IMPLEMENTASI PROTOKOL *UNIVERSAL PLUG AND PLAY* (UPnP) PADA SENSOR DAN AKTUATOR UNTUK OTOMASI LAMPU

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Muhammad Mahar Jahary
NIM: 145150300111004

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

IMPLEMENTASI PROTOKOL *UNIVERSAL PLUG AND PLAY* (UPnP) PADA SENSOR DAN AKTUATOR UNTUK OTOMASI LAMPU

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Muhammad Mahar Jahary
NIM : 145150300111004

Skripsi ini telah diuji dan dinyatakan lulus pada
18 Juli 2019
Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Mochammad Hannats Hanafi Ichsan, S.ST, M.T.	Rakhmadhany Primananda, S.T, M.Kom.
NIP. 19881229 201903 1010	NIK. 201609 860406 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D.
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 18 Juli 2019

Muhammad Mahar Jahary
NIM : 145150300111004

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga penulis dapat menyelesaikan laporan skripsi yang berjudul "Implementasi Protokol *Universal Plug and Play* (UPnP) pada Sensor dan Aktuator untuk Otomasi Lampu". Skripsi ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar Sarjana Teknik Fakultas Ilmu Komputer Univesitas Brawijaya.

Penulis menyadari bahwa skripsi ini tidak lepas dari bantuan banyak pihak, sehingga pada kesempatan ini dengan segala kerendahan hati dan penuh rasa hormat penulis menghaturkan terima kasih yang sebesar-besarnya terutama untuk :

1. Kepada Orang Tua penulis Munaji dan Yun Kusumastuti yang selalu mendoakan, memberikan motivasi dan pengorbanan baik dari segi moril maupun materi kepada penulis sehingga penulis dapat menyelesaikan skripsi ini. Untuk Nugraheny Wahyu Try selaku sahabat terbaik, sahabat dari kbmcl, pkl xbee, dan grup coro yang selalu mendukung dan doanya,
2. Bapak Mochammad Hannats Hanafi Ichsan, S.ST, M.T. dan Bapak Rakhmadhany Primananda, S.T, M.Kom. selaku Pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis,
3. Bapak Dahnial Syauqy, S.T., M.T., M.Sc. selaku Ketua Program Studi Teknik Komputer,
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika,
5. Bapak Adharul Muttaqin, ST., MT. dan Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng. selaku dosen Penasihat Akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi,
6. Bapak Adharul Muttaqin, ST., MT. yang pertama kali menunjukkan ke penulis bagaimana cara menulis kode program,
7. Terimakasih juga kepada semua pihak yang telah membantu dalam penyelesaian skripsi ini yang tidak dapat disebutkan satu per satu.

Malang, 18 Juli 2019

penulis
maharjahary@gmail.com

ABSTRAK

Muhammad Mahar Jahary, Implementasi Protokol *Universal Plug and Play* (UPnP) pada Sensor dan Aktuator untuk Otomasi Lampu

Pembimbing: Mochammad Hannats Hanafi Ichsan, S.ST, M.T dan Rakhmadhany Primananda, S.T, M.Kom

Rumah pintar merupakan teknologi yang berada di dalam rumah yang mana dapat melakukan berbagai pekerjaan agar lebih mudah. Namun, untuk saat ini masih banyak teknologi yang terdapat pada rumah pintar belum terintegrasi antara perangkat satu dengan yang lainnya dan memiliki perangkat kendali masing-masing. Maka dari itu perlu standardisasi seperti halnya dalam penelitian ini dengan memanfaatkan protokol *Universal Plug and Play* (UPnP). UPnP memiliki kemampuan untuk melakukan pengenalan perangkat dan layanan yang dapat dijalankan dalam berbagai bahasa pemrograman. Dalam penelitian ini UPnP diambil sebagai standardisasi untuk berbagai komunikasi antar perangkat. Untuk setiap perangkat menggunakan mikrokontroler raspberry pi 3 yang memiliki *ethernet* atau wifi sebagai jalur komunikasi antar perangkat. Untuk perangkat *smart home* dalam penelitian ini yaitu ada dua, yang pertama adalah perangkat sensor pergerakan dan yang ke dua adalah perangkat sensor cahaya. Perangkat sensor pergerakan menggunakan sensor PIR dan aktuatornya adalah lampu dengan penghubung menggunakan *relay* sedangkan perangkat sensor cahaya menggunakan sensor LDR sebagai masukannya. Untuk *control point* dapat melakukan kendali atau hanya mencari perangkat yang berada pada jaringan tersebut menggunakan raspberry pi. Dalam pengujian terdapat waktu respon dari setiap skenario yang telah dibuat. Pengujian waktu rata-rata satu perangkat sensor dapat terhubung pertama kali dengan pengguna yaitu sebesar 293,32 milikdetik, untuk rata-rata waktu respon satu perangkat mengirim data kepada pengguna setelah dikenali sebesar 165,76 milidetik sedangkan untuk rata-rata waktu respon dua perangkat sensor terhubung untuk pertama kali dengan pengguna secara bersamaan yaitu 726 milidetik.

Kata kunci: *Universal Plug and Play (UPnP),* otomasi lampu*, Raspberry pi, smart home*

ABSTRACT

Muhammad Mahar Jahary, Implementasi Protokol *Universal Plug and Play* (UPnP) pada Sensor dan Aktuator untuk Otomasi Lampu

Supervisors: Mochammad Hannats Hanafi Ichsan, S.ST, M.T and Rakhmadhany Primananda, S.T, M.Kom

Smart home is the right technology at home which can do various jobs to make it easier. However, for now there are still many technologies available in smart homes that have not been installed between one device and another and have their own access devices. Therefore it is necessary to standardize as supported in this study by using the Universal Plug and Play (UPnP) protocol. UPnP has the ability to run devices and services that can be run in various programming languages. UPnP is taken as standardization for various communications between devices. For each device that uses a Raspberry Pi 3 microcontroller that has Ethernet or WiFi as a communication path between devices. There are two smart home devices in this study, the first is a motion sensor device and the second is a light sensor device. The motion sensor device uses a PIR sensor and its actuator is a lamp with a connector using a relay while the light sensor device uses the LDR sensor as its input. For control points you can use the device or search for suitable devices on this network using raspberry pi. In testing there is a response time for each scenario that has been made. Testing the average time of a sensor device can connect the first time with a user of 293.32 seconds, for the average response time of one device to send data to the user after being identified at 165.76 milliseconds while for the average response time of two sensor devices 726 millisecond.

Keywords: Universal Plug and Play (UPnP), lamp automation, Raspberry pi, smart home

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Manusia di dunia membutuhkan yang namanya rumah untuk tempat tinggal dan rumah sendiri menjadi kebutuhan yang sangat penting sebagai tempat berteduh. Seiring berjalannya jaman dan berkembangnya teknologi maka muncul sistem yang diterapkan di dalam rumah yang bernama *smart home. Smart home* adalah teknologi yang menggunakan sistem kerja cerdas dan otomatis dalam konsep kerjanya untuk membantu pekerjaan penghuni rumah dalam melakukan antivitas lebih mudah dengan konsep otomatis dibanding melakukannya secara manual (aditya, et al., 2015).

Dalam rumah cerdas terdapat peralatan listrik yang sering digunakan yaitu lampu atau pencahayaan. Ada sebuah sistem kontrol lampu otomatis yang mana dapat mengurangi campur tangan manusia dalam operasionalnya. Sistem kontrol lampu otomatis akan berdampak pada efisiensi dan efektif dalam menghindari penggunaan lampu yang menyala secara sia-sia tanpa adanya aktivitas. Tujuannya adalah untuk menghindari atau mengurangi dampak dari pemborosan pada energi listrik (Handoko, 2016). Namun masih banyak rumah yang belum memakai sistem rumah cerdas yang berfokus pada sistem lampu dan masih memakai sakelar analog yang menyebabkan konsumsi energi listrik yang cenderung berlebihan dan kurang tepat (Ridwanda, et al., 2014).

Pada sistem kontrol lampu otomatis sudah banyak metode komunikasi yang digunakan untuk penerapan rumah cerdas. Metode komunikasi tersebut antara lain menggunakan jaringan GSM (Pragmawati, 2016), *web service* (Masykur & Prasetiyowati, 2016), dan radio frekuensi (Muchlas, et al., 2006). Dengan banyaknya metode untuk kontrol lampu otomatis maka akan mempersulit kontrol karena banyak ketidakcocokan antara metode satu dengan metode lainya. Solusi agar bisa menghubungkan metode komunikasi satu dengan yang lainnya dapat menggunakan protokol *Universal Plug and Play* (UPnP).

