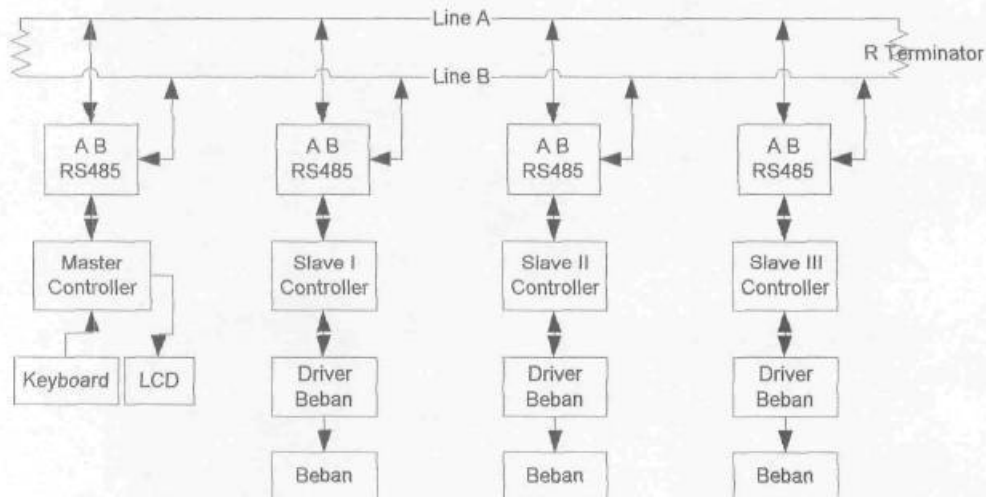


### III. PERENCANAAN SISTEM

Secara umum komunikasi *master slave* dengan *serial* RS485 dibagi menjadi dua bagian utama, yaitu pusat (*master*) kontrol dan bagian yang dikontrol (*slave*). Pada pusat (*master*) kontrol terdiri dari : *keyboard*, *display* LCD, *master controller*, dan RS485. Sedangkan pada bagian yang dikontrol terdiri dari RS485, *slave controller*, dan rangkaian *driver* beban. Berikut ini merupakan blok diagram pusat (*master*) kontrol dan bagian yang dikontrol (*slave*) beserta jalur (*line*) yang digunakan :



Gambar 3.1

Blok Diagram Sistem Kontrol Menggunakan Komunikasi *Serial* RS485

Sistem kontrol ini didasari oleh prinsip kerja dari komunikasi *serial* RS485, dimana pada rangkaian komunikasi *serial* RS485, satu buah *master* dapat mengendalikan 32 buah *slave* dan dapat mencapai jarak 4000 *feet*. Hal ini tentu

sangat berguna untuk mengatasi permasalahan jarak antara pusat kontrol dan bagian yang dikontrol dengan jumlah beban yang banyak pula.

Pada pusat (*master*) kontrol disediakan sebuah *keyboard* untuk *input* perintah, *display* LCD untuk melihat apakah perintah sudah dilaksanakan dengan benar dan untuk pengecekan status, sebuah *microcontroller* 89C51 yang merupakan otak sistem dari *master controller* dan yang memproses semua perintah sebelum dikirim melalui komunikasi *serial* RS485, serta rangkaian komunikasi *serial* RS485 yang merupakan media atau jalur yang digunakan untuk menghubungkan pusat (*master*) kontrol dan bagian-bagian yang dikontrol (*slave*). Kemudian pada *slave* terdiri dari rangkaian komunikasi *serial* RS485 yang merupakan jalur penghubung, *microcontroller* 89C2051 yang akan menjalankan perintah ke rangkaian *driver* beban yang terhubung dengan beban yang dikontrol.

