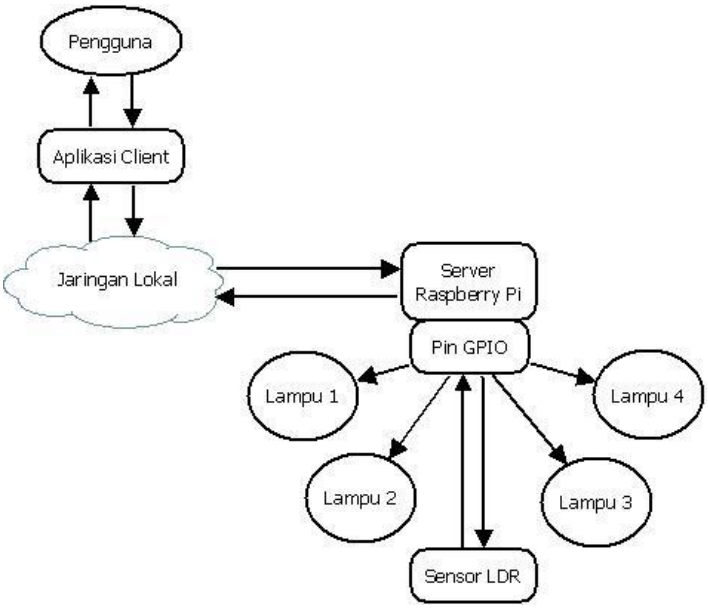
BAB III

IMPLEMENTASI

3.1. Rancangan Sistem

Sistem pada *server* pengendali lampu dengan Raspberry Pi ini bekerja saat klien dan server telah terkoneksi dalam jaringan lokal, kemudian server akan menunggu klien melakukan *request* permintaan yang akan ditangkap oleh server. *Request* dari klien tersebut akan memulai tindakan pada sistem server dimana server akan mengeksekusi apa yang telah dibuat sesuai dengan program. Hasil dari pemrosesan tersebut memiliki status yang akan dikembalikan kepada klien. Gambaran cara kerja dari keseluruhan alat dapat dilihat di Gambar 3.1.

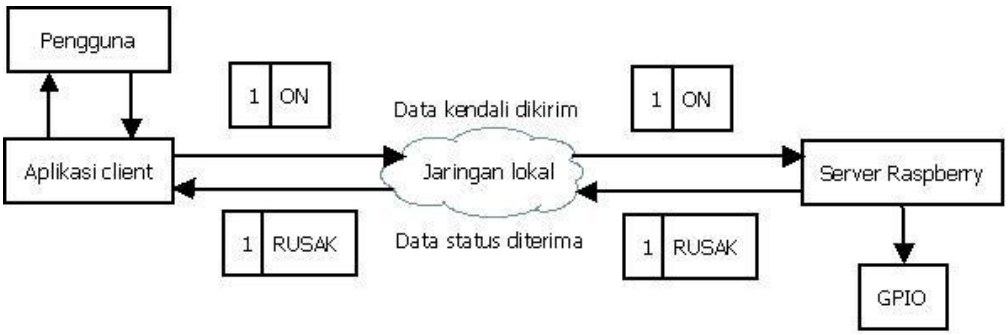


Gambar 3.1. Rancangan Sistem

3.2. Format Data Kendali dan Status

Dilihat dari gambar 3.1 Rancangan Sistem, apabila klien dan server telah terkoneksi maka server akan menunggu *request* dari klien, dimana *request* tersebut berisi data *input* bagi server untuk melakukan eksekusi.

Data *input* yang diterima oleh server akan dicocokan dengan program yang telah dibuat untuk selanjutnya dilakukan eksekusi. Data *input* ini tentunya memiliki perbedaan untuk setiap perintah yang dibaca per-karakter. Data *String* yang dikirim per-karakter dari klien ditangkap oleh server kemudian diterjemahkan untuk mengetahui tindakan apa yang akan dilakukan server terhadap pin GPIO-nya, seperti Gambar 3.2. yang menggambarkan format data sistem kerja kendali lampu ini.