Universal Plug and Play (UPnP) adalah arsitektur yang memungkinkan suatu *device* maupun aplikasi untuk dapat terhubung di dalam sebuah jaringan, dapat berkomunikasi dengan *device* atau aplikasi lainnya yang berada dalam jaringan dan tanpa perlu adanya konfigurasi secara manual serta menghilangkan batasan dalam sistem operasi maupun bahasa pemrograman untuk dapat digunakan pada pembuatan sebuah aplikasi. Dengan sifat universal pada UPnP peralatan elektronik dapat secara dinamis terhubung dalam jaringan tersebut, mendapatkan alamat IP, menyatakan kemampuan, serta mempelajari kehadiran dan kemampuan peralatan lain secara otomatis (Sun, 2006).

Mengacu dari penelitian sebelumnya yang berjudul "Implementasi purwarupa perangkat rumah cerdas pervasif berbasis protokol *Universal Plug and Play* (UPnP) dan Raspberry pi *General Purpose Input/Output* (GPIO)" dalam penelitiannya membuat sebuah alat untuk menyalakan lampu menggunakan kontrol manual dari ponsel atau PC dengan menggunakan metode komunikasi

Universal Plug and Play (UPnP) (Akbar, et al., 2015). Kelemahan dari alat tersebut adalah masih menggunakan kontrol manual dari ponsel atau PC yang tidak jauh berbeda dengan menggunakan sakelar analog. Solusi dari kelemahannya yaitu menggunakan sebuah sensor sebagai kontrol otomatis.

Sensor merupakan perangkat yang memiliki fungsi dapat mengubah suatu besaran fisik yang akan menjadi suatu besaran listrik, kemudian keluaran dari sensor dapat diproses dengan rangkaian elektronik atau dengan menggunakan sistem digital. Sensor sendiri dapat dibedakan menjadi dua jenis yaitu sensor fisika dan sensor kimia. Untuk sensor fisika yaitu membaca suatu besaran yang berdasarkan hukum fisika seperti halnya cahaya, suhu, suara, gaya, kecepatan dan percepatan. Sedangkan sensor kimia yaitu merubah suatu besaran dari zat kimia yang dirubah menjadi besaran listrik (Setiawan, 2009).

Berdasarkan dari pembahasan dan penjelasan yang terdapat dilatar belakang, maka peneliti ingin merancang sebuah sistem yang dapat menyalakan lampu secara otomatis menggunakan protokol *Universal Plug and Play* (UPnP) dimana untuk nilai masukannya sendiri menggunakan sensor dan untuk otaknya memanfaatkan mikrokomputer pada Raspberry Pi. Harapan dari penelitian yang akan dibuat yaitu dapat mengurangi campur tangan manusia untuk mengontrol lampu dan lebih menghemat penggunaan sumber daya listrik.

1.2 Rumusan Masalah

Berdasarkan Pembahasan pada latar belakang, maka penulis membuat rumusan masalah antara lain sebagai berikut:

1. Bagaimana cara pengenalan perangkat *smart home* menggunakan protokol UPnP dalam suatu jaringan?
2. Bagaimana cara mengimplementasikan protokol *Universal Plug and Play* (UPnP) pada sensor dan aktuator untuk otomasi lampu?
3. Bagaimana kinerja perangkat *smart home* dari segi waktu respon saat menggunakan protokol UPnP?
4. Bagaimana kinerja perangkat *control point* saat menerima paket data dari berbagai macam perangkat *smart home* secara bersamaan dari segi waktu respon?

1.3 Tujuan

Dengan adanya rumusan masalah yang telah ada, maka didapat tujuan yang terkait pada rumusan masalah tersebut adalah :

Tujuan umum:

Dapat Menerapkan protokol *Universal Plug and Play* (UPnP) pada *smart home* untuk otomasi lampu.

Tujuan khusus:

1. Dapat berkomunikasi antar perangkat *smart home* dengan *control point* secara otomatis dengan menggunakan fungsi *description* pada UPnP.
2. Dapat mengimplementasikan perangkat *smart home* dengan menggunakan raspberry pi, sensor PIR, LDR, dan relay. selanjutnya dapat mengimplementasikan fungsi yang terdapat pada protokol UPnP.
3. Perangkat *smart home* dapat berkomunikasi secara *realtime* dengan waktu respon antar perangkat *smart home* kurang dari 5 detik.
4. *Control point* dapat menerima banyak data dari berbagai perangkat *smart home* secara *realtime.*

1.4 Manfaat

Bangga dengan diri sendiri saat dapat menciptakan sebuah alat atau sistem yang dapat berguna untuk membantu orang lain. Selanjutnya dapat ikut berkontribusi pada penelitian selanjutnya mengenai topik *realtime* dan *smart home.* Serta dapat bermanfaat untuk perkembangan protokol *Universal Plug and Play* (UPnP) pada kinerja *realtime* dan *smart home* kedepannya.

1.5 Batasan Masalah

Adanya penggunaan dalam batasan masalah disini agar menjadi lebih terfokus dan terarah serta tidak menyimpang dari pembahasan yang terdapat dilatar belakang. Oleh karenanya penulis membuat batasan masalah penelitian ini sebagai berikut :

1. Protokol yang digunakan adalah *Universal Plug and Play* (UPnP).
2. Mikrokomputer yang digunakan adalah Raspberry Pi 3 sebanyak 3 buah.
3. Sensor PIR (*Passive Infra Red*) 1 buah.
4. Sensor LDR (*Light dependent Resistor*) 1 buah.
5. Relay modul 1 *cannel.*
6. Lampu 1 buah.
7. Fitur pada kontrol lampu adalah on dan off.
8. Nilai yang didapat dari sensor cahaya berupa satu dan nol.
9. Jaringan yang digunakan adalah jaringan Wifi lokal (LAN).
10. Mengabaikan keamanan jaringan.

1.6 Sistematika Pembahasan

Sistematika yang digunakan oleh penulis adalah implementasi dan terdiri dari beberapa bab. Guna untuk dapat memahami dengan lebih jelas mengenai

pembahasan proposal, maka dilakukan pengelompokan untuk setiap materi yang mana adalah sebagai berikut :

BAB I PENDAHULUAN

Bab ini berfungsi untuk menjelaskan dari beragam informasi umum dan masalah yang ada untuk menguatkan sebagai dasar pembuatan skripsi. Isi dari bab satu yaitu dari latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan.

BAB II LANDASAN KEPUSTAKAAN

Bab ini berisikan teori-teori yang didapat dari berbagai jurnal, kutipan buku serta dari web yang telah dibuat oleh lembaga resmi yang dapat dipertanggung jawabkan mengenai definisi dan pengertian. Bab ini juga menjelaskan konsep dasar untuk mendukung sistem otomasi lampu menggunakan protokol *Universal Plug and Play* (UPnP), sensor PIR, sensor LDR, dan Raspberry Pi.

BAB III METODOLOGI

Bab ini berisi langkah-langkah yang harus dilakukan dan berisi metode penelitian mengenai sistem otomasi lampu menggunakan protokol *Universal Plug and Play* (UPnP).

BAB IV PERANCANGAN SISTEM

Bab ini membahas mengenai kebutuhan persyaratan dan rekayasa kebutuhan yang diperlukan untuk merancang serta menjelaskan gambaran umum mengenai sistem yang akan dirancang.

BAB V IMPLEMENTASI

Pada bab ini membahas mengenai penerapan sistem yang mengacu dari perancangan pada bab sebumnya.

BAB VI PENGUJIAN DAN ANALISIS

Bab ini membahas tentang rangkaian pelaksanaan untuk melakukan pengujian dan selanjutnya melakukan analisis pada *prototype realtime* menggunakan protokol *Universal Plug and Play* (UPnP).

BAB VII KESIMPULAN

Bab ini berisikan beberapa poin kesimpulan yang didapat dari pembuatan sistem dan pengujian pada perangkat keras serta pengujian pada perangkat lunak yang sudah dikembangkan untuk skripsi ini. Selanjutnya berupa saran untuk memudahkan pengembangan selanjutnya dari pelaksanaan penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tinjauan pustaka yang mencakup mengenai dasar teori yang dibutuhkan untuk penelitian. Pengertian dari suatu tinjauan pustaka sendiri adalah membahas mengenai penelitian yang sebelumnya sudah ada. Untuk dasar teori berfungsi sebagai penunjang untuk menyusun penelitian ini.