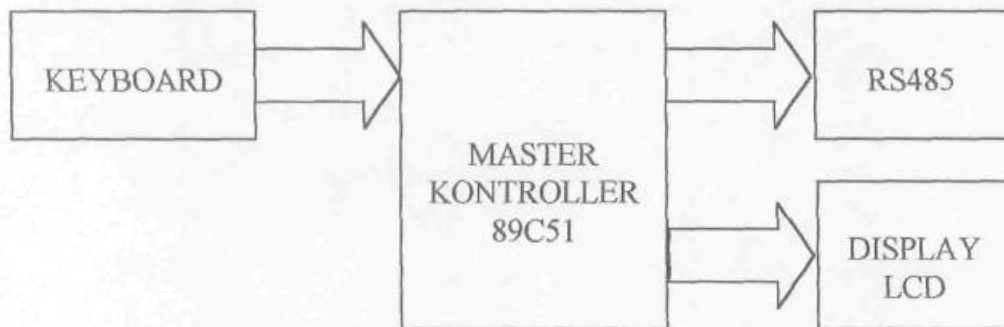
Jalur (*line*) yang digunakan dalam perencanaan ini adalah kabel telpon biasa dengan dua kabel, dan *resistor* dengan nilai 100  $\Omega$  pada kedua ujung *line* komunikasinya. Pada bagian *driver* beban *slave controller* dipasang sebuah saklar tiga kaki yang berfungsi sebagai saklar manual seperti pada saklar lampu biasa. Saklar ini ditempatkan dekat dengan bebannya.

Awalnya *master* dan *slave* diinisialisasi dengan kondisi *receive enable*, kemudian menunggu perintah dari *keyboard*. Pada tampilan awal LCD ada dua pilihan *all* atau *device*, *all* artinya sistem akan mengecek tiap *device* secara keseluruhan dari satu sampai tiga, pertama mengecek *device* satu setelah mengetahui statusnya maka menunggu perintah selanjutnya, dinyalakan (bila keadaan beban mati), dimatikan (bila keadaan beban nyala) atau mengecek status *slave* berikutnya. Sedangkan pilihan *device* menunjuk salah satu *device* yang

ingin di-cek keadaan beban dengan memasukkan nomor *slave*-nya, lalu menunggu perintah apakah dinyalakan atau dimatikan bila tidak kembali ke menu utama.

## 1. PERENCANAAN HARDWARE

### 1.1 Master (IC AT89C51)



Gambar 3.

Blok Diagram Rangkaian Master Controller

*Microcontroller* AT89C51 sebagai otak utama dari *master* kontrol, sehingga *master* kontrol ini dapat bersifat mandiri. Beberapa bagian dari perencanaan *master controller* dapat dijelaskan sebagai berikut :

1.1.1 Rangkaian Clock dan Power On Reset. Rangkaian *clock* berfungsi untuk membangkitkan sinyal *clock* untuk menjalankan *microcontroller*. Rangkaian yang dibuat adalah rangkaian yang dianjurkan oleh ATMEL, selaku pembuat *microcontroller* ini. *Microcontroller* ini mempunyai *internal clock* generator yang berfungsi sebagai sumber *clock*, namun masih diperlukan rangkaian tambahan sederhana untuk membangkitkan

*clock* yang berupa kristal atau keramik *resonator* dan *kapasitor*. Nilai kristal yang digunakan disini adalah 11,0592 MHz. Rangkaian kristal tersusun atas komponen kristal, dua buah *kapasitor* dimana nilai kapasitansi dari *kapasitor* yang digunakan sebesar 33 pF. Harga kapasitansi dari *kapasitor* diambil berdasarkan ketentuan yang diberikan oleh data *sheet*, yaitu berkisar 20 pF – 40 pF untuk kristal *oscillator* dan antara 30 pF – 50 pF digunakan untuk keramik *resonator*. Dan karena digunakan kristal *oscillator* maka diambil nilai kapasitor sebesar 33 pF.

Rangkaian *reset* yang dibuat untuk me-*reset* minimum sistem sehingga proses bisa dijalankan mulai dari awal lagi. Sinyal *reset* di-*input*-kan pada *pin* RST dari *microcontroller*, yang mana mempunyai *input* ke sebuah *Schmitt Trigger*. *Reset* yang dirancang yaitu '*power on reset*', yaitu *reset* yang terjadi pada saat sistem mendapat catu daya pertama kali. Untuk membuat rangkaian *reset*, ada beberapa hal yang perlu diperhatikan, yaitu :

- Lebar *pulsa* untuk sinyal *reset* harus lebih besar dari 24 periode osilasi atau dua *machine cycle*. Satu *machine cycle* ( $T$ ) =  $1/f_{oscillator}$
- Waktu naik (*rise time*) dari catu daya ( $V_{cc}$ ) tidak lebih dari satu milidetik
- Waktu *start up* dari *oscillator* tidak lebih dari 10 milidetik.
- Tegangan *reset* yang diperlukan pada *pin microcontroller* adalah lebih dari atau sama dengan 3,5 Volt.