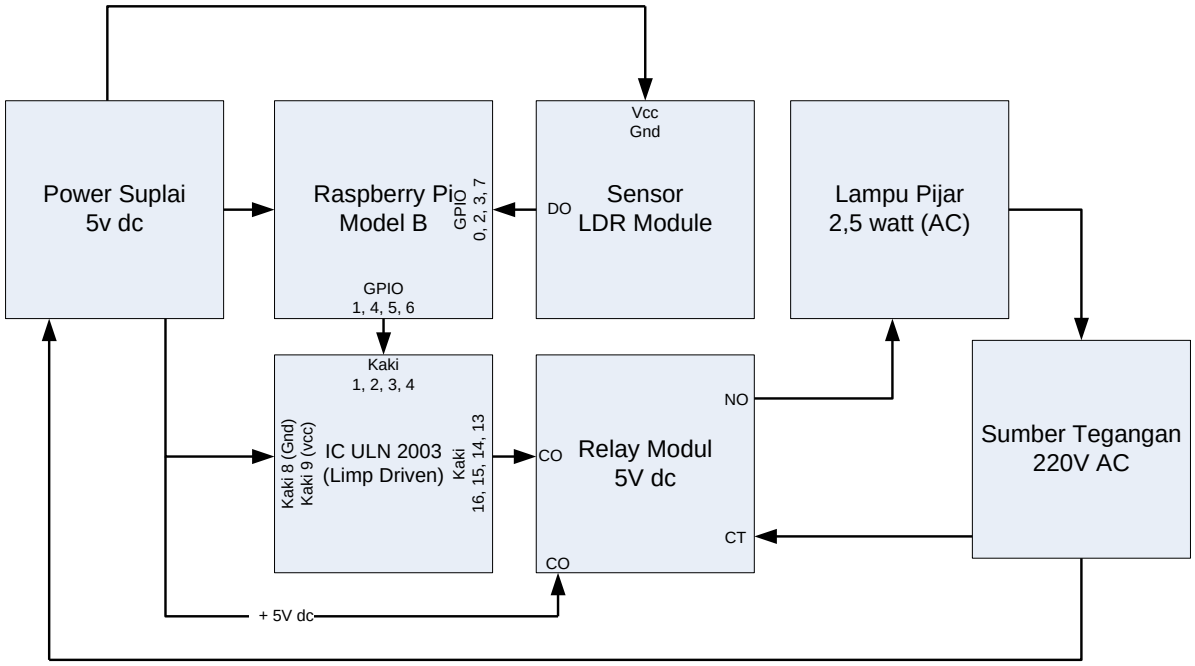
Gambar 3.2. Format Data Kendali dan Status

Setelah server melakukan eksekusi dari *request* klien, server akan memberikan *feedback* berupa data status yang diperoleh dari sensor yang terdapat pada sisi server. Server hanya akan mengirimkan status jika terjadi masalah atau kerusakan pada lampu yang dipilih untuk dinyalakan dalam bentuk data *string*.

3.3. Rancangan Hardware dan Software

3.3.1. Rancangan Hardware

Gambar 3.3 menggambarkan secara keseluruhan sistem kerja pada *hardware* yaitu Raspberry Pi terhubung dengan IC ULN2003 sebagai *driver* untuk menggerakan kontaktor pada relay sehingga jalur pada lampu akan terhubung dan menyalakan lampu. Setelah itu sensor cahaya akan membaca keadaan lampu untuk memberikan masukan pada raspberry yang akan dijadikan status untuk dikirimkan ke client.