2.1 Tinjauan Pustaka

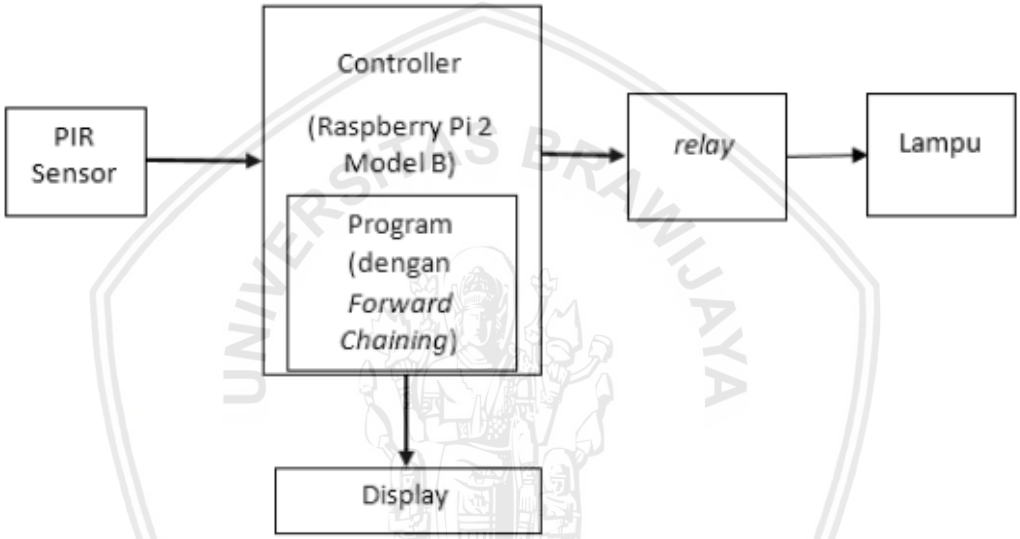
Tinjauan pustaka berfungi untuk membahas mengenai perbandingan dengan penelitian sebelumnya yang masih memiliki keterkaitan dengan penelitian yang penulis buat. Mengenai tujuan dari tinjauan pustaka ialah untuk mengkaji hasil dari penelitian sebelumnya yang nantinya akan dijadikan sebagai dasar dalam melaksanakan penelitian ini. Disini penulis membahas 2 penelitian yang sudah pernah dilakukan sebelumnya, antara lain sebagai berikut.

Tabel 2.1 Tinjauan Pustaka

No.	Nama Penulis [Tahun], Judul	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
1.	Stephanie & Herry Sujaini [2015], Sistem Otomasi Lampu pada Bangunan Publik dengan Metode *Forward Chaining.*	Pembuatan alat otomasi lampu menggunakan Raspberry pi dan menggunakan sensor PIR.	- Metode yang digunakan *Forward chaining.*	- metode yang akan digunakan protokol UPnP.
2.	Sabriansyah Rizqika Akbar, Barlian Henryranu, Maystia Tri Handono, & Achmad Basuki [2015], Implementasi Purwarupa Perangkat Rumah Cerdas Pervasif Berbasis Protokol *Universal Plug and Play* (UPnP) dan Raspberry Pi *General Purpose Input/Output* (GPIO)	Pembuatan alat rumah cerdas menggunakan Raspberry Pi dengan Protokol UPnP	- Alat yang digunakan ponsel & PC sebagai kontrol LED.	- Alat yang akan digunakan adalah sensor PIR dan sensor LDR sebagai nilai masukan.

1. Sistem Otomasi Lampu pada Bangunan Publik dengan Metode *Forward Chaining*

Menurut (Stephanie & sujaini, 2015) di Indonesia penggunaan sumber daya listrik setiap tahunnya meningkat, jika dibandingkan penggunaan listrik untuk tahun 2010 sendiri sebesar 147,3 TWh dengan tingkat rasio elektrifikasi Indonesia yang baru mencapai 66,51% maka rasio tersebut tergolong rendah. Dalam penggunaan atau efisiensi dalam pemakaian energi listrik sangat penting disini, maka dari itu dengan meningkatkan penggunaan listrik dengan efisien maka penulis membuat sebuah alat dimana penggunaan untuk lampu akan secara otomatis mati dan menyala tergantung dari input yang bersal dari sensor *Passive Infrared Receiver* (*PIR*). Untuk metodenya sendiri menggunakan *Forward Chaining.*

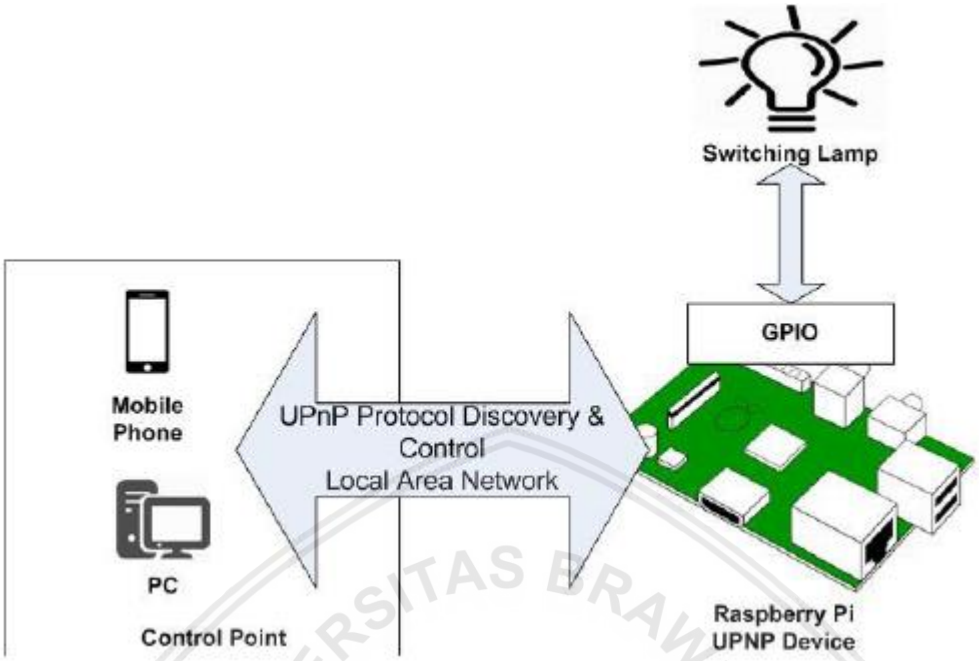


Gambar 2.1 Arsitektur sistem lampu dengan metode *forward chaining*

Sumber : (Stephanie & Sujaini, 2015)

2. Implementasi Purwarupa Perangkat Rumah Cerdas *Pervasif* Berbasis Protokol *Universal Plug and Play* (UPnP) dan Raspberry Pi *General purpose Input/Output* (GPIO)

Menurut (Akbar, et al., 2015) banyak purwarupa sistem peralatan rumah cerdas yang menggunakan arsitektur dan kontrolnya memiliki variasi yang berbeda dengan peralatan rumah cerdas lainnya sehingga akan menyita waktu dalam proses konfigurasi pada jaringan pada setiap peralatan. Maka dari itu penulis menggunakan mikrokomputer Rasberpy Pi sebagai pusat kontrol karena mikrokomputer bekerja secara pervasive dan dapat ditemukan oleh perangkat lain seperti pada komputer atau pada *smartphone.* Dengan menggunakan protokol *Universal Plug and Play* (UPnP) yang diintregasikan dengan *General Purpose Input/Output* (GPIO) maka akan mudah untuk *switching* pada lampu menggunakan aplikasi generik *control point* UPnP *spy.*



Gambar 2.2 Blok Diagram Purwarupa Sistem

Sumber : (Akbar, et al., 2015)

2.2 Dasar Teori

Dasar teori adalah tahap yang membahas adapun teori-teori untuk menunjang pada pembuatan penelitian ini. Dasar teori yang akan dibahas masuk dalam sub bab dimulai dari 2.2.1 sampai 2.2.5.