Dari gambar 3.3 dapat diketahui pada saat catu daya baru saja dinyalakan, *kapasitor* bersifat seperti *short circuit* sehingga tegangan pada *pin* RST adalah sebesar tegangan  $V_{cc}$  (5 V). Sesuai dengan karakteristik

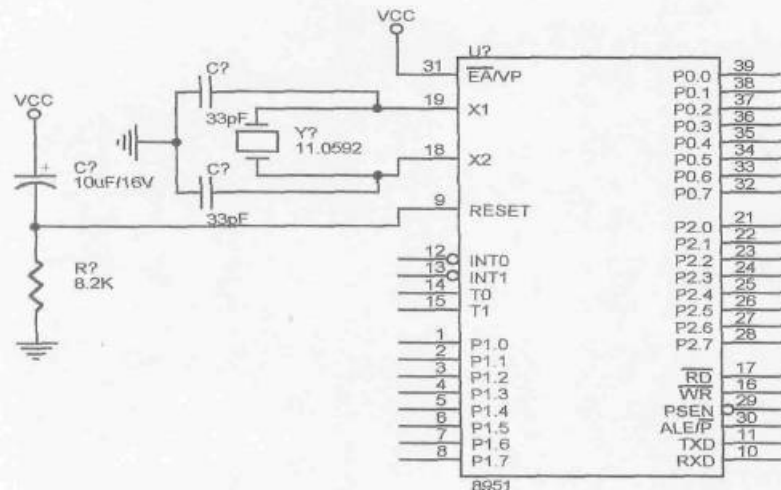
pengisian *kapasitor*, maka tegangan jatuh pada *kapasitor* akan meningkat seiring dengan waktu. Pada suatu ketika tegangan *kapasitor* ini akan cukup besar sehingga tegangan pada *pin* RST lebih kecil daripada tegangan *reset* minimum yang diijinkan. Dengan asumsi bahwa waktu *start up* dari *oscillator* adalah 10 milidetik dan frekuensi *oscillator* adalah 11,0593 MHz, maka lebar pulsa *reset* pada *pin* RST harus lebih dari harga berikut :

$$t = \left( 10^{-2} + \frac{24}{12 \cdot 10^6} \right) s = 10,002 \text{ ms}$$

Dengan ketentuan bahwa tegangan *reset* minimum adalah sebesar 3,5 Volt, maka akan diperoleh nilai *kapasitor* dan *resistor* yang harus digunakan pada rangkaian dengan rumus tegangan *pin* RST seperti berikut ini :

$$\begin{aligned} V_{RST} &= V_{cc}(1 - e^{-t/(R \cdot C)}) \\ 3,5 &= 5(1 - e^{-0,010002/(R \cdot C)}) \\ 5 e^{-0,010002/(R \cdot C)} &= 1,5 \\ \ln(e^{-0,010002/(R \cdot C)}) &= \ln 0,3 \\ 0,010002/(R \cdot C) &= 1.2039728043 \\ R \cdot C &= 8,307496 \text{ ms} \end{aligned}$$

Ada banyak nilai *kapasitor* dan *resistor* yang memenuhi persyaratan. Sesuai dengan data *sheet*, dipilih nilai *resistor* sebesar 8,2 K $\Omega$  dan nilai *kapasitor* sebesar 10  $\mu$ F. Dengan menggunakan nilai-nilai ini, didapat nilai *pulsa reset* sebesar 82 ms ( lebih besar dari lebar *pulsa reset* minimum).

