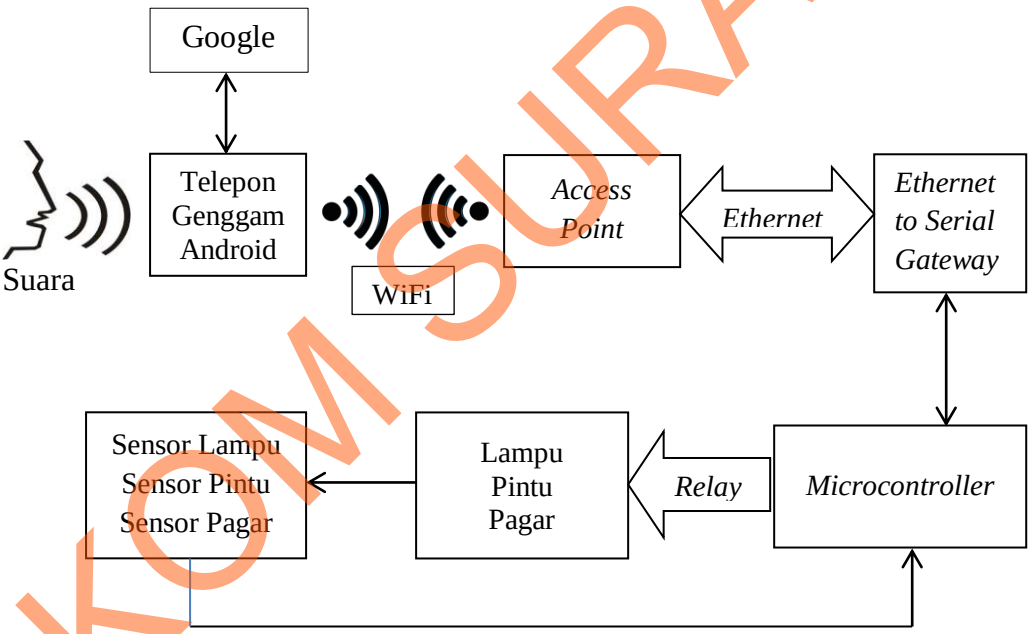
BAB III

METODE PENELITIAN

3.1 Model Penelitian

Pengerjaan Tugas Akhir ini dapat terlihat jelas dari blok diagram yang tampak pada Gambar 3.1. Blok diagram tersebut menggambarkan proses dari input suara hingga perintah ke aktuator. Terdapat beberapa komponen penting pada blok diagram tersebut antara lain adalah telepon genggam berbasis android, *wireless access point*, *microcontroller*, dan rangkaian *relay driver*.



Gambar 3.1 Blok Diagram

Pada Gambar 3.1 dapat dijelaskan sebagai berikut:

a. User melakukan input melalui Google *voice*, kemudian input dari *user* diproses oleh *Google* dengan cara berkomunikasi dengan *server*. Setelah berkomunikasi dengan *server*, Google *voice* menerima data dari *server* berupa beberapa *string* perkiraan dari *input* yang diberikan oleh *user*. Beberapa *string* yang diterima akan di saring lagi untuk mendapatkan hasil input yang dberikan oleh *user*

dengan cara mencocokkan data tersebut dengan *database* perintah yang terdapat pada program Android.

b. Setelah mendapat perintah yang valid makan program Android akan mengkonversi perintah tersebut menjadi *character* untuk dikirim ke *microcontroller* melalui jalur WiFi. *Acces point* menerima sinyal perintah dari telepon genggam yang berupa *character* dan meneruskannya ke *usb to serial gateway* melalui jalur *ethernet*.

c. USB *to serial gateway* menerima sinyal perintah dari *access point* melalui jalur *ethernet* dan mengubah sinyal perintah tersebut menjadi sinyal serial agar dapat di terima oleh *microcontroller*.

d. *Microcontroller* menerima sinyal dari telepon genggam dan mengirimkan perintah dari telepon genggam untuk menggerakan aktuator yang terhubung dengan peralatan yang ada di rumah.

e. Aktuator digunakan untuk mengatur peralatan yang ada. Setelah menerima sinyal perintah dari *microcontroller*, maka aktuator akan mengaktifkan atau menonaktifkan *relay*. Aktuator itu sendiri terdiri dari rangkaian *relay-relay*. Setiap *relay* terhubung dengan satu macam alat misalnya lampu atau alat pengunci pintu.

f. Rangkaian sensor menerima inputan dari peralatan yang ada dirumah dan memberitahukan keadaan dari setiap alat yang dikendalikan kepada *microcontroller* agar informasi tersebut dapat diteruskan ke telepon genggam sehinga *user* dapat mengetahui keadaan dari alat.

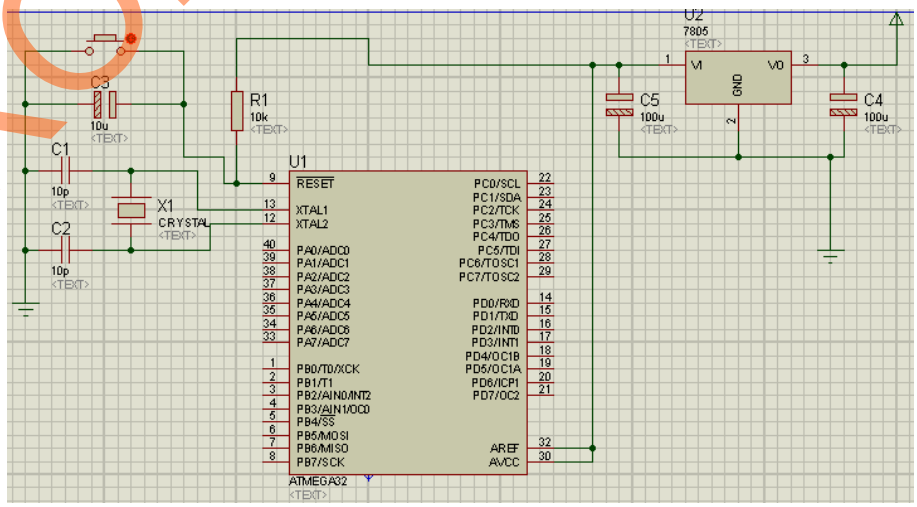
3.2 Perancangan Perangkat Keras

Perangkat keras yang diperlukan guna mendukung kelancaran sistem antara lain terdiri dari rangkaian *relay driver*, rangkaian sensor, rangkaian max232, modul WIZ110SR serta penggunaan rangkaian *minimum system*.

3.2.1 *Rangkaian Microcontroller*

Pada penelitian ini menggunakan *microcontroller* sebagai alat pengendali sistem. *Microcontroller* yang digunakan adalah AVR ATmega32. Untuk menjalankan *microcontroller* dibutuhkan rangkaian *minimum system*. Rangkaian *minimum system* adalah rangkaian dasar yang dibutuhkan *microcontroller* agar dapat berfungsi.