2.2.1 Raspberry Pi 3

Raspberry Pi merupakan suatu komputer yang memiliki ukuran mini yang mana hanya sebesar kartu kredit. Untuk namanya sendirinya sendiri diambil dari nama buah yang lumayan familiar yaitu Raspberry dan nama dari Pi sendiri diambil dari Phyton bahasa pemrograman. Phyton dijadikan nama dari komputer mini ini. Namun, tidak menutup kemungkinan untuk dapat menggunakan bahasa pemrograman yang lain pada Raspberry Pi. Pada Raspberry Pi mempunyai sebuah komponen pendukung yang hampir serupa dengan komponen yang terdapat pada komputer pada umumnya seperti menggunakan CPU, RAM, USB, Port, HDMI, GPU, Ethernet, GPIO, dan Audio Jack. Untuk penyimpanan pada Raspberry Pi tidak menggunakan sebuah *Harddick drive* (HDD) tetapi menggunakan sebuah *Micro SD* yang memiliki besar kapasitas tidak kurang dari 4GB. Untuk menyalakan raspberry pi membutuhkan sumber daya listrik sebesar 5V yang diambil dari adaptor atau *micro USB power* dan arusnya minimal 700mA (Sayuti, 2015).

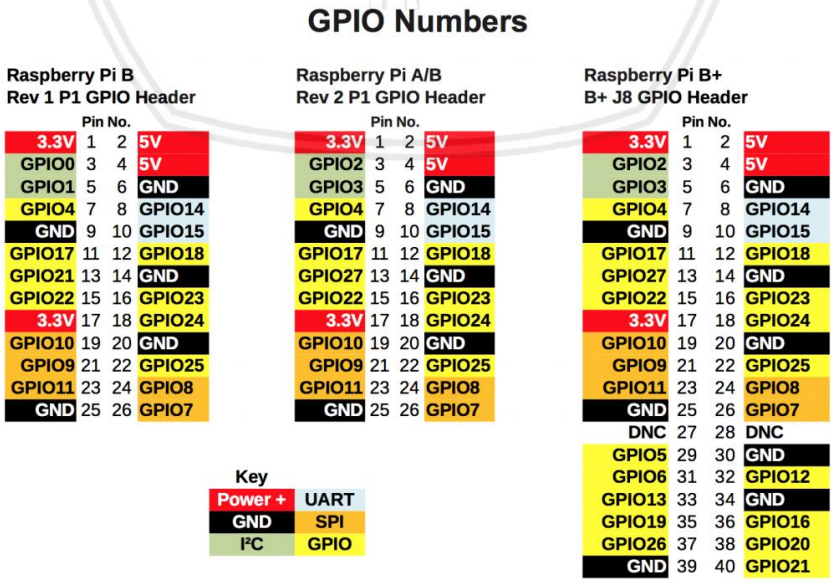
Gambar 2.3 Raspberry Pi

Sumber : (Raspberry Pi, 2017)

Tabel 2.2 Spesifikasi Raspberry Pi 3

Item	Parameter
SoC	Broadcom BCM2837
CPU	4x ARM Cortex-A53, 1.2GHz
GPU	Broardcom VideoCore IV
RAM	1GB LPDDR2 (900 MHz)
Network	10/100 Ethernet, 2.4 GHz 802.11n wireless
Bluetooth	Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage	MicroSD
GPIO	40-pin header, populated
Ports	Ethernet, 3.5mm analogue audio-video jack, HDMI, 4x USB 2.0, Display Serial Interface (DSI), Camera Serial Interface (CSI)

Sumber : (Raspberry Pi, 2017)



Gambar 2.4 Pin GPIO pada Raspberry Pi

Sumber : (Bachrudin, et al., 2017)

Pada Gambar 2.4 merupakan gambaran dan penamaan dari setiap pin yang terdapat di mikrokomputer Raspberry pi. Fungsi dari pin tersebut untuk membaca *input* dan mengontrol *output* berdasarkan kondisi yang berbeda-beda tergantung dari program yang dibuat pada Raspberry pi. Sebagaian dari GPIO digunakan untuk *input/output* digital dan sebagai antarmuka sebuah protokol. Untuk fungsi GPIO sebagai *output* yaitu dapat mengendalikan *output* dari perangkat keras sepertinya lampu, motor, dan relay. Fungsi *input* yaitu membaca status atau nilai dari tombol, *switch,* dan sensor. Raspberry pi memiliki 40 pin yang terdiri dari 2 pin sumber tegangan 5V, 2 pin sumber tegangan 3,3V, 8 pin untuk *ground,* 2 pin serial dan 26 pin untuk GPIO (Bachrudin, et al., 2017).

2.2.1.1 Library WiringPi

WiringPi merupakan sebuah *library* yang berfungsi untuk mengakses pada setiap pin GPIO menggunakan bahasa C yang terdapat pada BCM2835, BCM2836, dan perangkat SoC BCM2837 dan digunakan pada semua Raspberry pi. Untuk versi terbaru WiringPi dapat menggunakan bahasa C, C++, RTB dan bahasa pemrograman yang lainya. WiringPi menggunakan lisensi GNU *LGPLv3* yang dikembangkan langsung oleh Raspbian 32–bit. Fitur pada *library* WiringPi antara lain sebagai berikut (Henderson, 2009):

1. Mengakses pin GPIO pada Raspberry pi.
2. Dapat mengatur nilai dari PWM.
3. Terdapat sebuah *Serial library.*
4. Terdapat sebuah I2C *library.*
5. Terdapat sebuah SPI *library.*
6. Terdapat sebuah *shift library.*

2.2.2 Sensor PIR (*Passive Infra Red*)

Sensor PIR merupakan sensor yang memiliki kinerja bersifat pasif dimana fungsi sensor PIR hanya dapat menerima masukan dari pancaran sinar infra merah yang berasal dari luar tanpa dapat memancarkan sinar infra merah dari sensor tersebut. Sensor PIR ini bekerja dengan menangkap sumber inframerah dengan cara membandingkan antara objek 1 dengan objek yang lain, seperti halnya manusia dengan dinding dimana sensor ini membandingkan pancaran infra merah dari manusia dan dinding. Jika terdapat sebuah pergerakan yang tertangkap maka ada perubahan juga pada pembacaan dari sensor. Fungsi lainnya adalah mendeteksi perubahan atau perbedaan suhu antara nilai sekarang dengan nilai sebelumnya.

Sensor PIR (Passive Infra Red) bekerja dengan sangat sederhana dan mudah untuk diaplikasikan pada perangkat cerdas karena sensor ini hanya memerlukan sumber tegangan input sebesar DC 5V dan angkat tersebut cukup efektif dimana jarak jangkauan sensor ini hingga 5 meter. Dalam penggunaannya Sensor PIR pada kondisi saat tidak mendeteksi sebuah pergerakan maka untuk keluaran nilai dari

modul yaitu LOW sedangan ketika sensor PIR mendeteksi adanya sebuah pergerakan maka keluaran pada modul berubah menjadi HIGH. Untuk lebar dari pulsa pada kondisi HIGH yaitu sebesar kurang lebih 0,5 detik. Untuk sensitifitas dari sensor PIR (Passive Infra Red) yang dapat mendeteksi sampai jarak sebesar 5 meter yang mana sangat memungkinkan untuk dapat membuat suatu alat pendeteksi pergerakan dengan presentasi dari keberhasilannya lebih tinggi. Gambar dari sensor PIR adalah seperti yang ditunjukkan pada Gambar 2.5.

Gambar 2.5 Sensor PIR

Sumber : (PIR, 2018)

Bagian-bagian dari Sensor PIR yaitu penyaring infra merah, lensa Fresnel, sensor Pyroelektrik, komparator, dan penguat amplifier. Untuk kerjanya sendiri adalah menerima pancaran dari infra merah yang masuk melewati lensa Fresnel dan pada akhirnya mengenai pada bagian pyroelektrik, dikarenakan sinar infra merah menghasilkan panas dengan itu sensor pyroelektrik menghasilkan suatu besaran pada arus listrik. Pada arus listrik ini yang nanti akan menimbulkan tegangan serta dapat dibaca secara analog pada sensor. Selanjutnya untuk komponen yang berfungsi sebagai sinyal tersebut akan dikuatkan oleh komponen penguat dan dibandingkan pada komparator dengan nilai tegangan referensi dengan nilai tertentu (keluaran dalam bentuk 1-bit). terakhir sensor PIR (*Passive Infra Red*) sendiri hanya asapat memberikan keluaran dengan nilai logika 0 dan 1 saja.

