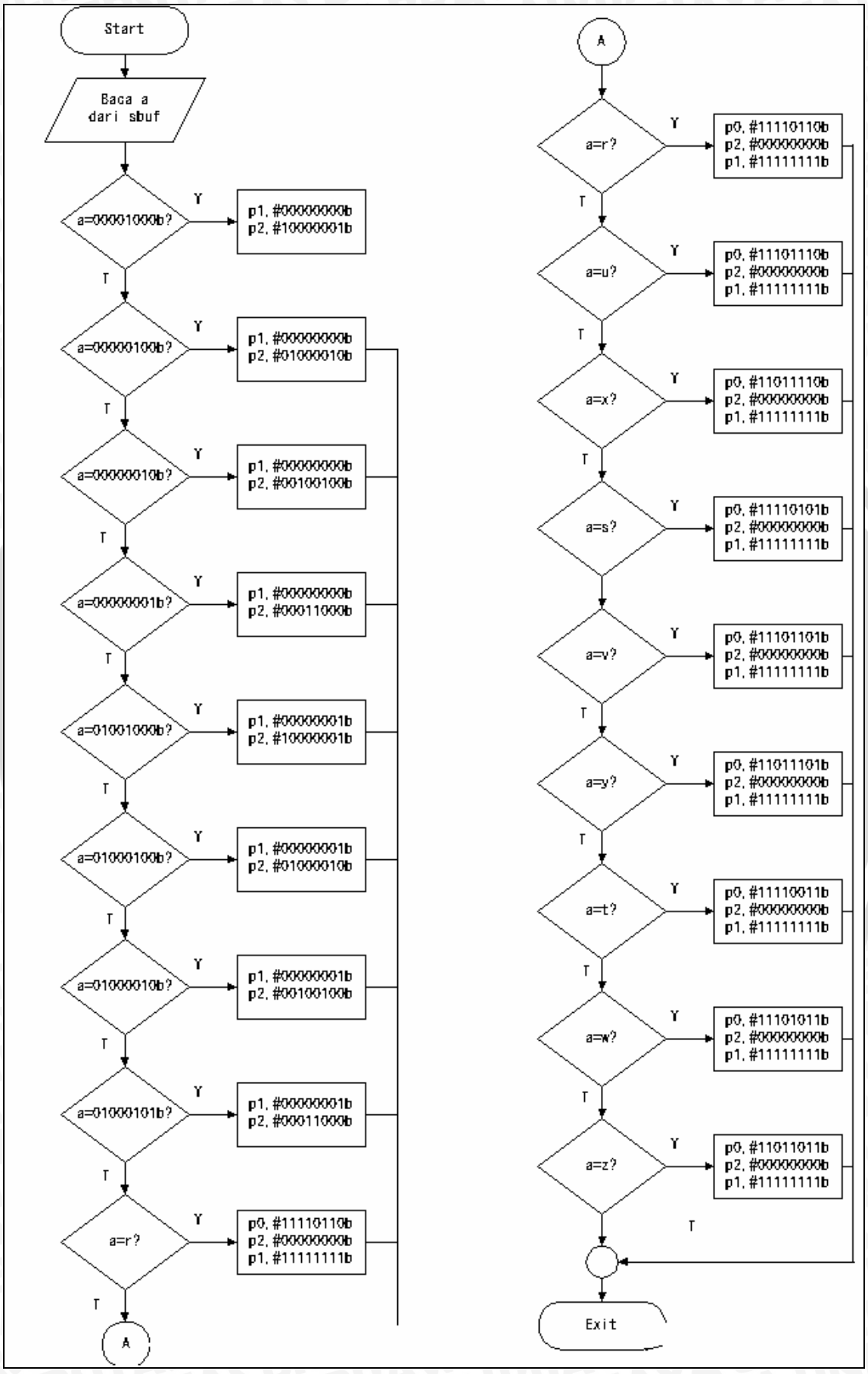
4.5.1 Program Mikrokontroler

Pada program mikrokontroler ini.semuanya menggunakan interupsi, tidak ada program utama kecuali inisialisasi *baudrate* dan interupsi. Untuk Gambar 12.a adalah diagram alir program utama. Pertama adalah inisialisasi mode timer. Di sini yang digunakan adalah timer 1 mode 2. Kemudian menset *baudrate. Baudrate* yang digunakan adalah 9600. Langkah selanjutnya menentukan mode serial. Kemudian timer diaktifkan. Setelah timer diaktifkan, kemudian menginisialisasi kombinasi port-port mikrokontroler untuk kondisi normal (semua *LED* berwarna hijau). Yang terakhir adalah inisialisasi interupsi.