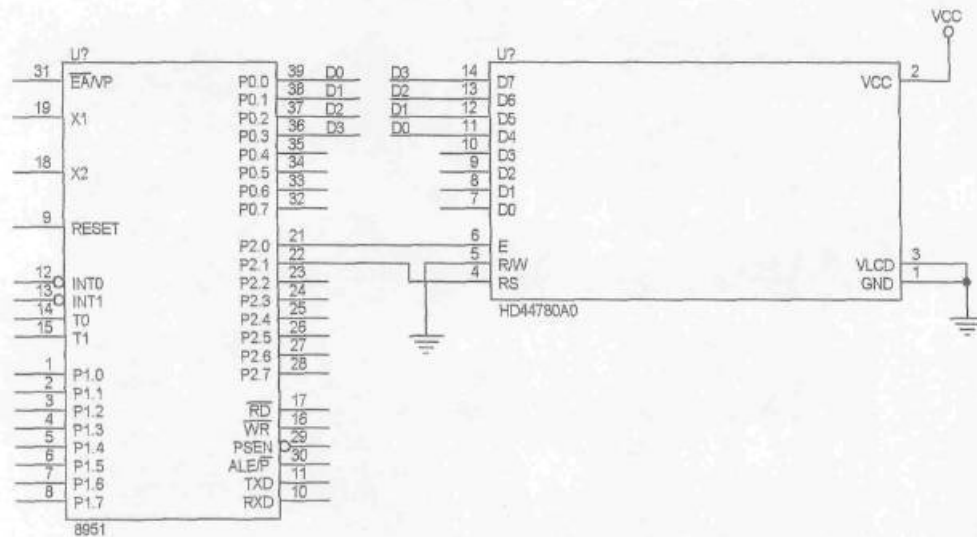


Gambar 3.3

## Rangkaian Clock AT89C51 dan Power On Reset

1.1.2 Rangkaian Microcontroller dengan LCD. Sebagai tampilan dari menu, dipilih *display* LCD Hitachi M1632 dengan *backlight*. LCD ini mempunyai tampilan dua baris dengan 16 karakter tiap baris. Untuk *backlight* dapat diakses dengan memberikan *supply* tegangan 5 Volt yang pada kaki positif diberi dioda agar tegangan di LCD hanya sekitar 4,3 Volt. LCD ini dapat diakses dengan 4 bit *interface* dengan sistem *bus*. Kelebihan dari akses dengan sistem *bus* ini terletak pada pemakaian *port* atau *pin* dari *microcontroller* yang sedikit saja.

Untuk *master microcontroller* AT89C51 rangkaian LCD-nya seperti gambar 3.4 :



Gambar 3.4

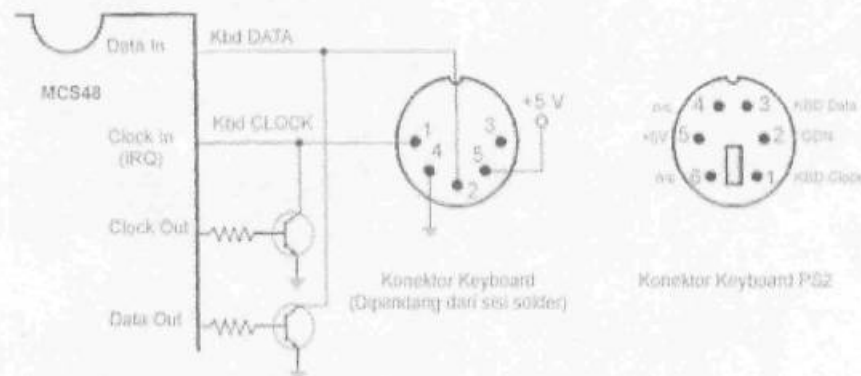
## Rangkaian AT89C51 dengan LCD

Dari gambar 3.4 dapat dilihat bahwa LCD dibuat *write only* yaitu dengan menghubungkan kaki R/W ke *ground*. Hal ini dilakukan karena *type* LCD ini mempunyai sistem *bus* model Motorola, sedangkan sistem *bus* MCS51 memakai sistem Intel. Alasan lainnya karena fungsi untuk membaca posisi LCD tidak digunakan pada sistem ini.

Kaki E *clock* dihubungkan dengan P2.0, *pin* ini berfungsi sebagai *clock* yang aktif setiap *cycle* siklus dijalankan. Sedangkan kaki RS berfungsi sebagai *register select* dimana bila RS berlogika '0' maka berarti penulisan instruksi *register*, dan bila berlogika '1' berarti penulisan data *register*.

Sebelum digunakan LCD perlu diinisialisasi terlebih dahulu dalam hal ini pentabelannya dilakukan oleh *software*-nya.