Untuk menjalankan *microcontroller* diperlukan rangkaian *minimum system*. Rangkaian *minimum system microcontroller* terdiri dari rangkaian *reset*, rangkaian *oscillator*, dan rangkaian *regulator*. Tegangan inputnya 12v akan diregulasi menjadi 5v oleh *regulator*. Untuk skematik rangkaian *minimum system* dapat dilihat pada Gambar 3.2:



Gambar 3.2 Rangkaian *Minimum System*

Pada Gambar 3.2 dapat dijelaskan bahwa *crystal* yang digunakan adalah 11.0592MHz untuk meminimalkan *error* yang terjadi dan sumber tegangan yang digunakan untuk *microcontroller* adalah 5v.

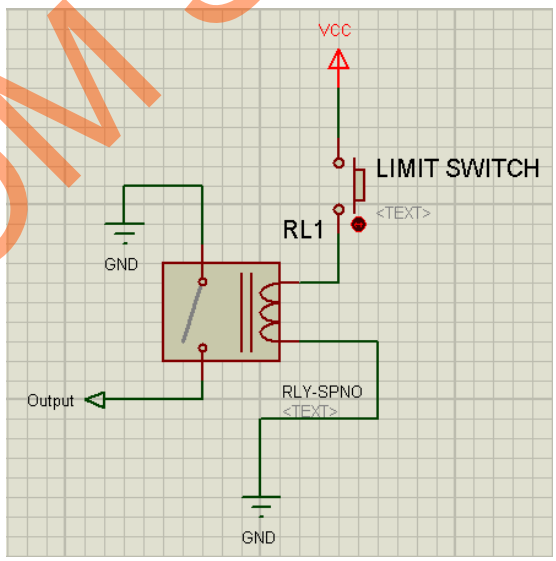
3.2.2 Rangkaian Sensor Input

Pada rangakaian sensor input yang menggunakan *port* B sebagai jalur input dan terdapat 7 buah inputan dari sensor yang mendeteksi status dari tiap perangkat yang dikendalikan. Berikut potongan program *microcontroller* untuk input.

```
PORTB=0xFF;
DDRB=0x00;
```

Pada PORTB disi 0xFF = 255 karena port B diberi nilai awal 1 (*pullup internal*). Sedangkan untuk DDRB diisi 0x00 = 0 karena semua pin pada port B digunakan untuk input.

Untuk skematik dari rangkaian sensor input dapat dilihat pada Gambar 3.3.



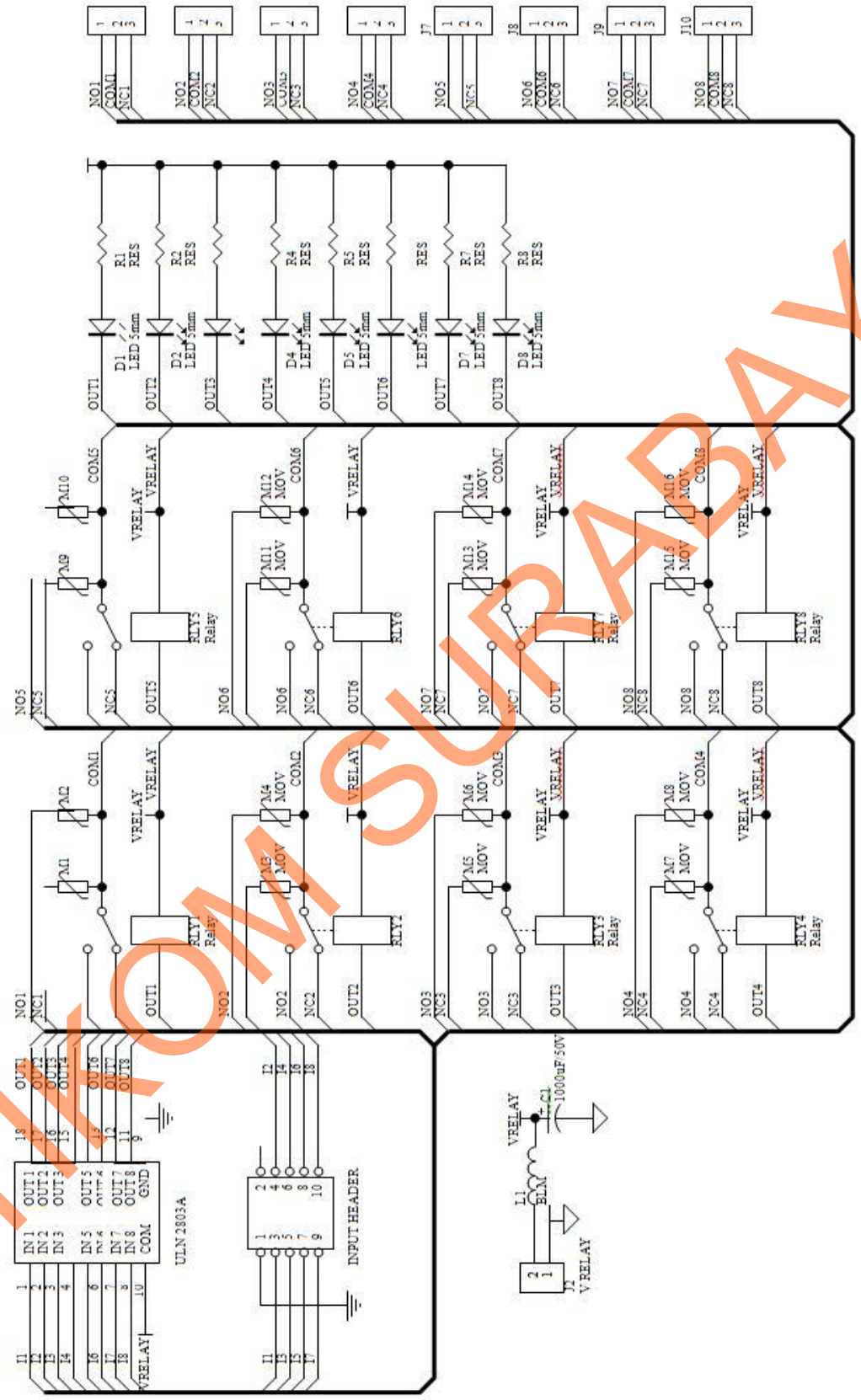
Gambar 3.3 Skematik Sensor Input

Ketika *limit switch* aktif maka *coil* pada *relay* mendapat tegangan sehingga *relay* menjadi aktif dan menghubungkan antara *ground* dengan pin input sehingga

microcontroller mendapat inputan. Pin input dihubungkan dengan *ground* karena *microcontroller* diatur untuk menerima inputan *low* (aktif *low*).

3.2.3 Rangkaian *Relay Driver*

Pada rangkaian *relay driver* menggunakan IC ULN2803A sebagai penguat arus untuk menggerakkan *relay* dan IC ini dilengkapi dengan *suppression diode* untuk mencegah arus balik. Untuk skematik dari rangkaian *relay driver* ini dapat dilihat pada Gambar 3.4.



Gambar 3.4 Skematik *Relay Driver*

Relay driver mendapat inputan dari *microcontroller* melalui IC ULN2803A setelah itu mengalirkan tegangan ke *coil relay* untuk mengaktifkan *switch*. Pada rangkaian ini terdapat LED untuk indikator dari output.