Gambar 3.3 Diagram Blok Hardware

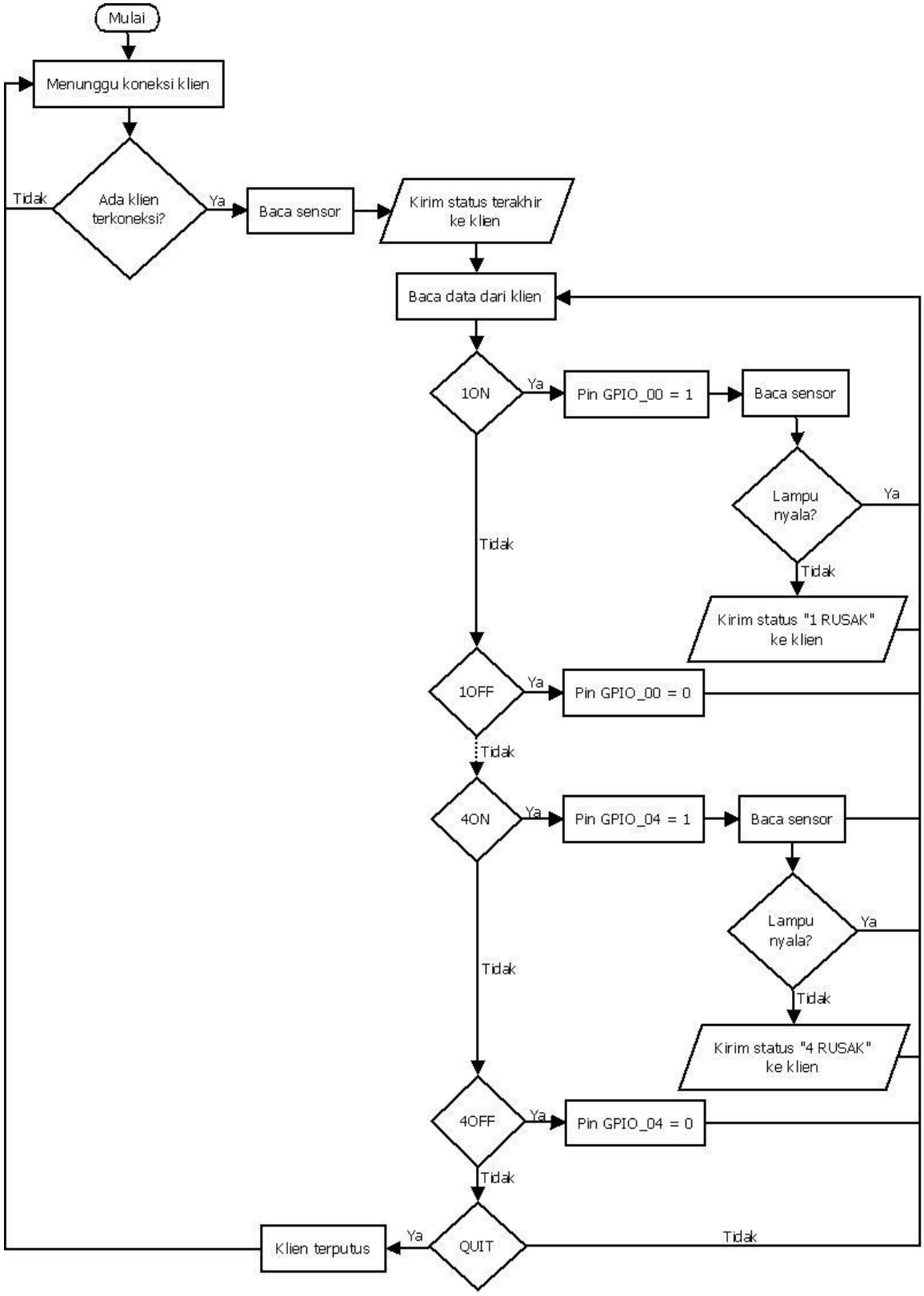
3.3.2. Rancangan Software

Gambar 3.4 menggambarkan alur program yang dibuat pada server. Ketika server dijalankan, server akan menunggu koneksi dari *client*, setelah koneksi terhubung maka server akan mengirimkan status keadaan lampu terlebih dahulu kepada klien, kemudian server menunggu *request* dari *client* dan mengeksekusinya. Dari hasil eksekusi tersebut server akan memberikan informasi status sesuai dengan aslinya.

Jika data yang dikirim klien terdapat karakter yang dibaca “ON” maka server akan memberikan nilai 1 pada salah satu pin-nya sesuai karakter pertama yang dikirim klien (lampu nyala), selanjutnya program akan membaca *method* sensor yang akan memastikan keadaan lampu menyala atau mengalami kerusakan. Jika lampu menyala server akan kembali menunggu data selanjutnya dari klien, namun jika lampu rusak server akan mengirimkan status terlebih dahulu ke klien yang menyatakan lampu tersebut rusak baru kemudian server kembali menunggu data dari klien.

Jika data yang dikirim klien terdapat karakter yang dibaca “OFF” maka server akan memberikan nilai 0 pada salah satu pin GPIO-nya (lampu padam), kemudian server kembali menunggu data dari klien.

Jika data yang dikirim klien terdiri dari karakter yang dibaca “QUIT” maka server akan memutuskan koneksi tersebut dan menunggu koneksi kembali dari klien.