Untuk diagram alir program penanganan interupsi ditunjukkan dalam Gambar 12.b. Prosesnya adalah pertama mikrokontroler menunggu datangnya kombinasi bit atau karakter dari program serial yang ada di *PC*. Jika interupsi serial dikarenakan adanya karakter yang diterima, maka dilakukan pemeriksaan terhadap karakter yang diterima tersebut. Jika karakter tersebut cocok dengan karakter didalam program mikrokontroler, maka langsung diaktifkan kombinasi port sesuai dengan kombinasi port yang ditangani karakter tersebut. Misalnya apakah diterima kombinasi bit 0001000. Jika ya maka set port 2 (10000001), port 1 (00000000). Jika tidak maka dicocokkan untuk kombinasi selanjutnya. Jika pada kombinasi port yang terakhir tidak cocok, maka keluar ke perintah inisialisasi interupsi. Program ini juga dilengkapi dengan pengembalian karakter yang diterima. Pengembalian karakter ini berfungsi untuk pengecekan rangkaian antarmuka serial.



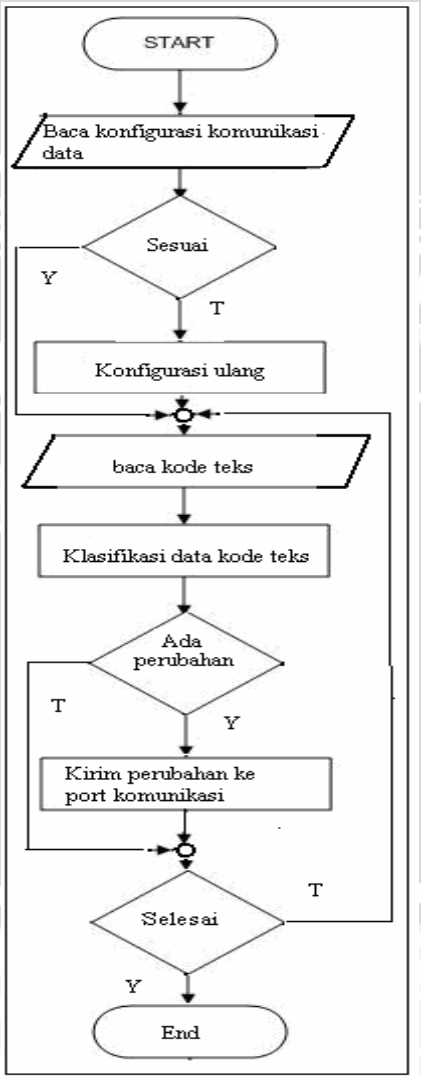
Gambar 4.12.a. Diagram Alir Program (utama) Mikrokontroler



Gambar 4.12.b. Diagram Alir Program Mikrokontroler

4.5.2 Program Antarmuka

Pertama program antarmuka membaca konfigurasi komunikasi data. Jika konfigurasi belum sesuai dengan yang diinginkan *user*, maka akan kembali atau dikonfigurasi ulang. Jika sesuai maka program antarmuka akan mulai proses data. Data yang diterima berupa kode-kode teks. Proses pengolahan data yang pertama adalah menerima kode-kode teks. Kemudian kode-kode teks tersebut dicocokkan dengan data-data yang telah diklasifikasikan menurut alamat alarm dan kondisi alarm. Jika sesuai maka kode kode teks tersebut dipecah menjadi dua yaitu menurut alamat alarm dan kondisi alarm. Kemudian dideteksi apa ada perubahan data pada masukan data selanjutnya.Jika ada maka akan dikirim ke port serial. Diagram alir program serial dapat dilihat dalam Gambar 4.13.



Gambar 4.13. Diagram Alir Program Antarmuka