Untuk pengaturan output dari *microcontroller* berikut potongan programnya.

```
PORTA=0x00;
DDRA=0xFF;
```

Pada PORTA disi 0x00 = 0 karena port A diberi nilai awal 0. Sedangkan untuk DDRA diisi 0xFF = 0 karena semua pin pada port B digunakan untuk *output*.

3.2.4 Perancangan Interface I/O

Rangkaian I/O dari *microcontroller* mempunyai *control* direksi yang tiap bitnya dapat dikonfigurasikan secara individual, maka dalam perancangan I/O yang digunakan ada yang berupa operasi port ada pula yang dikonfigurasi tiap bit I/O. Berikut ini akan diberikan konfigurasi dari I/O *microcontroller* tiap bit yang ada pada masing-masing port yang terdapat pada *microcontroller*.

Port A digunakan untuk memberikan inputan pada *relay driver*. Untuk konfigurasi port A dapat dilihat pada Tabel 3.1.

Tabel 3.1 Konfigurasi Port A

Port	Fungsi
Port A.0	Lampu 1
Port A.1	Lampu 2
Port A.2	Lampu 3
Port A.3	Kunci Pintu 1
Port A.4	Kunci Pintu 2
Port A.5	Motor *Enable*
Port A.6	Motor Maju
Port A.7	Motor Mundur

Port B digunakan untuk menerima inputan dari sensor. Untuk konfigurasi port B dapat dilihat pada Tabel 3.2

Tabel 3.2 Konfigurasi Port B

Port	Fungsi
Port B.0	Sensor Lampu 1
Port B.1	Sensor Lampu 2
Port B.2	Sensor Lampu 3
Port B.3	Sensor Pintu 1
Port B.4	Sensor Pintu 2
Port B.5	Sensor Pagarmax
Port B.6	Sensor Pagarmin

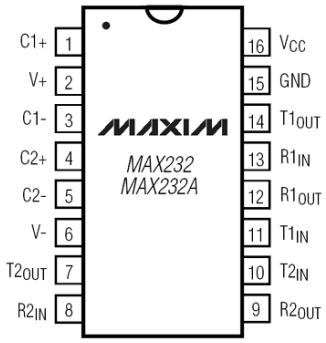
Port D digunakan untuk komunikasi dengan modul WIZ110SR melalu jalur serial. Untuk konfigurasinya dapat dilihat pada Tabel 3.3.

Tabel 3.3 Konfigurasi Port D

Port	Fungsi
Port D.0	Receive serial (RX)
Port D.1	Transmit serial (TX)

3.2.5 IC max232

Gambar 3.5 merupakan konfigurasi dari IC max232.



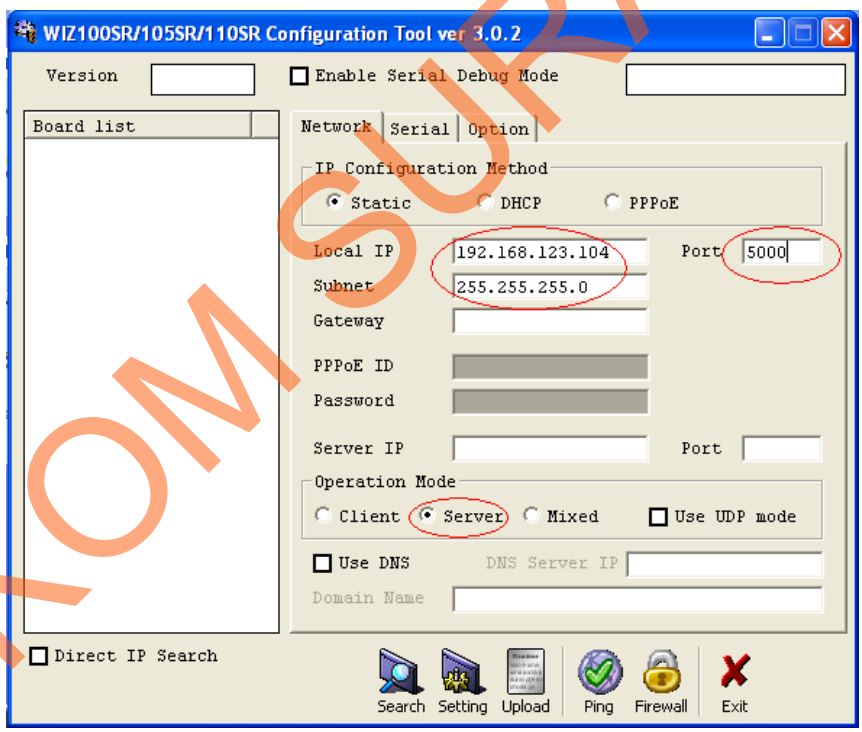
Gambar 3.5 Konfigurasi IC max232

IC max232 ini digunakan untuk mengkonversi level tegangan dari tingkat rs232 ke tingkat TTL karena *microcontroller* memiliki tingkat tegangan TTL (5v). Berbeda dengan tingkat rs232 yang tegangannya bisa mencapai lebih dari 5 volt.

Apabila tidak menggunakan IC ini akan menyebabkan rusaknya *microcontroller* terutama pin rx dan tx sehingga *microcontroller* tidak dapat melakukan komunikasi serial.

3.2.6 Konfigurasi WIZ110SR

Microcontroller dapat berkomunikasi melalui jaringan berbasis *internet protocol* menggunakan modul WIZ110SR, untuk itu diperlukan beberapa pengaturan pada modul WIZ110SR. Pengaturan tersebut dapat dilakukan melalui WIZ110SR *Configuration Tool*. Tampilan jendela pengaturan modul WIZ110SR dapat dilihat pada gambar 3.6



Gambar 3.6 Layar Editor Konfigurasi WIZ110R

Pada gambar 3.6 dijelaskan bahwa pertama kali pada konfigurasi WIZ110SR untuk menjadi *mode server* yaitu kita pilih tombol search supaya kotak *box* pada konfigurasi keluar IP *default* dari modul beserta informasi lainya seperti versi *firmware* dari modul dan *mac address*.

Langkah-langkah keseluruan dari pengaturan modul *ethernet* WIZ110SR adalah sebagai berikut :

1. Modul WIZ110SR dikoneksikan dengan komputer yang akan digunakan untuk proses konfigurasi melalui *network switch*.
2. Konfigurasi modul dilakukan dengan menggunakan WIZ110SR *configuration tool* seperti pada Gambar 3.6.
3. Untuk memulai proses konfigurasi tekan tombol *search* pada *tool* untuk menampilkan daftar modul yang terkoneksi ke jaringan. Daftar modul akan tampil di sebelah kiri (*Board List*) pada gambar 3.6.
4. Pilih salah satu *board* yang akan dikonfigurasi. Ketika dipilih, pada bagian kanan akan muncul konfigurasi yang telah disimpan ke dalam modul sebelumnya.
5. Setelah *board* dipilih, masukkan IP *address* dan *subnetmask* pada kolom yang tersedia. Untuk IP diisi dengan 192.168.123.104 dengan *subnet* 255.255.255.0 .
6. Pada bagian *port* diisi sesuai dengan *port* komunikasi yang digunakan. Untuk *port* yang digunakan adalah *port* 5000.
7. Pada bagian *operation mode* pilih *server*, karena WIZ110SR akan difungsikan sebagai *server*.
8. Pada tool ini terdapat 2 tab yang wajib dikonfigurasi. Masing- masing tab tersebut memiliki fungsi sebagai berikut :

   a. Network

   Mengkonfigurasi modul WIZ110SR terkait dengan bagaimana modul tersebut dapat berkomunikasi melalui jaringan, seperti *IP Address, Subnet*