Tabel 2.3 Spesifikasi Sensor PIR

Item	Parameter
Model	HC-SR501
Operating range	4.5-20V DC
Current	50uA
Level output	High 3.3 V / Low 0V
Trigger	L unrepeatable trigger/H repeatable trigger

Item	Parameter
Delay time	0.5 - 200S
Blocking time	2.5S
PCB dimensions	32 mm x 24 mm
Angle sensors	100 degree cone angle
Operating temperature	-15 -+ 70 Degrees
Sensor lens size	Diameter 23 mm
Range	6 meter, 110° x 70°

Sumber: (Putra, 2016)

Desain sensor PIR berupa pendeteksi pancaran infra merah pada panjang gelombang diantara 8 sampai 14 mikrometer. Untuk nilai gelombang yang diluar gelombang tersebut, maka sensor tidak dapat membacanya. Pada tubuh manusia memiliki gelombang 9 sampai 10 mikrometer (standar nilai 9,4 mikrometer), untuk nilai dari panjang gelombang tersebut mudah untuk dideteksi oleh sensor PIR dan pada umumnya sensor PIR memang dirancang dan dibuat untuk dapat mendeteksi keberadaan manusia (Putra, 2016).

2.2.3 LDR (*Light Dependent Resistor*)

Modul LDR merupakan resistor yang memiliki perubahan pada nilai resistansi yang berdasarkan pada intesitas cahaya. Elemen yang terkandung pada LDR antara lain kadmium sulfida (CdS) yang fungsinya dapat menangkap kepekaan terhadap cahaya. Berikut ini cara kerja dari LDR, apabila dalam keadaan cahaya redup maka kondisi dari bahan piringan elektron bebas yang terkandung dalam jumlah yang sangat kecil itu hanya menghasilkan arus aliran listrik yang sedikit.

Dapat dikatakan bahwa bahan piringan kurang cocok untuk menghantarkan arus listrik karena termasuk bahan konduktor yang buruk. Sedangkan apabila dalam keadaan cahaya terang maka dapat menghasilkan elektron bebas yang cocok untuk mengalirkan aliran muatan listrik sehingga termasuk bahan konduktor yang baik (Lindiawati, et al., 2013).

Gambar 2.6 Sensor LDR

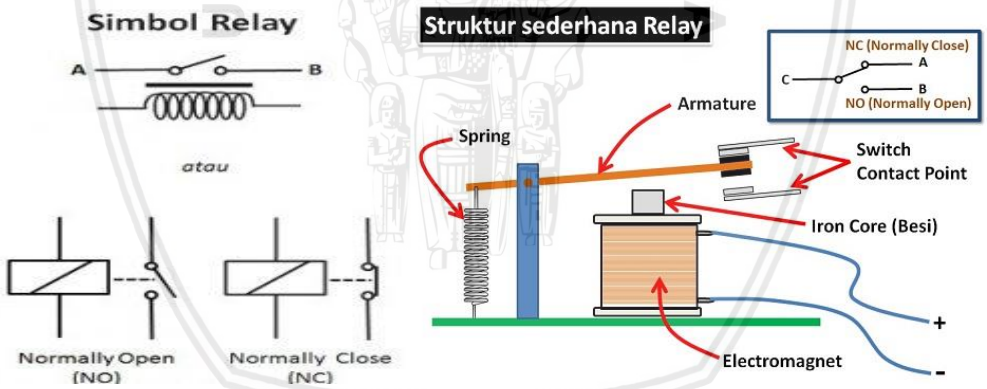
Sumber : (Lindiawati, et al., 2013)

2.2.4 *Relay Module*

Relay adalah sebuah sakelar (*switch)* yang mana cara pengoperasiannya menggunakan listrik dan membentuk sebuah komponen yang bernama elektromekanikal yang memiliki dua bagian utama yaitu yang dinamakan elektromagnet (*coil)* dan sebuah mekanikal (kontak sakelar/*switch*). Untuk penejlasan dari prinsip yang terdapat di *relay* adalah elektromagnetik yang mana dapat menggerakkan kontak sakelar yang menghasilkan arus listrik yang memiliki tegangan kecil (*low power*) untuk bisa menghantarkan sebuah arus listrik dimanamempunyai tegangan yang lebih tinggi (Saleh & Haryanti, 2017). Untuk tampilan dari *relay* 1 *channel* dapat dilihat pada Gambar 2.7.

Gambar 2.7 *Relay Module*

Sumber : (Relay, 2019)



Gambar 2.8 Simbol dan Struktur Sederhana *Relay*

Sumber : (Saleh & Haryanti, 2017)

Pada Gambar 2.8 menunjukan simbol dan struktur sederhana yang dimiliki oleh *relay.* Di dalam *relay* sendiri terdapat empat komponen mendasar yaitu *armature*, elektromagnet *(coil)*, *switch contact point* (sakelar), dan *spring.* Pada bagian *contact point* memiliki dua jenis antara lain *normally close* (NC) dan *normally open* (NO). Pengertian dari *normally close* (NC) adalah dimana terdapat kondisi awal saat belum diaktifkan akan selalu dalam kondisi posisi tertutup *(close).* Sedangkan *normally open* (NO) adalah suatu kondisi awal saat sebelum diaktifkan maka posisinya akan terbuka *(open).*

Sehubungan dengan *relay* dimana dapat dikategorikan dalam sebuah sakelar, untuk itu terdapat yang namanya *pole* dan *throw* digunakan dalam sakelar yang terdapat dalam *relay* juga. Pengertian dari *pole* adalah jumlah dari kontak

(contact) yang terdapat pada modul *relay.* kalau untuk *throw* yaitu jumlah keadaan yang dapat dimiliki pada sebuah kontak *(contact)* (Saleh & Haryanti, 2017).

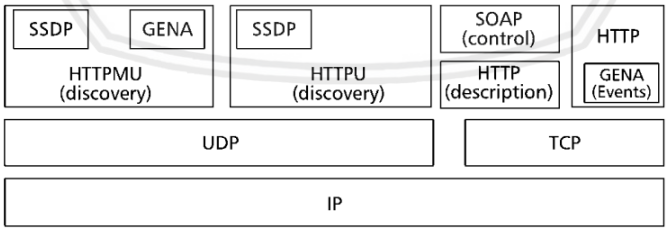
Tabel 2.4 Spesifikasi *Relay*

Item	Parameter
Supply Voltage	3.75 to 6V
Supply Current with Relay De-Energized	2mA
Supply Current with Relay Energized	70 to 72mA
Input Control Signal	Active Low
Input Control Signal Current	1.5 to 1.9mA
Relay Max Contact Voltage	250VAC or 30VCD
Relay Max Contact Current	10A

Sumber : (Relay, 2019)

2.2.5 UPnP (*Universal Plug and Play*)

UPnP adalah suatu arsitektur yang didesain untuk menghubungkan perangkat-perangkat yang terhubung dengan suatu jaringan dan juga dapat mendefinisikan standar baku pada semua perangkat untuk dapat mendeskripsikan spesifikasi serta layanan yang diberikan. Standar yang digunakan pada UPnP adalah standar yang sekarang sudah ada seperti halnya TCP, UDP, XML, HTTP dan IP. Seperti pada umumnya dalam sebuah internet yang menggunakan XML, UPnP sendiri juga menggunakan XML dan berkomunikasi menggunakan HTTP. UPnP juga dapat berjalan pada hampir semua operasi sistem yang sekarang sudah ada (Presser, et al., 2015). Untuk arsitektur UPnP sendiri dapat dilihat pada Gambar 2.9.



Gambar 2.9 Arsitektur UPnP

Sumber : (Presser, et al., 2015)

Fitur-fitur yang terdapat pada UPnP sebagai berikut:

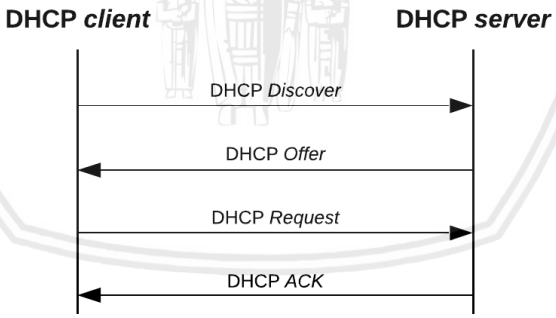
1. Device Connectivity : berfungsi mengatur protokol pada setiap perangkat untuk berkomunikasi dengan perangkat lain.
2. Ad-Hoc Networking : berfungsi menggabungkan perangkat UPnP secara dinamik tanpa membutuhkan konfigurasi manual.

3. Zero-Configuration Network : berfungsi untuk pengguna agar tidak mengatur secara manual pada konfigurasi jaringan di perangkat tersebut.
4. Standar Base Arsitecture : membuat arsitektur UPnP didasarkan pada open standar.
5. Platform Independence : set pada arsitektur UPnP adalah protokol bukan menggunakan API.
6. Media and Device Independence : berfungsi untuk membuat UPnP dapat berjalan dimanapun selama terdapat sebuah IP.
7. Programmatic and Manual Device Control : berfungsi untuk memberikan suatu izin pada aplikasi yang sudah terprogram untuk mengatur perangkat yang terdapat pada home network (Presser, et al., 2015).