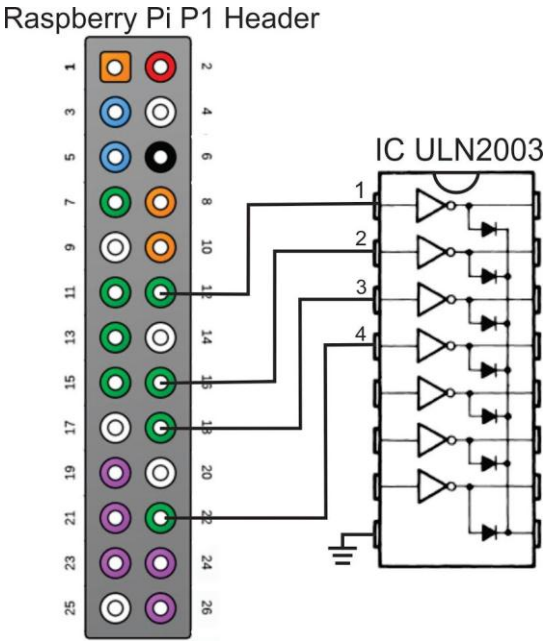


Gambar 3.4. Rancangan Software

3.4. Implementasi

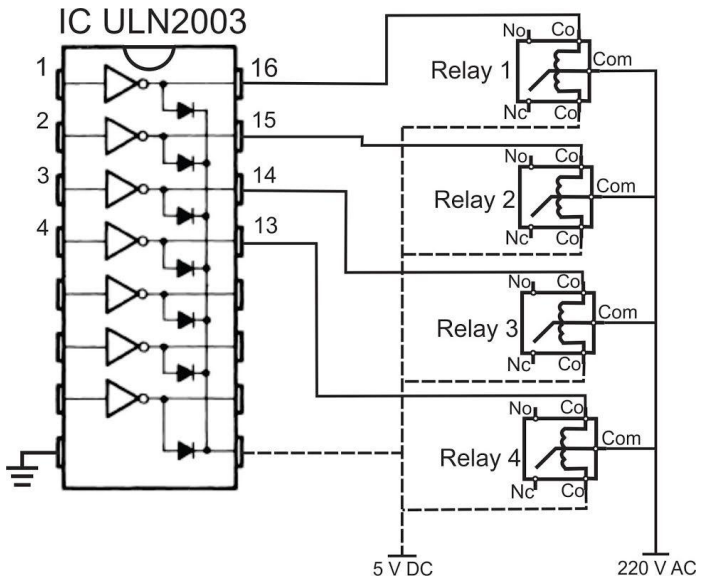
3.4.1. Hardware

Dari rancangan sistem *hardware* pada Gambar 3.3 dapat dijelaskan lebih rinci untuk setiap sambungan rangkaian.



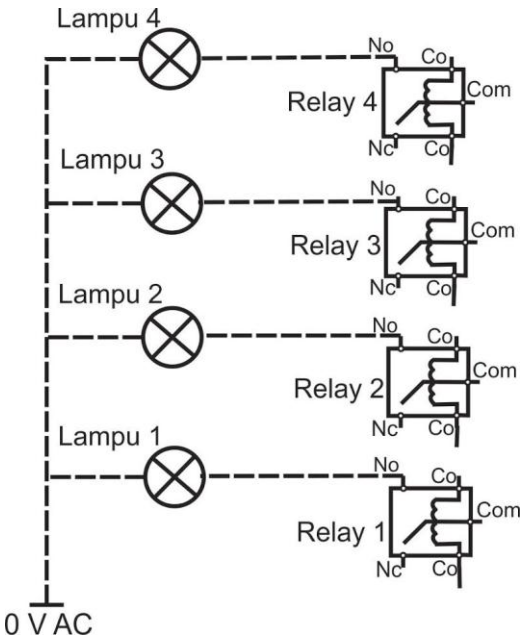
Gambar 3.5. Raspberry Menuju IC ULN 2003

Pin raspbery (12, 16, 18, dan 22) raspberry terhubung ke IC ULN 2003. Pin ini merupakan Pin GPIO (1, 4, 5, dan 6) yang digunakan sebagai jalur *output* raspberry yang akan masuk ke kaki IC 1, 2, 3 dan 4 seperti pada Gambar 3.5 sebagai *input* untuk IC ULN2003 yang berfungsi sebagai *driver* relay.