Mask, *Gateway*, dan *Port*. Pada tab ini, beberapa hal yang dapat dikonfigurasi adalah sebagai berikut:

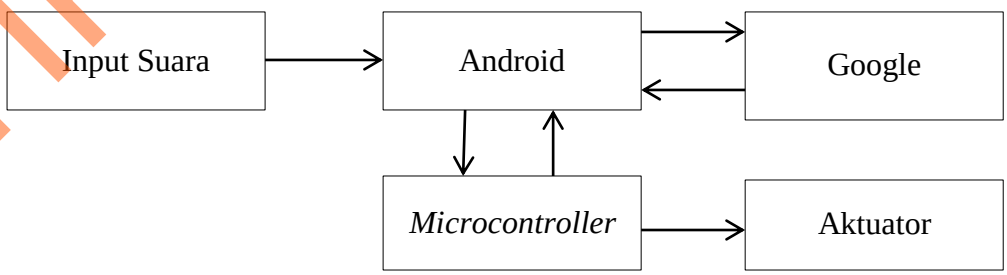
1. *IP Configuration Method*, digunakan untuk menentukan pengaturan alamat IP. Pengaturan alamat IP yang digunakan yaitu menggunakan *static* IP .
2. *Operation Mode*, digunakan untuk menentukan mode operasi dari modul WIZ110SR. Mode yang digunakan adalah mode mixed.

b. Serial

Mengkonfigurasi modul terkait dengan bagaimana modul dapat berkomunikasi dengan *microcontroller melalui Universal Asyncronus Receiver Transmitter* (UART) seperti *Baud Rate* (*Speed*), Jumlah bit data setiap paket (DataBit), *Parity*, *Stop* Bit, dan *Flow Control*. Setelah semua terkonfigurasi sesuai (Network & Serial) tekan tombol setting untuk mengirimkan konfigurasi ke modul WIZ110SR.

3.3 Perancangan Perangkat Lunak

Perangkat lunak yang dibuat untuk sistem ini terdiri dari 2 bagian besar yaitu program untuk menerima dan mengirim data. Untuk mempermudah penjelasan pada perancangan perangkat lunak dapat dilihat pada Gambar 3.7.



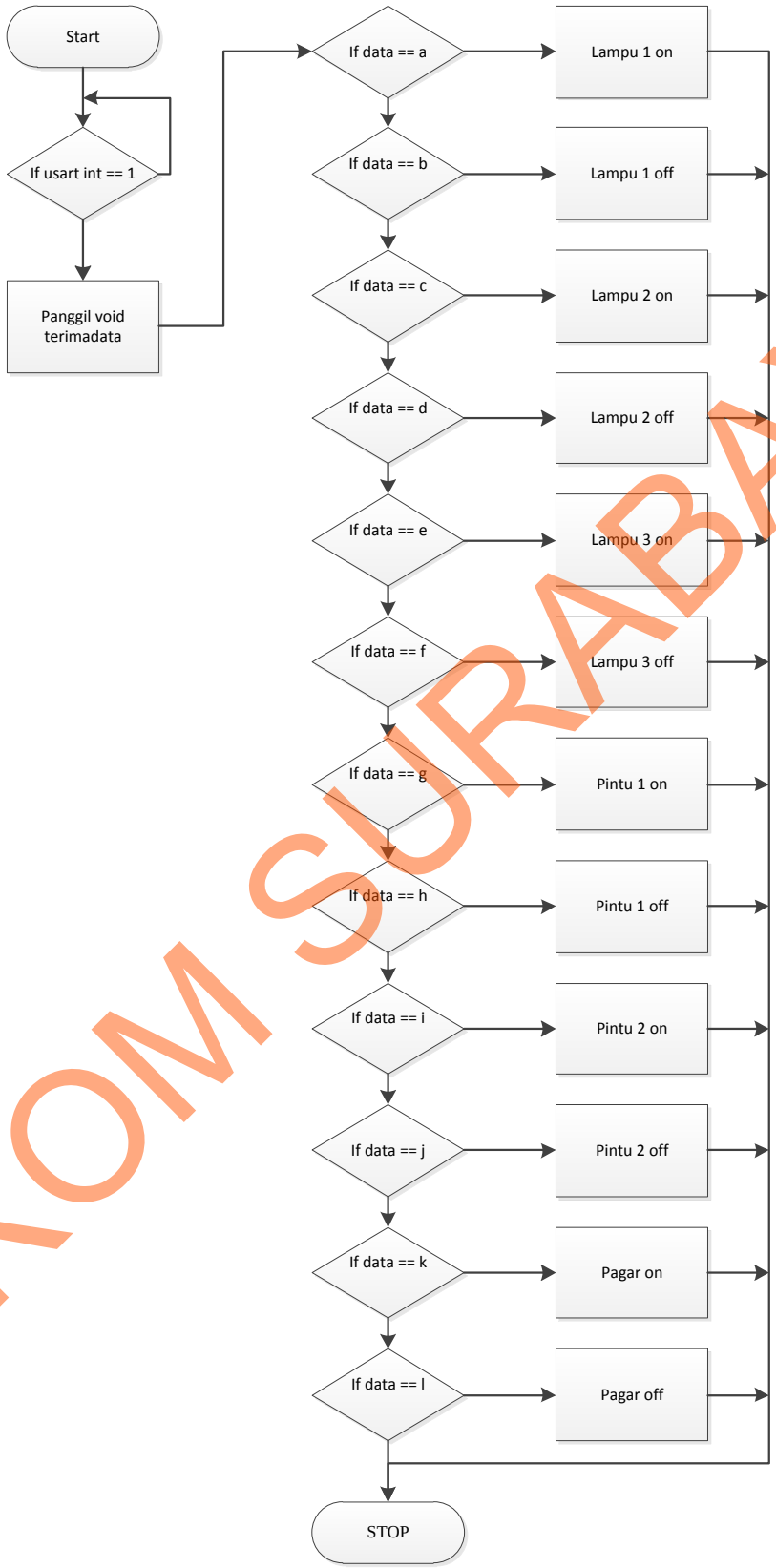
Gambar 3.7 Blok Diagram Umum

User melakukan input suara pada telepon genggam berbasis Android. Setelah itu Android akan melakukan komunikasi dengan *server* Google untuk

melakukan pengecekan input suara. Berikutnya Android mengolah data yang didapat dari *server* Google kemudian mengirimkan perintah ke *microcontroller*. Perintah dari Android diolah kembali oleh *microcontroller* yang kemudian mengaktifkan aktuator untuk menyalakan perangkat elektronik.