1.1.3 Rangkaian Mikrokontroller dengan Keyboard PC. Sebagai *input* dipakai *keyboard* PC (Personal Computer), dengan tujuan agar I/O yang digunakan lebih sedikit dibandingkan jika menggunakan *keypad*. Gambar 3.5 adalah gambar rangkaiannya:



Gambar 3.5<sup>1</sup>

#### Rangkaian Konektor Keyboard PC

Kbd Data dihubungkan dengan P1.4 dari *microcontroller* dan Kbd Clock ke *pin* INT0. Kbd Data berisi *scan code* dari tombol yang ditekan atau dilepas. *Scan code* yang dikirimkan oleh *keyboard*, bukan karakter ASCII secara langsung, untuk menjadi karakter ASCII maka harus diterjemahkan terlebih dahulu melalui suatu tabel.

1.1.4 Rangkaian Mikrokontroller dengan RS485. Sebagai media *transfer* data *serial* antara *master* dan *slave* dipakai sistem RS485, dengan rangkaian seperti gambar 3.6 :

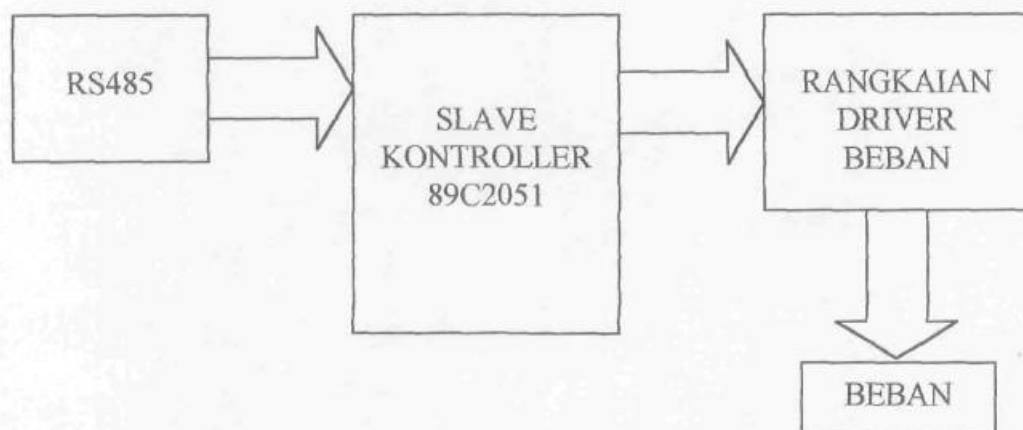
<sup>1</sup> <http://alds.stts.edu>



*Resistor* 100  $\Omega$  berfungsi sebagai *terminator* yang mencegah terjadinya keadaan logika tidak menentu pada saluran. Hal ini dilakukan terutama untuk transfer data kecepatan tinggi. Selain itu resistor 100  $\Omega$  juga berfungsi mencegah terjadinya gelombang pantul. Menurut data *sheet resistor* 100  $\Omega$  ini minimal berjumlah dua buah yang dipasang pada kedua ujung saluran komunikasi.

1.1.5 Rangkaian Gabungan. Rangkaian gabungan dari ketiga bagian yang telah dibahas menjadi sebuah *master controller* untuk sistem ini. Gambar rangkaian gabungannya dapat dilihat pada lampiran 1.

#### 1.2 Slave (AT89C2051)



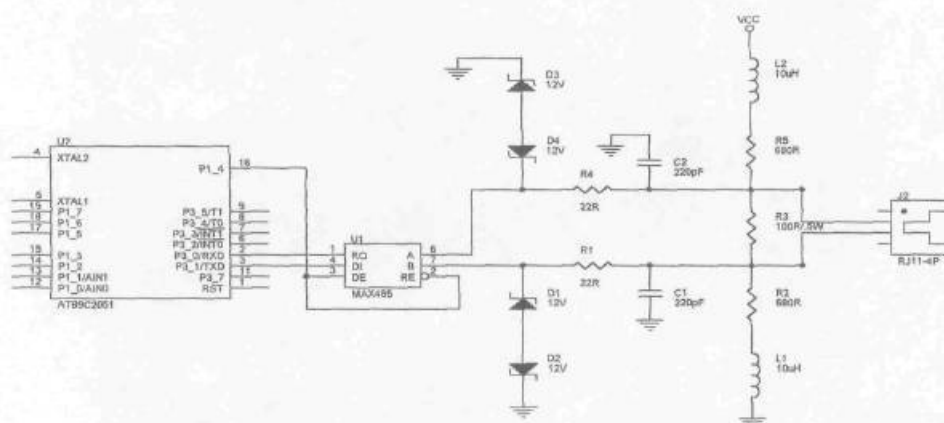
Gambar 3.7

Blok Diagram Slave Controller

Sebagai otak dari *slave controller* dipakai *microcontroller* AT89C2051 dengan pertimbangan bahwa dalam perencanaannya *microcontroller* ini hanya mengontrol satu buah *driver* beban. Mengenai rangkaian *clock* dan