Tahapan dalam UPnP antara lain *sebagai berikut :*

a. Addressing

Addressing sebagai fungsi awal untuk menghubungkan perangkat yang dapat terkoneksi dengan IP jaringan (WLAN) pada *access point.* Proses *addressing* juga menyimpan informasi SSID dan *Password* dari *access point* kedalam memori dan berguna untuk terkoneksi secara otomatis pada siklus berikutnya. Penggunaan DHCP pada tahap *addressing* untuk menjalankan layanan pada server. Dapat dilihat pada Gambar 2.10 DHCP *client* dapat mengirimkan permintaan kepada server dan DHCP server dapat mengalokasikan alamat IPv4 secara dinamis kepada *client* (Sismoro, 2001).



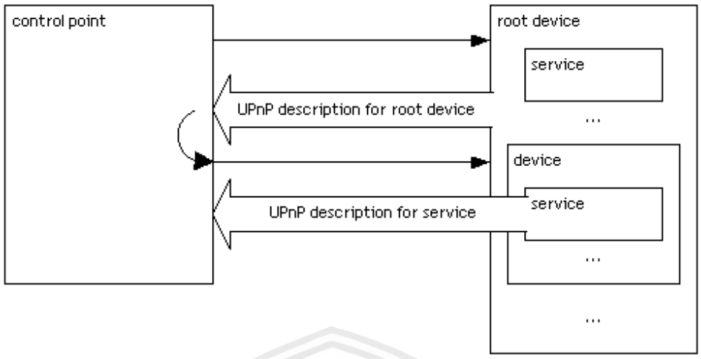
Gambar 2.10 Ilustrasi Diagram DHCP *Client* dan *Server*

Sumber : (Sismoro, 2001)

b. Discovery

Discovery berfungsi menerima serta mengidentifikasikan permintaan yang dikirim oleh *control point* berupa UPnP *discovery* dan dapat memberikan pelayanan berupa pesan yang dikirim kembali pada *control point* berupa berkas bertipe XML yang berisikan lokasi dengan format yang sedang berlaku. Pada Gambar 2.11 menjelaskan bahwa *advertise* dikirim secara *multicase* ke dalam jaringan dengan menggunakan UDP. Permintaan pada UPnP menggunakan *discovery* M-SEARCH yang akan dikirimkan dari *control point* menggunakan HTTP

dengan cara *multicast* menggunakan UDP (HTTPMU). Sedangkan untuk balasan dari pesan sebelumnya menggunakan UDP (HTTPU) dengan cara *multicast* juga.

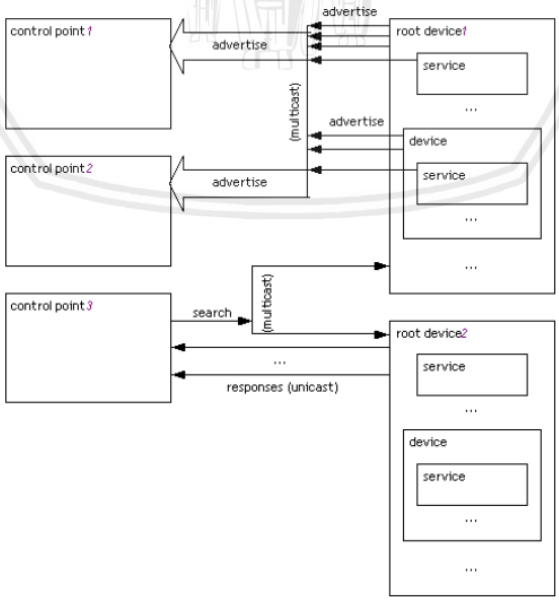


Gambar 2.11 Diagram *Discovery* pada UPnP

Sumber : (Presser, et al., 2015)

b. Description

Description berfungsi untuk menerima serta mengidentifikasikan permintaan dari UPnP *description* yang dikirim *control point* dan dapat melayani dengan membalas berupa berkas XML yang berisikan deskripsi dari perangkat dan layanan yang tersedia dengan format yang sedang berlaku. Pada Gambar 2.12 menjelaskan bahwa *description* dibedakan dalam dua jenis yaitu deskripsi perangkat dan deskripsi layanan. Deskripsi perangkat antara lain memiliki informasi perangkat dari nama perangkat, tipe, model, produsen, dan nomor seri. Dalam deskripsi layanan berisikan mengenai nama layanan, URL dan tipe untuk permintaan terhadap deskripsi layanan, *control,* maupun *eventing*.



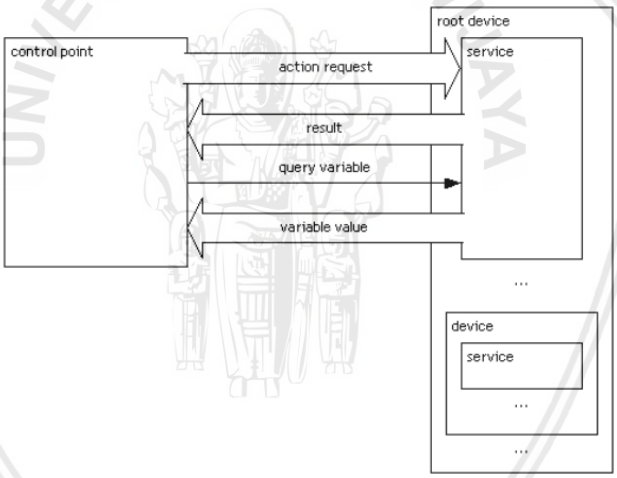
Gambar 2.12 Diagram *Description* pada UPnP

Sumber : (Presser, et al., 2015)

Cara kerjanya untuk mendapatkaninformasi mengenai deskripsi perangkat yaitu dengan cara melakukan dari pihak *control point* mengirimkan sebuah permintaan HTTP GET kepada URL deskripsi perangkat yang sebelumnya didapatkan pada tahap *discovery.* Kalau pada permintaan deskripsi layanan dengan mengirimkan sebuah permintaan HTTP GET kepada URL deskripsi layanan yang telah diperoleh sebelumnya pada deksripsi perangkat.

c. Control

Control berfungsi untuk mengidentifikasikan permintaan dari UPnP *control* dikirim dari *control point* dan selanjutnya melayaninya dengan menjalakan *method* yang diinstruksikan dan membalas berupa pesan yang berisikan hasil perolehan dari instruksi *method* yang menggunakan format saat ini. Pada Gambar 2.13 *control point* dapat melakukan *remote procedure call* dimana fungsi tersebut adalah dengan menjalankan *method* tertentu telah disediakan dari layanan. *Control point* bekerja dengan cara melakukan *control* dengan mengirimkan sebuah permintaan yang bernama HTTP POST kepada URL *control d*ari layanan tersebut. Pesan permintaan berformat XML dengan berdasar dari protokol *Simple Object Access Protocol* (SOAP).

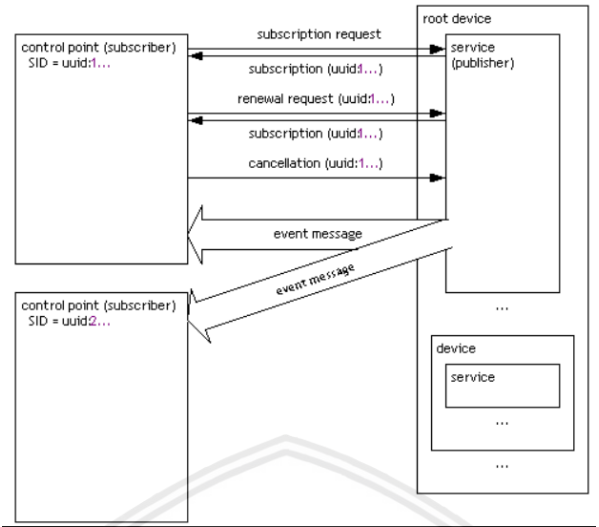


Gambar 2.13 Diagram *Control* pada UPnP

Sumber : (Presser, et al., 2015)

d. Eventing

Eventing berfungsi untuk menyimpan *device description* untuk digunakan sebagai langganan agar lebih mudah mendapatkan status dari kondisi *device* tersebut. Cara kerjanya dengan mengirimkan paket *eventing* kepada *device* yang telah dijadikan langganan kemudian *device* akan membalas dengan status dari kondisi saat itu juga.