3.3.1 Menerima Data dari Android

Dalam melakukan penerimaan data pada *microcontroller* dibuat perancangan *flowchart* sebagai berikut :



Gambar 3.8 *Flowchart* Terima Data *Microcontroller*

Ketika ada data yang masuk pada *buffer* serial maka *interrupt* serial akan aktif dan menjalankan void terima data yang membaca perintah dari *user* dan mengaktifkan *relay driver* yang terhubung dengan perangkat elektronik.

Daftar perintah yang dikirim oleh telepon genggam berbasis Android dapat dilihat pada Tabel 3.4.

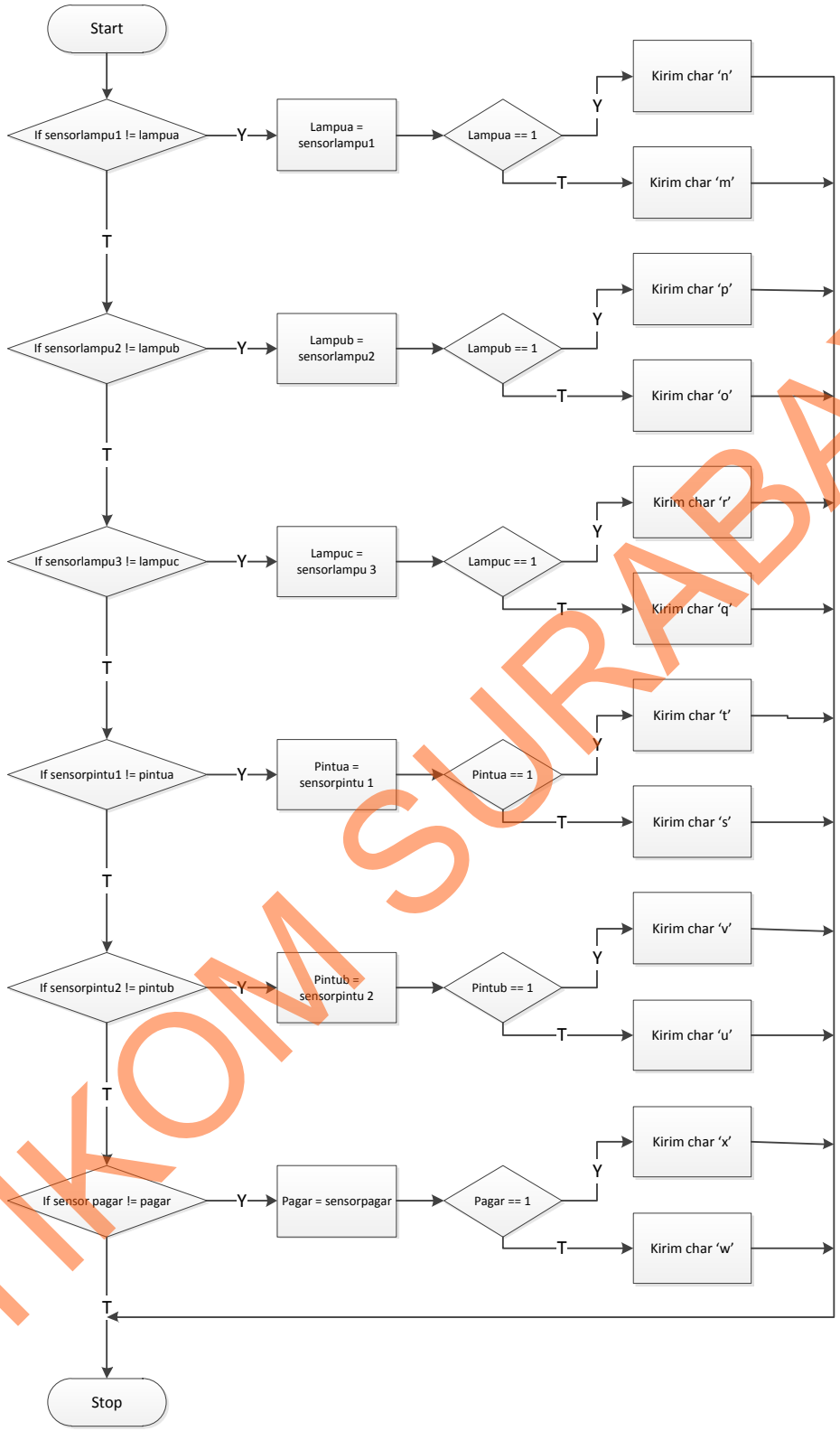
Tabel 3.4 Daftar Perintah Android

Perintah	Aksi
a	Mengaktifkan lampu 1
b	Mematikan lampu 1
c	Mengaktifkan lampu 2
d	Mematikan lampu 2
e	Mengaktifkan lampu 3
f	Mematikan lampu 3
g	Membuka kunci pintu 1
h	Menutup kunci pintu 1
i	Membuka kunci pintu 2
j	Menutup kunci pintu 2
k	Membuka pagar
l	Menutup pagar

```
interrupt [USART_RXC] void usart_rx_isr(void)
{
   terimadata();
}
void terimadata()
{
    data=getchar();delay_ms(1000);
    if(data=='a'){lampu1=1;}
    if(data=='b'){lampu1=0;}
    if(data=='c'){lampu2=1;}
    if(data=='d'){lampu2=0;}
    if(data=='e'){lampu3=1;}
    if(data=='f'){lampu3=0;}
    if(data=='g'){pintu1=1;}
    if(data=='h'){pintu1=0;}
    if(data=='i'){pintu2=1;}
    if(data=='j'){pintu2=0;}
    if(data=='k'){pagarbuka();}
    if(data=='l'){pagartutup();}
}
```

3.3.2 Mengirim Data ke Android

Dalam melakukan pengiriman data ke telepon genggam berbasis Android pada *microcontroller* dibuat perancangan *flowchart* sebagai berikut.



Gambar 3.9 *Flowchart* Kirim Data *Microcontroller*

Pada proses pengiriman data, *microcontroller* mendeteksi adanya perubahan kondisi dari sensor. Apabila terjadi perubahan maka *microcontroller* akan mengirimkan data ke telepon genggam berbasis android sesuai dengan *state* masing-masing sensor.

Daftar perintah yang dikirim oleh *microcontroller* ke Android dapat dilihat pada Tabel 3.5.