rangkaian *power on reset* digunakan rangkaian yang sama yang telah dijelaskan pada rangkaian *master controller*. Ditambah dengan rangkaian sensor untuk mengetahui status / keadaan dari beban yang diinginkan (mati, nyala atau error). Saklar tiga kaki digunakan sebagai saklar manual beban yang berfungsi layaknya saklar *on / off* biasa.

1.2.1 Rangkaian Microcontroller dengan RS485. Untuk rangkaian *slave* digunakan *microcontroller* AT89C2051, hubungannya dengan RS485 sama dengan rangkaian *microcontroller* dengan RS485 pada rangkaian *master*. Perencanaan *slave* seperti gambar 3.8 dibawah ini :



Gambar 3.8<sup>3</sup>

Rangkaian AT89C2051 dengan RS485

1.2.2 Rangkaian Mikrokontroller dengan Driver Beban. Sebagai rangkaian *driver* beban dipilih triac dengan rangkaian *zero cross circuit* yang terdapat dalam MOC 3041. MOC 3041 adalah jenis *optocoupler zero cross circuit* merupakan proteksi yang aman bagi bagian *microcontroller*

<sup>3</sup> [www.centrin.net.id/delta.com](http://www.centrin.net.id/delta.com)

dari arus jala-jala listrik AC 220 Volt. Untuk men-*trigger* LED sesuai dengan *data-sheet* arus maksimalnya 15 mA, maka nilai *resistor* yang dipakai 330  $\Omega$  dengan perhitungan :

$$R = (V_{cc} - 1 \text{ Volt}) / I_{\max}$$

$$R = (5 - 1) \text{ Volt} / 15 \text{ mA}$$

$$R = 266,67 \Omega$$

$$\text{Diambil } R = 330 \Omega.$$

Sebagai sensor arus dipakai H11AA1 dengan 4 buah *diode* IN4001 dan *resistor* 22  $\Omega$ . 4 buah *diode* berfungsi sebagai menyearah sekaligus sebagai referensi tegangan untuk *input* sekitar 1,5 Volt, sedangkan *resistor* 22  $\Omega$  berfungsi untuk membatasi arus sehingga tidak melebihi arus maksimum yang ditentukan sesuai data *sheet* dari H11AA1. Nilai *resistor* didapat dari rumus :

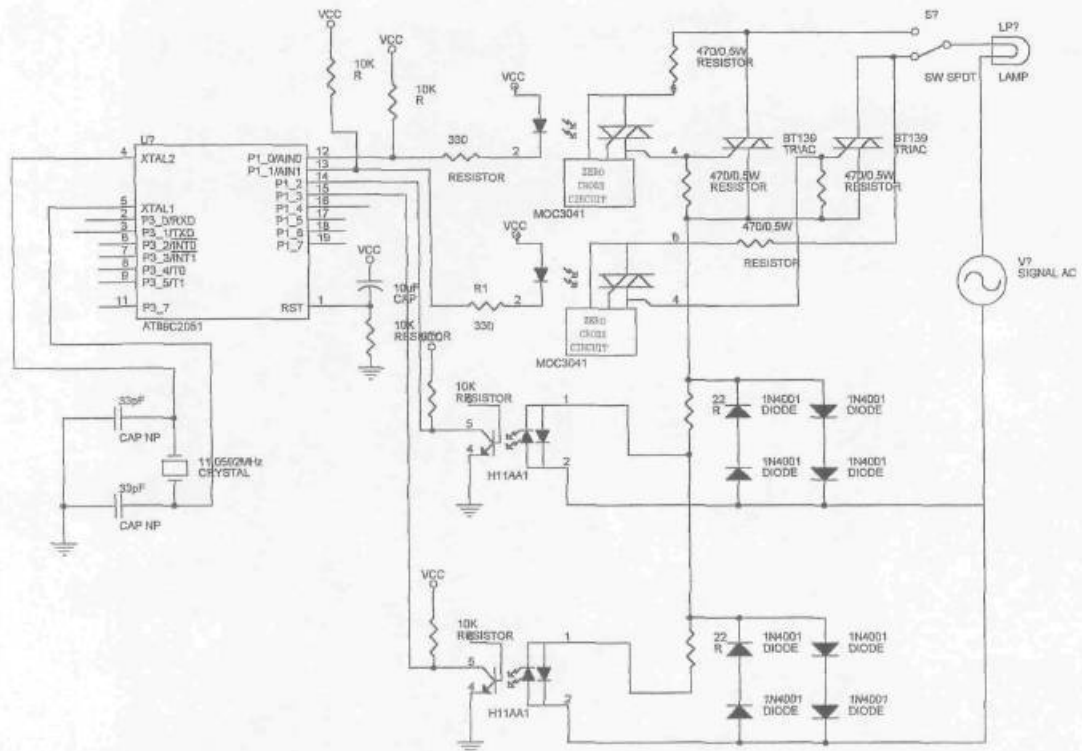
$$R = (1,4 - 1) \text{ Volt} / I_{\max}$$