Gambar 2.14 Diagram *Eventing* pada UPnP

Sumber : (Presser, et al., 2015)

Untuk *eventing* dapat dilihat pada Gambar 2.14 yang menjelaskan bahwa untuk berlangganan dari suatu perangkat dengan melakukan pesan permintaan berlangganan kepada URL langganan dari layanan serta memberikan layanan berupa menyediakan sebuah permintaan untuk *event message.* Saat pesan permintaan diterima, selanjutnya perangkat dapat memberikan infomasi berupa identitas untuk pengenalan pelanggan serta *subscription ID,* SID, dan masa aktif untuk berlangganan. Setelah berlangganan perangkat dapat mengirimkan pesan berupa *event* yang berisikan *nama dari* state variabel dan nilai terbaru. Pesan yang berisikan *event* menggunakan protokol *General Event Notification Architecture* (GENA) dan juga HTTP (Presser, et al., 2015).

2.2.5.1 Library GUPnP

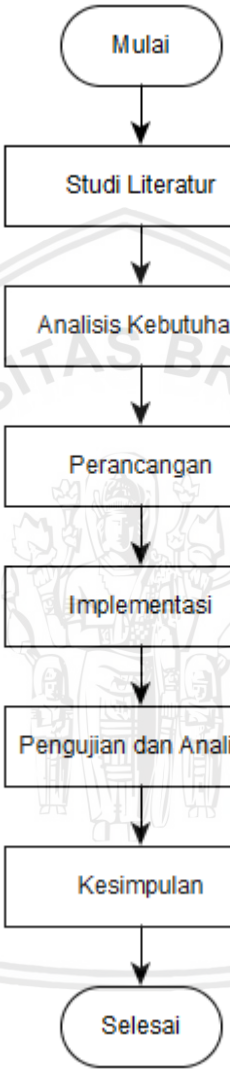
*Library G*UPnP adalah sebuah program dasar yang sudah jadi dimana berfungsi untuk menunjang pembuatan program UPnP *device* lebih cepat dan mudah. *Library G*UPnP ini bersifat *open source* yang membuatnya bisa diakses oleh siapa saja. Pada *library* GUPnP menggunakan bahasa pemrograman yaitu bahasa *C.* Penambahan yang dapat mempermudah penggunaan *library G*UPnP yaitu terdapat *libsoup* dan *Gobject.* Berikut adalah beberapa fitur utama dari *library G*UPnP (Georg, et al., 2018).

1. Implementasi GSSDP yang berfungsi sebagai *resource announcement* dan *discovery* menggunakan *SSDP.*
2. *G*UPnP memiliki fitur yang terdapat pada UPnP yaitu sebagai *resource announcement* dan *discovery, description, control,* dan *event notification.*
3. GUPnP-DLNA adalah *library* untuk membangun aplikasi berbasis *AV* (*audio / video*).

4. *G*UPnP-IGD adalah sebuah *library* yang bertujuan mempermudah fungsi-fungsi dari DLNA. Fiturnya seperti menyalakan meida, *transcoding* ke dalam media, dan sebagainya.
5. *G*UPnP-*Tools* adalah seperangkat utilitas dan demo yang bekerja dengan UPnP. fitur tersebut berupa pengontrol *AV,* jaringan lampu, unggah berkas, *generic control point,* dan *SSDP commandline client.*

BAB 3 METODOLOGI

Bab ini menjelasan mengenai metode apa saja yang akan digunakan untuk melakukan penelitian serta akan menjabarkan tujuan pada setiap langkah pada penelitian ini. Berikut adalah tahap dalam melakukan penelitian ini dalam bentuk flowchart.



Gambar 3.1 Flowchart Metodologi Penelitian

3.1 Metodologi Penelitian

Pada metodologi penelitian akan menjabarkan setiap langkah pada penelitian. Dalam hal ini langkah yang akan dibahas secara mendalam antara lain meliputi studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis, kesimpulan an saran. Untuk penelitiannya sendiri tentang implementasi protokol *Universal Plug and Play* pada sensor dan aktuator untuk otomasi lampu. Untuk tipe dari penelitian ini yaitu berupa implementatif yang mana memiliki sifat observasi dengan menggunakan sebuah rancang bangun

sederhana. Berikut adalah bentuk diagram alir untuk metodologi penelitian yang ditunjukkan pada gambar 3.1

3.1.1 Studi Literatur

Pada bagian studi literatur dilakukan pengumpulan data dan mempelajari literatur yang memiliki relevan untuk mendukung penelitian ini. Untuk sumber yang diperoleh berasal dari jurnal, buku, *e-book*, skripsi, disetasi, thesis, Penelitian sebelumnya dan lain-lain dimana isi dari studi literatur tersebut membahas tentang :

- *Smart home*
- Raspberry Pi
- Sensor PIR (*Passive Infra Red*)
- LDR (*Light Dependent Resistor*)
- *Relay Module*
- *Universal Plug and Play* (UPnP)

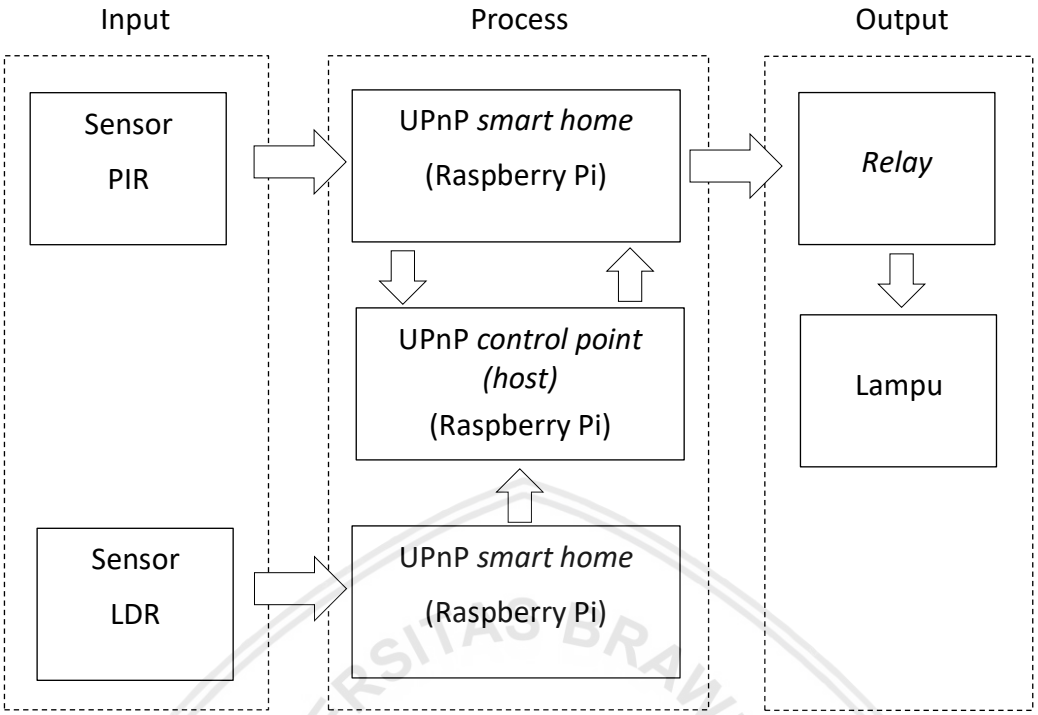
3.1.2 Analisis Kebutuhan

Pada bagian ini menjelaskan mengenai analisis kebutuhan yaitu untuk menganalisis pada kebutuhan yang akan dirancang dan diuji. Kemudian analisis kebutuhan dilakukan berupa mengidentifikasikan kebutuhan sistem apa saja yang diperlukan. Dengan adanya pengidentifikasian kebutuhan maka akan mempermudah dalam merancang hingga membuat sistem tersebut. Kebutuhan tersebut antara lain :

- Raspberry Pi
- Sensor PIR (*Passive Infra Red*)
- LDR (*Light Dependent Resistor*)
- Stop kontak
- Kabel Jumper
- *Relay 1 channel*
- Lampu

3.1.3 Perancangan Sistem

Untuk tahap selanjutnya dari analisis kebutuhan yaitu perancangan sistem. Pada bagian perancangan sistem berguna mempermudah pemahaman mengenai bagaimana sistem akan berjalan. Untuk keseluruhan sistem akan dijelaskan menggunakan diagram blok yang dapat dilihat pada Gambar 3.2. Dari setiap diagram blok mewakili fungsi tertentu yang jika disatukan akan membuat sebuah sistem secara utuh.