Tabel 3.5 Daftar perintah *microcontroller*

Perintah	Kondisi
n	Lampu 1 nyala
m	Lampu 1 mati
o	Lampu 2 nyala
p	Lampu 2 mati
q	Lampu 3 nyala
r	Lampu 3 mati
s	Kunci pintu 1 nonaktif
t	Kunci pintu 1 aktif
u	Kunci pintu 2 nonaktif
v	Kunci pintu 2 aktif
w	Pagar tertutup
x	Pagar terbuka

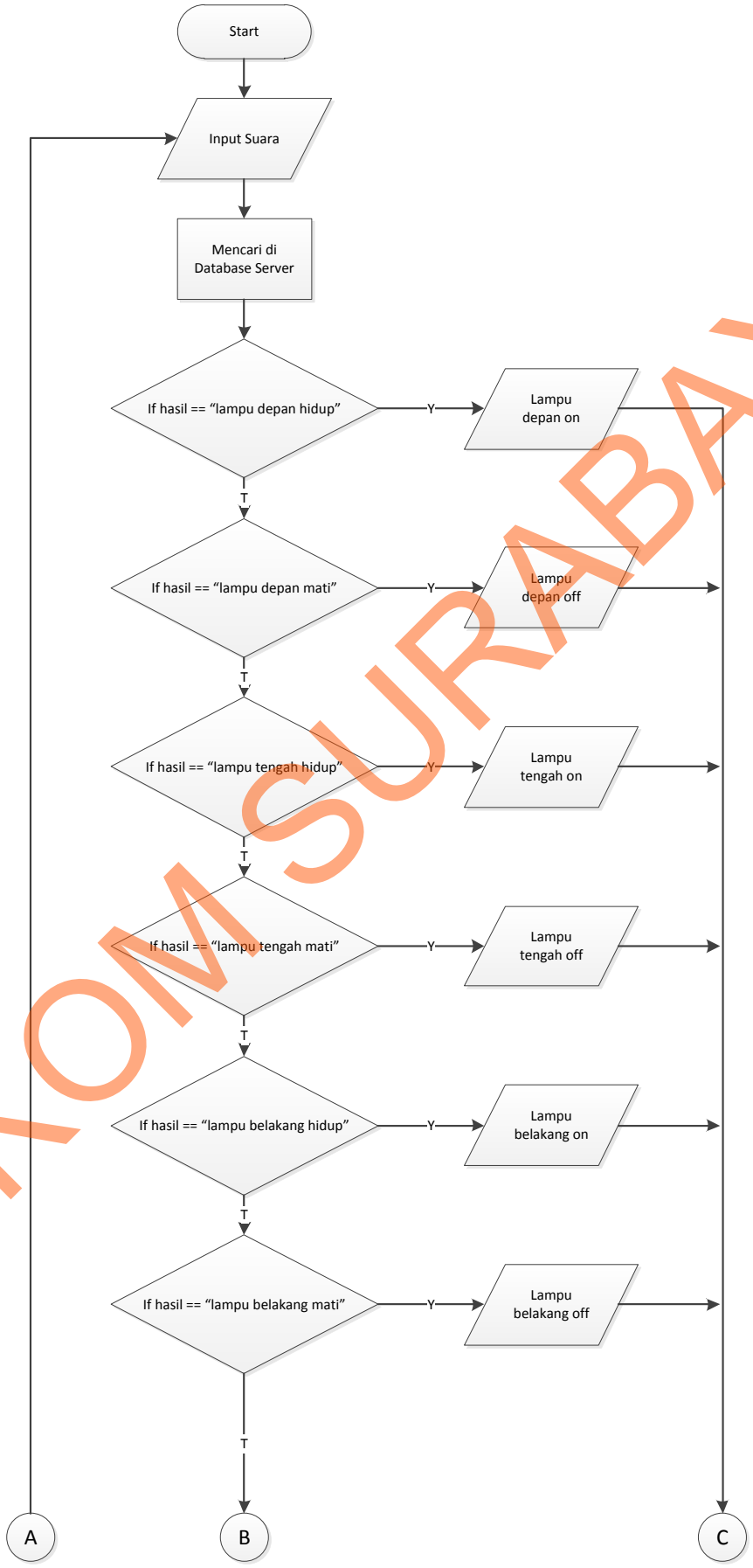
```
void kirimdata()
{
if(sensorlampu1 != lampua)
    {
        lampua=sensorlampu1;
        if(lampua){putchar('n');}
        if(!lampua){putchar('m');}
    }
     if(sensorlampu2!=lampub)
    {
        lampub=sensorlampu2;
        if(lampub){putchar('p');}
        if(!lampub){putchar('o');}
    }
     if(sensorlampu3!=lampuc)
    {
        lampuc=sensorlampu3;
        if(lampuc){putchar('r');}
        if(!lampuc){putchar('q');}
    }
      if(sensorpintu1!=pintua)
```

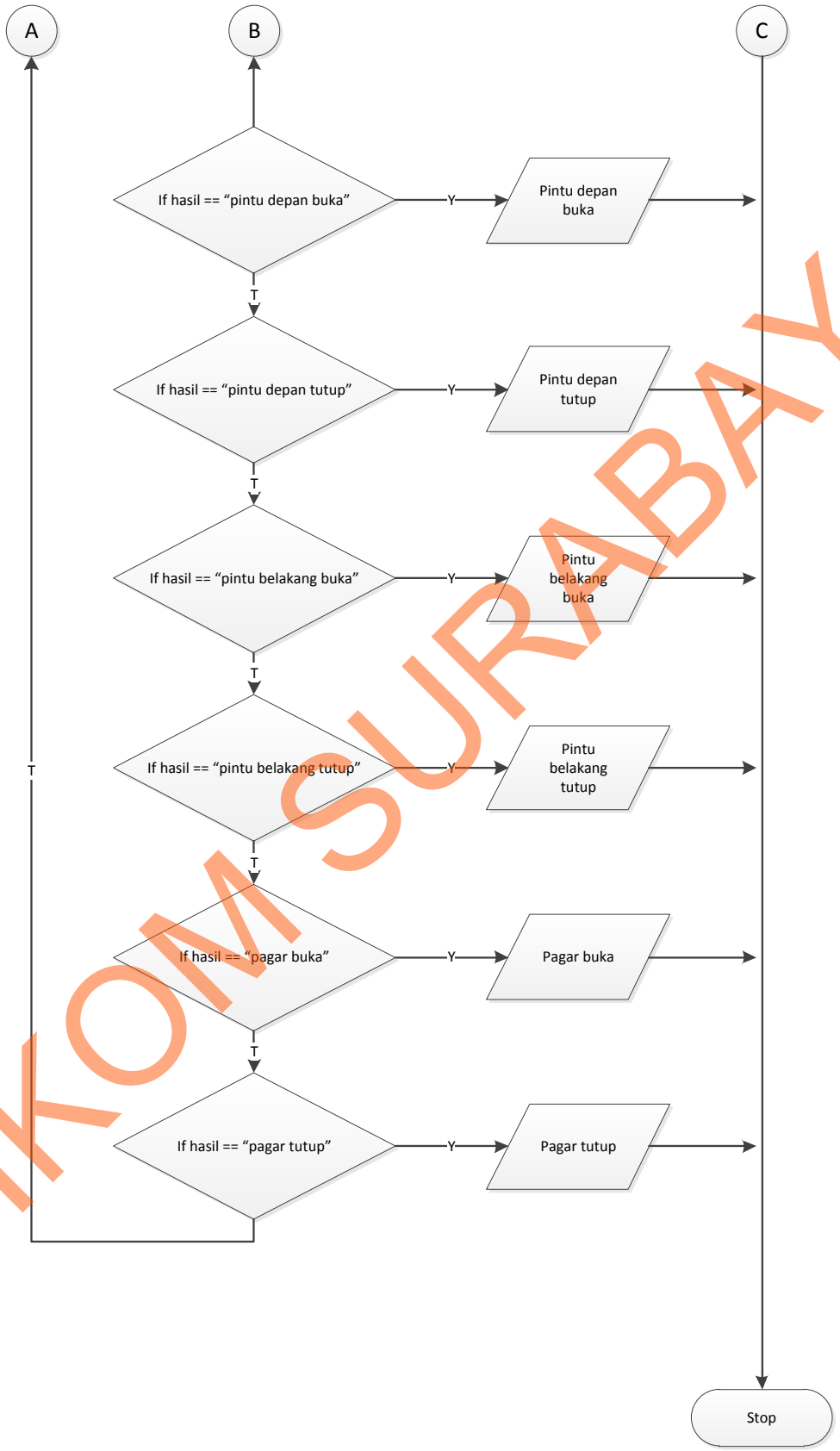
```
    {
        pintua=sensorpintu1;
        if(pintua){putchar('t');}
        if(!pintua){putchar('s');}
    }
         if(sensorpintu2!=pintub)
    {
        pintub=sensorpintu2;
        if(pintub){putchar('v');}
        if(!pintub){putchar('u');}
    }
         if(sensorpagar!=pagar)
    {
        pagar=sensorpagar;
        if(pagar){putchar('x');}
        if(!pagar){putchar('w');}
    }

}
```