$$R = 0,4 \text{ Volt} / 60 \text{ mA}$$

$$R = 6,67 \Omega$$

$$\text{diambil } R \approx 22 \Omega$$

Nilai resistor yang diambil 22  $\Omega$  untuk menghindari arus yang berlebihan pada detektor beban. Apabila nilai resistor terlalu besar maka untuk beban yang kecil kemungkinan tidak terdeteksi dengan baik. Resistor dengan nilai 10  $\Omega$  sebenarnya memenuhi criteria diatas tetapi nilai tersebut dinilai terlalu kritis.



Gambar 3.9

### Rangkaian AT89C2051 dengan Driver Beban

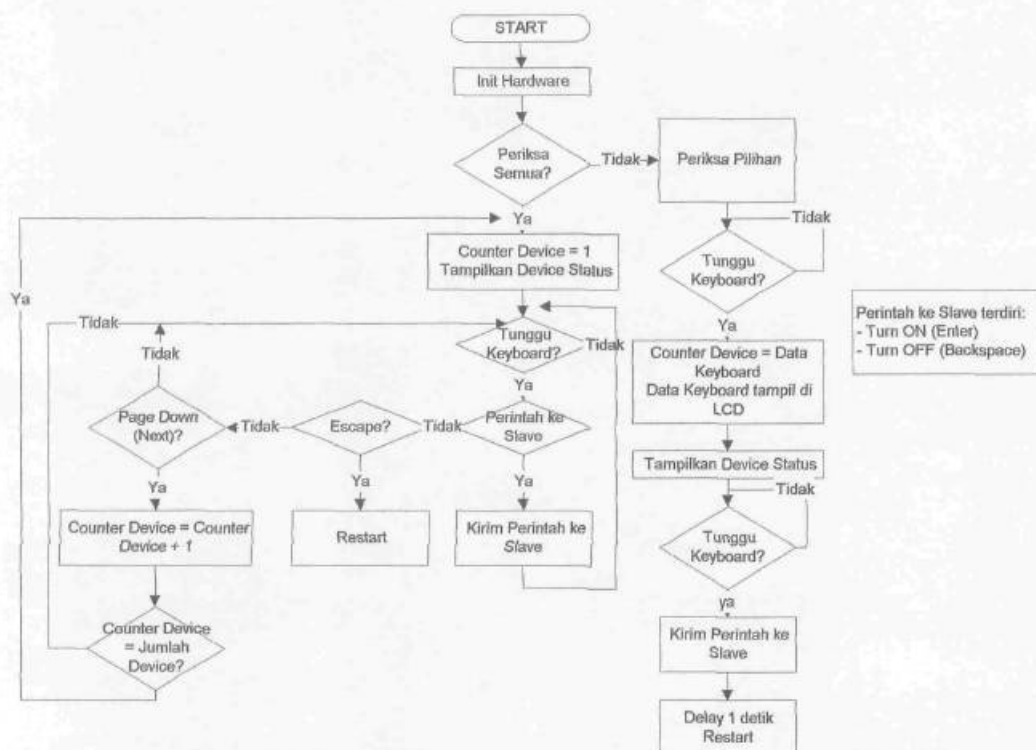
Saklar SPDT adalah saklar tiga kaki yang berfungsi sebagai saklar manual. Fungsinya untuk menyalakan langsung beban yang diinginkan tanpa perlu menyalakan / mematikan menggunakan *master controller*. Saklar ini diletakkan berdekatan dengan beban agar mudah dalam pengoperasiannya.