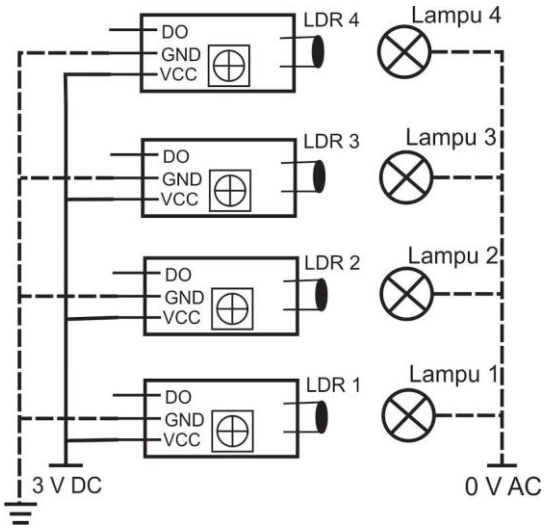
Gambar 3.6. IC ULN 2003 Menuju Relay

Pada IC ULN 2003 kaki dengan nomor 16, 15, 14 dan 13 disambungkan ke kaki Coil pada tiap-tiap relay seperti pada Gambar 3.6. jalur ini merupakan *output* dari IC yang dihubungkan ke relay. Untuk kaki Coil yang lain pada tiap-tiap relay deberikan tegangan sebesar 5 Volt DC agar relay dapat bekerja. Kaki Com atau CT relay tersambung langsung pada tegangan 220 Volt AC atau tegangan PLN.



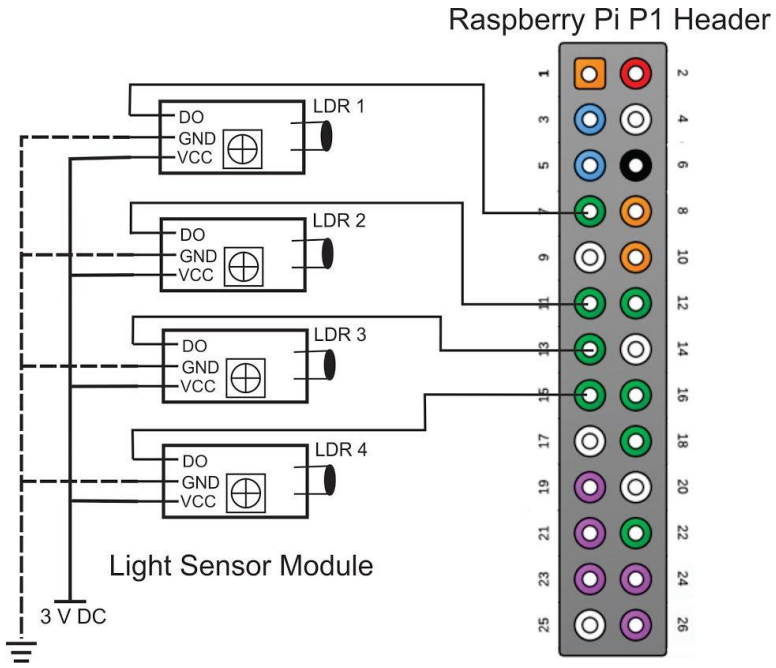
Gambar 3.7. Relay Menuju Lampu

Kaki No pada tiap-tiap relay di sambungkan pada lampu yang ada seperti pada Gambar 3.7. Kaki No relay akan tersambung dengan kaki Com pada saat relay bekerja dan akan menghidupkan lampu.



Gambar 3.8. Lampu Menuju Sensor

Light Sensor Modul dengan LDR dipasang berdekatan dengan tiap-tiap lampu seperti pada gambar 3.8. Sensor ini akan menangkap intensitas cahaya saat lampu menyala. Agar tidak terjadi gangguan dari cahaya yang diterima maka tingkat intensitas cahaya lampu harus disesuaikan.



Gambar 3.9. Sensor Menuju Pin Raspberry

Pada Gambar 3.9. *output* sensor pada pin DO (*Digital Output*) dihubungkan ke pin 7, 11, 13 dan 15. Yang akan dijadikan *input* agar raspberry dapat mebaca keadaan lampu yang terpasang.

Gambar rangkaian secara keseluruhan dapat dilihat pada lembar lampiran.

3.4.2. Software

Pembuatan program aplikasi server yang akan digunakan untuk pengontrol lampu, menggunakan pemrograman pemrograman soket dengan bahasa pemrograman Java. GPIO diakses menggunakan *library* pi4j dengan IDE NetBeans.

Aplikasi server yang dibuat dengan tools NetBeans memiliki ekstensi file .jar. Di dalam aplikasi ini memiliki dua kelas, yang menjadi satu kesatuan yang telah dibuat dalam bentuk aplikasi berekstensi .jar. Kelas pertama berisi program penanganan koneksi server dan klien serta penanganan untuk *input* dan *output* data pada server. Sedangkan kelas kedua berisi program penanganan untuk mengontrol lampu serta membaca *input* status yang deperoleh dari *Light Sensor Module*.