3.3.3 Mengirim Data ke Microcontroller

Dalam melakukan pengiriman data ke *microcontroller* dibuat *flowchart* sebagai berikut.





Gambar 3.10 *Flowchart* Kirim Data Android

Aplikasi menunggu inputan suara dari *user*. Setelah mendapatkan inputan, maka aplikasi melakukan pencarian ke *database* servernya. Setelah itu aplikasi akan menyimpan beberapa kemungkinan dari hasil pencarian kedalam satu *variabel*. Hasil dari variabel tersebut disortir sesuai dengan inputan dari *user*. Setelah mendapat data yang cocok maka aplikasi mengirimkan data tersebut ke *microcontroller* melalui WiFi.

Daftar perintah dari *voice input* dapat dilihat pada Tabel 3.6.

Tabel 3.6 Perintah hasil *voice input*

Hasil	Aksi
“lampu depan hidup”	kirim char ‘a’
“lampu depan mati”	kirim char ‘b’
“lampu tengah hidup”	kirim char ‘c’
“lampu tengah mati”	kirim char ‘d’
“lampu belakang hidup”	kirim char ‘e’
“lampu belakang mati”	kirim char ‘f’
“pintu depan buka”	kirim char ‘g’
“pintu depan tutup”	kirim char ‘h’
“pintu belakang buka”	kirim char ‘i’
“pintu belakang tutup”	kirim char ‘j’
“pagar buka”	kirim char ‘k’
“pagar tutup”	kirim char ‘l’

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data)
{
if (requestCode == REQUEST_CODE && resultCode == RESULT_OK)
{
ArrayList<String> matches = data.getStringArrayListExtra(
RecognizerIntent.EXTRA_RESULTS);
String result="";
//String a="";
boolean found = false;
for(int i=0;i<matches.size() && !found;i++){
result=matches.get(i).toString();
//a=a+result+"\n";
if(result.equalsIgnoreCase("lampu depan hidup"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
```

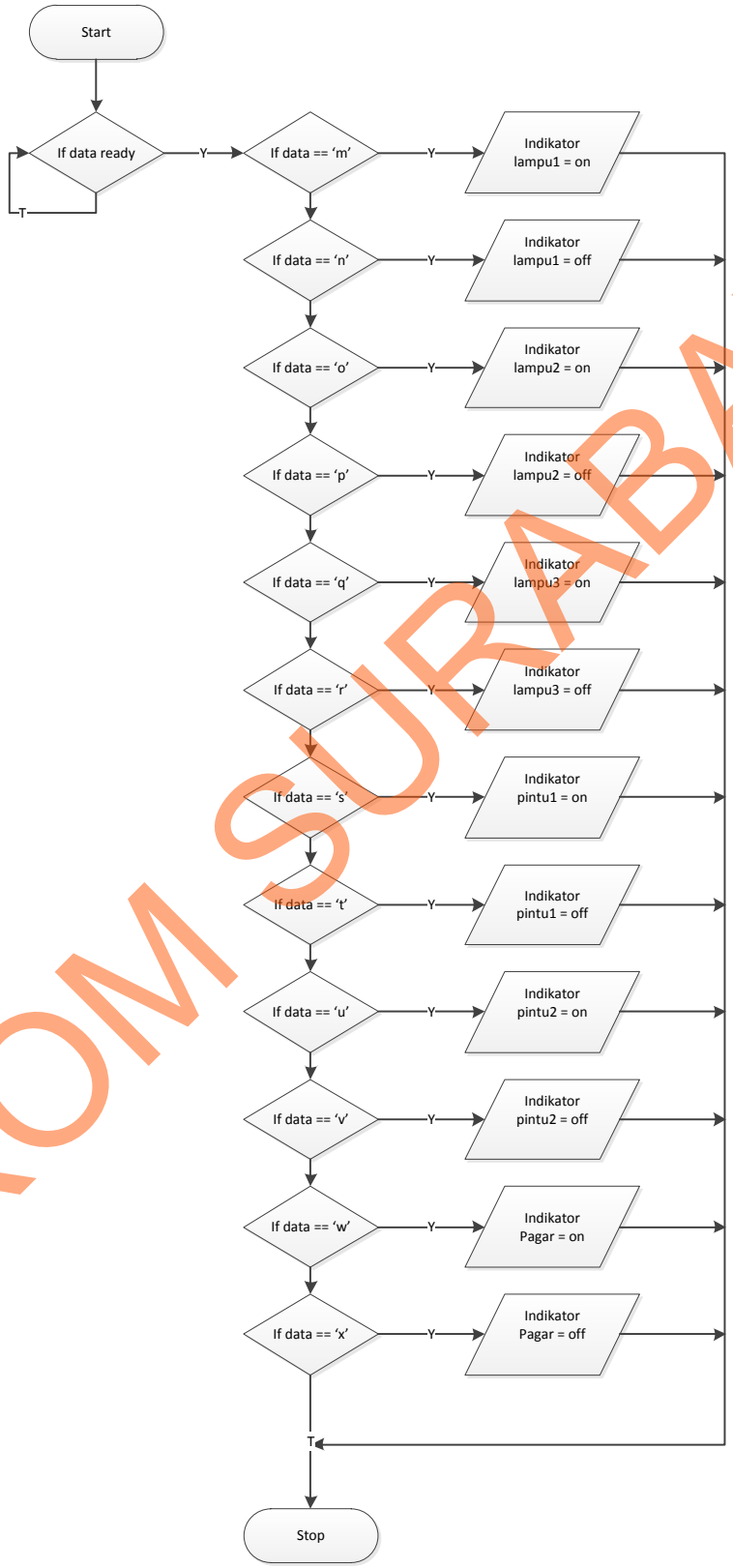
```
kirim.writeBytes("a");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("lampu depan mati"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("b");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("lampu tengah hidup"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("c");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("lampu tengah mati"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("d");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("lampu belakang hidup"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("e");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
```