1.2.3 Rangkaian Gabungan. Rangkaian gabungan untuk *slave* dapat dilihat pada lampiran 2.

## 2. PERENCANAAN SOFTWARE

### 2.1 Master

Perencanaan *software* untuk master berfungsi mengirimkan perintah dari *keyboard* ke *slave*, menampilkan status di *display* LCD, dan menunggu balasan / jawaban *slave*. Dalam perencanaan *software* inisialisasi *hardware* termasuk didalamnya inisialisasi untuk *keyboard* dan LCD, serta pertama kali RS485 dibuat *receive enable*. Berikut disertakan bagan *flowchart* yang mendukung perencanaan dan pembuatan *software master control*.



Gambar 3.10

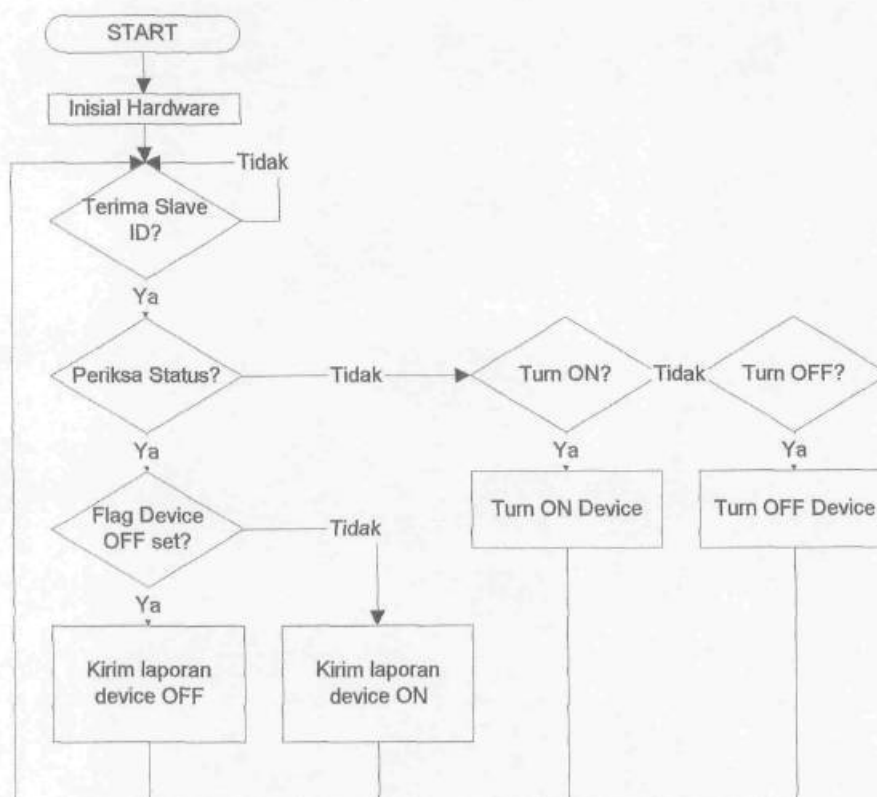
Algoritma Program Master

Tabel 3.1  
Perintah Master Ke Slave

| Kode             | Arti               |
|------------------|--------------------|
| 41 H             | Check Status       |
| 5A H (Enter)     | Turn On            |
| 66 H (Backspace) | Turn OFF           |
| 71 H (Page Down) | Device Selanjutnya |

## 2.2 Slave

Perencanaan *software* untuk *slave* berfungsi menerima perintah dari *master*, melaksanakan dan memberikan laporan dari pelaksanaan perintah tersebut kepada *master*. Pada saat inisialisasi pertama kali RS485 dibuat *receive enable* dan kedua triac di-*clear* / tidak aktif. Berikut disertakan bagan *flowchart* perencanaan *software slave*.



Gambar 3.11

Algoritma Program Slave

Tabel 3.2

Laporan Slave Ke Master

| Kode | Arti           |
|------|----------------|
| 41 H | Device On      |
| 42 H | Device Off     |
| 43 H | Device Failure |