Program pada kelas pertama (Raspberry.java)

Pembuatan import kelas pertama

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
```

Import yang digunakan pada kelas pertama digunakan untuk penanganan soket dan data *input/output*.

Pembuatan kelas Raspberry

```
public class Raspberry extends Thread {

    Socket server = null;
    DataInputStream in = null;
    DataOutputStream out = null;

    public Raspberry(Socket socket) {
        this.server = socket;
    }
```

Pada kelas Raspberry terdapat inisialisasi variabel diantaranya variable soket, data *input* dan data *output*.

Pembuatan *method* run()

```
public void run() {
   try {
      in = new DataInputStream
        (server.getInputStream());
      out = new DataOutputStream
        (server.getOutputStream());
      System.out.println("Client connected...");

String lastStatus = SensorLed.sensor1.getState()+","+

       SensorLed.sensor2.getState()+ ","+

       SensorLed.sensor3.getState()+","+

       SensorLed.sensor4.getState();

out.writeUTF(lastStatus);
```

Program diatas akan menyiapkan variable in dan out, untuk data input dan output saat klien telah terkoneksi dan mengirimkan status terakhir keadaan lampu ke klien.

Pembuatan perulangan

```
while (true) {
   String input;
   while ((input = in.readUTF()) != null &&
     !input.startsWith("QUIT")) {
```

```
        if (input.startsWith("1ON")) {
            SensorLed.lampu_1.high();
            cekStatusLampu("1","NYALA");
        }
        else if (input.startsWith("1OFF")) {
            SensorLed.lampu_1.low();
            cekStatusLampu("1","MATI");
            }
        else if (input.startsWith("2ON")) {
            SensorLed.lampu_2.high();
            cekStatusLampu("2","NYALA");
            }
        else if (input.startsWith("2OFF")) {
            SensorLed.lampu_2.low();
            cekStatusLampu("2","MATI");
            }
        else if (input.startsWith("3ON")) {
            SensorLed.lampu_3.high();
            cekStatusLampu("3","NYALA");
            }
        else if (input.startsWith("3OFF")) {
            SensorLed.lampu_3.low();
            cekStatusLampu("3","MATI");
            }
        else if (input.startsWith("4ON")) {
            SensorLed.lampu_4.high();
            cekStatusLampu("4","NYALA");
            }
        else if (input.startsWith("4OFF")) {
            SensorLed.lampu_4.low();
            cekStatusLampu("4","MATI");
       }
}
```

Kode program diatas digunakan untuk menangani data yang didapat dari *request* klien (*input*). Berbentuk perulangan dengan menggunakan while maka selama nilainya benar (klien terkoneksi) ini akan terus diulang. Data yang masuk baru akan dicocokan jika data dari klien tidak sama dengan null. Pencocokan data akan menentukan nilai pada pin Raspberry (nilai 1 = high, nilai 0 = low).

Pembuatan program klien *disconnect*

```
        out.close();
        in.close();
        server.close();
         System.out.println("Client disconnected...");
        }
     } catch (IOException ex) {
     }
 }
```

Kode program diatas berfungsi untuk menangani keadaan ketika klien memutuskan koneksi. Kejadian ini ditandai dengan tampilnya pesan “Client disconnected…”.

Pembuatan *method* cekStatusLampu()

```
private void cekStatusLampu(String lampu, String
onOff){
   try {
      Thread.sleep(500);
      if (!SensorLed.isStatus()) {
         lampu = "1 RUSAK";
      } else {
          lampu = "1"+onOff;
      }
      out.writeUTF(status); ke client
      SensorLed.setStatus(false);
   } catch (IOException | InterruptedException ex) {
      System.out.println("Kesalahan (cekStatusLampu)
       "+ex.getMessage());
   }
}
```

Method cekStatusLampu digunakan untuk melakukan pengecekan terhadap kondisi lampu dan mengirimkan status “RUSAK” ke klien jika lampu mengalami kerusakan.

Pembuatan *method* main()

```
public static void main(String[] args) {
   final int PORT = 5555;
   SensorLed sensorLed = new SensorLed();
   try {
```

```
        System.out.println("Server running..");
        ServerSocket listener = new ServerSocket(PORT);
        while (true) {
            Raspberry raspberry = new
              Raspberry(listener.accept());
            raspberry.start
        }
    } catch (IOException ex) {
    System.out.println("Terjadi kesalahan " +
      ex.getMessage());
    }
}
```

Method ini berisi nomor port yang dipakai (5555), pemanggilan kelas SensorLed(), serta penanganan ketika ada klien masuk di port “5555”. Method ini juga yang akan dijalankan pertama kali ketika program berjalan yang ditandai dengan tampilnya pesan “Server running..”.

Program pada kelas kedua (SensorLed.java)

Pembuatan import kelas kedua