```
else if (result.equalsIgnoreCase("lampu belakang mati"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("f");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("pintu depan buka"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("g");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("pintu depan tutup"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("h");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("pintu belakang buka"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("i");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("pintu belakang tutup"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("j");
```

```
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("pagar buka"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("k");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
else if (result.equalsIgnoreCase("pagar tutup"))
{
found = true;
DataOutputStream kirim;
try {
kirim = new DataOutputStream(koneksi.getOutputStream());
kirim.writeBytes("l");
}
catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}
if (!found){
Toast.makeText(getApplicationContext(), "Perintah tidak dikenali",
50000).show();
}
}
super.onActivityResult(requestCode, resultCode, data);
}
```

3.3.4 Menerima Data dari *Microcontroller*

Dalam melakukan penerimaan data dari *microcontroller* dibuat flowchart sebagai berikut.



Gambar 3.11 *Flowchart* Terima Data Android

Pada *flowchart* di atas aplikasi menunggu data dari *microcontroller*. Setelah ada data yang masuk maka data tersebut disortir sesuai dengan *database* perintah yang terdapat pada aplikasi. Apabila ada data yang cocok dengan *database*, maka indikator pada aplikasi Android akan berubah.

Untuk perintah perubahan indikator pada program Android dapat dilihat pada Tabel 3.7.

Tabel 3.7 Perintah Indikator Android

Perintah	Aksi
m	Indikator lampu 1 hijau
n	Indikator lampu 1 merah
o	Indikator lampu 2 hijau
p	Indikator lampu 2 merah
q	Indikator lampu 3 hijau
r	Indikator lampu 3 merah
s	Indikator pintu 1 hijau
t	Indikator pintu 1 hijau
u	Indikator pintu 2 merah
v	Indikator pintu 2 merah
w	Indikator pagar hijau
x	Indikator pagar merah

```
public class dataterima extends Thread
{
public dataterima()
{
start();
}
public void run()
{
while(true)
{
while(!terimakasih.dataready()){}
data=terimakasih.pesan();
runOnUiThread(new Runnable() {

public void run() {
if (data[0]=='m')
{
indikator1.setImageResource(R.drawable.greenbutton);
}
else if(data[0]=='n')
{
indikator1.setImageResource(R.drawable.redbutton);
```

```
}
else if(data[0]=='o')
{
indikator2.setImageResource(R.drawable.greenbutton);
}
else if(data[0]=='p')
{
indikator2.setImageResource(R.drawable.redbutton);
}
else if(data[0]=='q')
{
indikator3.setImageResource(R.drawable.greenbutton);
}
else if(data[0]=='r')
{
indikator3.setImageResource(R.drawable.redbutton);
}
else if(data[0]=='s')
{
indikator4.setImageResource(R.drawable.greenbutton);
}
else if(data[0]=='t')
{
indikator4.setImageResource(R.drawable.redbutton);
}
else if(data[0]=='u')
{
indikator5.setImageResource(R.drawable.greenbutton);
}
else if(data[0]=='v')
{
indikator5.setImageResource(R.drawable.redbutton);
}
else if(data[0]=='w')
{
indikator6.setImageResource(R.drawable.greenbutton);
}
else if(data[0]=='x')
{
indikator6.setImageResource(R.drawable.redbutton);
}
layar.invalidate();
teks.setText(String.valueOf(data[0]));
}
});
}
  }
```

3.4 Metode Pengujian dan Evaluasi Sistem

Untuk mengetahui apakah sistem yang dibuat dapat berjalan sesuai yang diharapkan, maka akan dilakukan pengujian dan evaluasi sistem untuk setiap tahapan-tahapan dalam pembuatan sistem. Dimulai dari pengambilan suara,

pengiriman data ke *microcontroller*, menggerakan aktuator, menerima data dari sensor, dan mengirim data ke Android.

3.4.1 Pengambilan Suara dari Google *Voice Input*

Untuk mengetahui data hasil dari pengambilan suara melalui Google *voice input* yang diterima Android, maka dilakukan pengujian dengan cara menghubungkan telepon genggam berbasis Android dengan komputer untuk mengetahui data yang diterima. Kemudian data yang diterima akan diuji apakah sesuai dengan perintah yang diberikan.

3.4.2 Pengiriman Data dari Android ke *Microcontroller*

Untuk mengetahui data yang dikirim ke *microcontroller*, maka dilakukan pengujian dengan cara menghubungkan telepon genggam berbasis Android dengan komputer untuk mengetahui data yang dikirim. Kemudian data yang diterima akan diuji apakah sesuai dengan perintah yang diberikan.

3.4.3 Menggerakkan Aktuator

Untuk mengetahui apakah microcontroller dapat menggerakkan aktuator yang terhubung dengan perangkat elektronik, maka akan dilakukan pengujian dengan cara mengirim perintah untuk menjalankan perangkat. Kemudian pergerakan dari aktuator akan diuji apakah sesuai dengan perintah yang diberikan.

3.4.4 Sensor

Untuk mengetahui apakah *microcontroller* dapat membaca data dari sensor-sensor yang terhubung dengan perangkat elektronik, maka akan dilakukan pengujian dengan cara memberikan input sesuai dengan keadaan perangkat elektronik. Kemudian akan diuji respon dari *microcontroller* ketika diberi inputan dari sensor.

3.4.5 Pengiriman Data *Microcontroller*

Untuk mengetahui apakah *microcontroller* dapat mengirim data ke telepon genggam berbasis Android, maka akan dilakukan pengujian dengan cara menghubungkan *microcontroller* dengan komputer. Kemudian data yang diterima oleh komputer akan diuji.

3.4.6 Penerimaan Data Android

Untuk menguji apakah telepon genggam berbasis android dapat menerima data yang dikirim oleh *microcontroller*, maka akan dilakukan pengujian dengan cara menghubungkan dengan komputer. Kemudian akan dilakukan pengiriman data melalui komputer. Respon dari telepon genggam berbasis Android yang mendapatkan data dari komputer akan diuji.

3.4.7 Data Serial

Untuk menguji output data serial dari modul WIZ110SR, maka akan dilakukan pengujian dengan cara mengukur output serial dengan menggunakan *ossciloscope*. Hasil pengukuran dengan ossciloscope akan diuji.

3.4.8 Evaluasi Sistem Keseluruhan

Setelah melalui seluruh proses pengujian di atas maka perlu dilakukan pengujian sistem secara keseluruhan. Dimulai dari proses penginputan suara ke telepon genggam berbasis Android. Selanjutnya melihat data hasil inputan suara yang dikirim oleh telepon genggam berbasis Android. Setelah itu melalui proses penerimaan data *microcontroller* maka aktuator akan berjalan sesuai dengan inputan suara. Kemudian proses pengiriman data yang didapat melalui proses pembacaan sensor akan mengirimkan data sesuai dari inputan sensor yang akan mengaktifkan indikator pada telepon genggam berbasis Android. Jika keseluruhan

sistem telah berjalan sesuai dengan langkah-langkah penginputan suara, *decoding*, pengiriman data ke *microcontroller*, penggerakkan aktuator berjalan sesuai dengan yang diinginkan, maka secara keseluruhan sistem ini sudah dikatakan baik.