```
import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.io.gpio.GpioPinDigitalInput;
import com.pi4j.io.gpio.GpioPinDigitalOutput;
import com.pi4j.io.gpio.PinPullResistance;
import com.pi4j.io.gpio.PinState;
import com.pi4j.io.gpio.RaspiPin;
import
com.pi4j.io.gpio.event.GpioPinDigitalStateChangeEvent;
import com.pi4j.io.gpio.event.GpioPinListenerDigital;
```

Import yang digunakan pada kelas kedua yang akan menangani *input/output* pada pin GPIO raspberry.

Pembuatan kelas SensorLed

```
public class SensorLed {
static GpioPinDigitalOutput lampu_1, lampu_2, lampu_3,
  lampu_4;
```

```
GpioPinDigitalInput sensor_1, sensor_2, sensor_3,
  sensor_4;
static boolean status = false;
```

Program di atas menginisialisasi variable lampu, sensor dan status.

Deklarasi pin GPIO lampu

```
public SensorLed() {

   final GpioController gpio =
    GpioFactory.getInstance();

   lampu_1=gpio.provisionDigitalOutputPin(RaspiPin.GPI
    O_00, "LED #1", PinState.LOW);
   lampu_2=gpio.provisionDigitalOutputPin(RaspiPin.GPI
    O_01, "LED #2", PinState.LOW);
   lampu_3=gpio.provisionDigitalOutputPin(RaspiPin.GPI
    O_02, "LED #3", PinState.LOW);
   lampu_4=gpio.provisionDigitalOutputPin(RaspiPin.GPI
    O_03, "LED #4", PinState.LOW);
```

Deklarasi pin GPIO yang digunakan untuk lampu. Pin yang dideklarasikan diatas diberi nilai awal 0 (low).

Deklarasi pin GPIO sensor

```
sensor1=
  gpio.provisionDigitalInputPin(RaspiPin.GPIO_04,
  PinPullResistance.PULL_DOWN);
sensor1.addListener(new GpioPinListenerDigital() {
  Public void
    handleGpioPinDigitalStateChangeEvent(GpioPinDigita
    lStateChangeEvent event) {
    if((lampu_1.getState().isHigh()&&
      event.getState().isHigh())||lampu_1.getState().i
      sLow()&& event.getState().isLow())) {
         setStatus(false); //false = rusak
    }
    else {
      setStatus(true);
    }
  }
});

sensor2 =
  gpio.provisionDigitalInputPin(RaspiPin.GPIO_05,
  PinPullResistance.PULL_DOWN);
```

```
sensor2.addListener(new GpioPinListenerDigital(){
  Public void
   handleGpioPinDigitalStateChangeEvent(GpioPinDigita
   lStateChangeEvent event) {
   if ((lampu_2.getState().isHigh() &&
    event.getState().isHigh())||(lampu_2.getState().
    isLow() && event.getState().isLow())) {
       setStatus(false);
   }
   else {
       setStatus(true);
   }
  }
});

sensor3 =
 gpio.provisionDigitalInputPin(RaspiPin.GPIO_06,
 PinPullResistance.PULL_DOWN);
sensor3.addListener(new GpioPinListenerDigital(){
 public void
  handleGpioPinDigitalStateChangeEvent(GpioPinDigital
  StateChangeEvent event) {
   if ((lampu_3.getState().isHigh() &&
    event.getState().isHigh())||(lampu_3.getState().
    isLow() && event.getState().isLow())) {
      setStatus(false);
   }
   else {
      setStatus(true);
   }
  }
});

sensor4 =
 gpio.provisionDigitalInputPin(RaspiPin.GPIO_07,
 PinPullResistance.PULL_DOWN);
sensor4.addListener(new GpioPinListenerDigital(){
 public void
  handleGpioPinDigitalStateChangeEvent(GpioPinDigital
  StateChangeEvent event) {
   if ((lampu_4.getState().isHigh() &&
    event.getState().isHigh())||(lampu_4.getState().
    isLow() && event.getState().isLow())) {
      setStatus(false);
   }
   else {
      setStatus(true);
   }
  }
});
```

Program diatas mendeklarasikan 4 pin GPIO raspberry yang digunakan untuk sensor LDR. Program diatas juga berfungsi untuk pembacaan data pada sensor.

Pembuatan *method* isStatus() dan setStatus()

```
public static boolean isStatus() {
        return status;
    }

public static void setStatus(boolean status) {
        SensorLed.status = status;
    }
```

Kedua *method* diatas akan menampung dan menangani data yang didapat dari keadaan sensor dan akan dijadikan status lampu.

3.5. Pengujian Hardware dan Software

3.5.1. Pengujian Hardware

Hasil pengujian *hardware* disajikan dalam bentuk tabel dengan menggunakan alat ukur multimeter analog merek Sunwa YX-360.

Table 3.1. Output IC ULN 2003 Lampu ON

No.	Kaki	Output	Kondis Lampu
1	16	5 volt	Lampu 1 ON
2	15	5 volt	Lampu 2 ON
3	14	5 volt	Lampu 3 ON
4	13	5 volt	Lampu 4 ON

Tabel 3.1 merupakan hasil pengujian *output* pada IC ULN2003 pada saat kondisi lampu menyala. Pengukuran

menggunakan alat ukur multimeter analog merek Sunwa YX-360, dengan tegangan sumber 5 volt dc.

Table 3.2. Output IC ULN 2003 Lampu OFF

No	Kaki	Output	Kondis Lampu
1	16	0,2 volt	Lampu 1 OFF
2	15	0,2 volt	Lampu 2 OFF
3	14	0,2 volt	Lampu 3 OFF
4	13	0,2 volt	Lampu 4 OFF

Tabel 3.2 merupakan hasil pengujian *output* pada IC ULN2003 pada saat kondisi lampu mati. Pengukuran menggunakan alat ukur multimeter analog merek Sunwa YX-360, dengan tegangan sumber 5 volt dc.

Table 3.3. Output Sensor Modul LDR Lampu ON

Sensor	PIN	Output	Kondis Lampu
1	Digital Output	4,5 volt	Lampu 1 OFF
2	Digital Output	4,5 volt	Lampu 2 OFF
3	Digital Output	4,5 volt	Lampu 3 OFF
4	Digital Output	4,5 volt	Lampu 4 OFF

Tabel 3.3 merupakan hasil pengujian *output* pada Light Sensor Module pada saat kondisi lampu menyala. Pengukuran menggunakan alat ukur multimeter analog merek Sunwa YX-360, dengan tegangan sumber 5 volt dc.

Table 3.4. Output Sensor Modul LDR Lampu OFF

Sensor	Kaki	Output	Kondis Lampu
1	Digital Output	0 volt	Lampu 1 OFF
2	Digital Output	0 volt	Lampu 2 OFF
3	Digital Output	0 volt	Lampu 3 OFF
4	Digital Output	0 volt	Lampu 4 OFF

Tabel 3.4 merupakan hasil pengujian *output* pada Light Sensor Module pada saat kondisi lampu mati. Pengukuran menggunakan alat ukur multimeter analog merek Sunwa YX-360, dengan tegangan sumber 5 volt dc.

3.5.2. Pengujian Software

Masuk ke direktori tempat file program disimpan kemudian jalankan atau running program server dengan perintah “sudo java –jar Raspberry.jar”. Ketika program server telah berjalan maka akan menampilkan pesan “server running..”, seperti pada Gambar 3.10.

```
pi@raspberrypi ~/NetBeansProjects/Raspberry/dist $ sudo java -jar Raspberry.jar
server running..
```

Gambar 3.10. Server Running

“Client connected..” seperti pada Gambar 3.11 klien telah terkoneksi dengan server.

```
pi@raspberrypi ~/NetBeansProjects/Raspberry/dist $ sudo java -jar Raspberry.jar
server running..
client connected...
```

Gambar 3.11. Client Connected

“Client disconnected..” seperti pada Gambar 3.12.

```
pi@raspberrypi ~/NetBeansProjects/Raspberry/dist $ sudo java -jar Raspberry.jar
server running..
client connected...
client disconnected...
```

Gambar 3.12. Client Disconnected

3.5.3. Pengujian Sistem

Pada gambar 3.13 merupakan pengujian saat klien menyalakan lampu 1 dan 4. Setelah server menerima request dari klien maka server akan menyalakan lampu 1 dan 4 seperti pada Gambar 3.14.

Gambar 3.13. Aplikasi

Gambar 3.14. Prototype

Pada Gambar 3.15 merupakan hasil pengujian ketika klien menyalakan lampu 1, 3 dan 4, sedangkan pada lampu 2 mengalami kerusakan.Hasil pada server dapat dilihat pada Gambar 3.16.

Gambar 3.15. Aplikasi

Gambar 3.16. Prototype