Gambar 3.2 Diagram Blok Sistem

Untuk perancangan perangkat lunak yaitu dengan menerapkan prinsip kerja yang telah disediakan pada masing-masing perangkat berupa *device* dan *control point*. Urutan dari fungsi yang terdapat pada protokol UPnP yaitu dari *addressing, discovery, description,* dan *control.* untuk metode pengenalan perangkat keras menggunakan jaringan lokal (WLAN) pada *access point* dengan metode DHCP.

3.1.4 Implementasi Sistem

Tahap implementasi dibagi menjadi dua yaitu implementasi perangkat keras dan perangkat lunak. Untuk perangkat lunak ialah melakukan penerapan pada alat langsung yang sebelunya telah dirancang pada perancangan sistem. Pada bagian implementasi akan dibagi menjadi beberapa proses antara lain ialah implementasi protokol *Universal Plug and Play* (UPnP) *device, implementasi* protokol *Universal Plug and Play* (UPnP) *service* dan implementasi modul program untuk otomasi lampu. Untuk implementasi UPnP *device* menggunakan pustaka GUPnP dan pemrograman bahasa C. Untuk implementasi UPnP *device description* dan *service description* menggunakan bahasa *XML* yang telah digunakan sebagai standar pembuatan dari UPnP forum.

Pada implementasi perangkat keras yaitu menghubungkan komponen-komponen perangkat keras berdasarkan rancangan yang telah dibuat pada tahap perancangan perangkat keras. Implementasi sendiri berupa menghubungkan sensor kepada Raspberry pi dengan menggunakan jumper dan menancapkan relay dan lampu sebagai *output* dari sistem otomasi Lampu dengan protokol UPnP.

3.1.5 Pengujian dan Analisis Hasil Pengujian

Pada tahap pengujian skripsi dilakukan berbagai skenarion agar dapat menunjukkan apakah sistem telah bekerja dengan semestinya seperti dalam spesifikasi kebutuhan. Untuk itu ada beberapa tahap dalam pengujian yang dilakukan yaitu antara lain ialah :

1. Pengujian terhadap fungsi *addressing* pada UPnP
2. Pengujian terhadap fungsi *discovery* antara *device* dan *control point.*
3. Pengujian terhadap fungsi kontrol yang terdapat pada *control point.*
4. Pengujian waktu respon *device* setelah terhubung untuk pertama kali dan selanjutnya terhadap *control point.*

Data yang didapat dari pengujian akan dianalisis untuk mengetahui kinerja dari penerapan sistem otomasi lampu menggunakan protokol UPnP dimana data yang didapat akan memberikan hasil apakah sistem ini bekerja dengan baik dan layak.

3.1.6 Kesimpulan

Kesimpulan adalah hasil akhir dari tahap implementasi sistem protokol UPnP pada raspberry pi untuk otomasi lampu. Menarik kesimpulan ini diambil dari hasil pengujian fungsional berupa fungsi yang terdapat pada protokol UPnP yaitu *addressing, discovery, description,* dan *control* serta pengujian non fungsional berupa waktu respon dengan beberapa skenario pengujian. Selanjutnya analisis terhadap alat dan sistem yang telah dirancang. Untuk bagian paling akhir adalah saran dari penulis yang bertujuan agar dapat dikembangkannya sistem ini dan memperbaiki kesalahan yang telah dilakukan penulis dalam membuat sistem agar dapat lebih disempurnakan.

BAB 4 REKAYASA KEBUTUHAN

4.1 Gambaran Umum Sistem

Sistem otomasi lampu menggunakan sensor dan aktuator menggunakan protokol UPnP merupakan suatu sistem dimana masukan pada sensor adalah cahaya matahari dan inframerah yang dipancarkan oleh manusia. Saat sensor LDR menangkap sinar matahari maka keluaran dari sensor adalah mati pada lampu, sedangkan saat cahaya matahari sudah menghilang atau sensor sudah tidak menangkap cahaya matahari dengan sempurna maka nilai lampu akan menyala.

Sensor selanjutnya adalah PIR dimana saat sensor menangkap pancaran inframerah dari manusia maka lampu akan menyala dan saat sensor LDR tidak menangkap pancaran inframerah pada manusia dengan kurun waktu tertentu maka lampu akan diberi masukan 0 atau mati.

Pada sistem ini terdapat 2 *device* dan 1 *control point* dimana *control point* dapat berperan sebagai *control point.* Penggunaan protokol UPnP sendiri bertujuan untuk mempermudah pengaturan komunikasi antara *device* dan *control point.* Untuk secara keseluruhan protokol UPnP dapat secara otomatis berkomunikasi dengan perangkat lain yang memiliki fitur UPnP dan juga akan memperluas fungsi *smart home* sendiri pada fungsional rumah.

4.2 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem adalah tahapan untuk mendeskripsikan dan menguraikan mengenai semua kebutuhan yang diperlukan untuk sistem implementasi protokol *universal plug and play* (UPnP) pada sensor dan aktuator untuk otomasi lampu. Analisis Kebutuhan Sistem juga dijadikan dasar utama dalam langkah untuk mengembangkan sistem.

Untuk melakukan analisis kebutuhan sistem terdiri dari dua hal pokok yang harus dijabarkan yaitu analisis kebutuhan fungsional dan analisis kebutuhan nonfungsional. Analisis kebutuhan fungsional memiliki tujuan untuk cara mendapatkan informasi sistem sehingga dalam kebutuhan sistem berisi proses-proses apa saja yang dilakukan suatu sistem. Sedangkan analisis kebutuhan nonfungsional terdiri atas properti kebutuhan suatu sistem dari perangkat keras dan perangkat lunak.

4.2.1 Kebutuhan Fungsional

Berikut ini adalah kebutuhan fungsional yang harus bisa dilakukan oleh sistem:

1. Protokol UPnP (*Universal Plug and Play*)

Protokol UPnP adalah fungsi utama dari perancangan sistem sebagai komunikasi yang diimplementasikan pada perangkat Raspberry pi. Kebutuhan fungsional dari protokol UPnP antara lain sebagai berikut :

a. *Addressing* sebagai fungsi awal untuk menghubungkan perangkat yang dapat terkoneksi dengan IP jaringan (WLAN) pada *access point.* Proses *addressing* juga menyimpan informasi SSID dan *Password* dari *access point* kedalam memori dan berguna untuk terkoneksi secara otomatis pada siklus berikutnya.

b. *Description* berfungsi untuk menerima serta mengidentifikasikan permintaan dari UPnP *description* yang dikirim oleh *control point* dan melayani dengan membalas berupa berkas XML yang berisikan deskripsi dari perangkat dan layanan yang tersedia dengan format yang sedang berlaku.

c. *Control* berfungsi untuk mengidentifikasikan permintaan dari UPnP *control* yang dikirim oleh *control point* dan melayani dengan menjalakan *method* yang diinstruksikan dan membalas berupa pesan yang berisikan hasil yang diperoleh dari instruksi *method* dengan format yang sedang berlaku.

4.2.2 Kebutuhan Nonfungsional

Kebutuhan non fungsional dari sistem terdiri dari kebutuhan kinerja, kebutuhan perangkat keras dan kebutuhan perangkat lunak yang dijelaskan sebagai berikut.

a. Kebutuhan Kinerja

Kebutuhan kinerja berfungsi untuk menjelaskan batas waktu respon dari setiap *device* kepada *control point* yaitu maksimal sebesar 5 detik pada setiap proses pengiriman data untuk pertama kali atau setelah pengenalan pertama kali dari *device* ke *control point.* Penggunaan batas maksimal sebesar 5 detik karena mengacu pada standar secara umum untuk waktu respon pada perangkat *realtime.*

b. Kebutuhan Perangkat Keras

Kebutuhan untuk mendukung implementasi dalam pembuatan sistem dari segi perangkat keras maka dibutuhkan beberapa alat yang dijelaskan sebagai berikut:

1. Mikrokontroler Raspberry pi Seri 3

   Mikrokontroler Raspberry Pi seri 3 sebagai alat utama untuk pengolahan informasi dalam perancangan sistem.

2. Sensor PIR (*Passive Infra Red*)

   Sensor PIR dapat menerima pancaran sinar inframerah dari luar. Sensor PIR pengambilan informasi dari pergerakan manusia dalam keadaan memasuki jarak penangkapan oleh sensor.

3. Sensor LDR (*Light Dependent Resistor*)

Sensor LDR dibutuhkan untuk menangkap kepekaan terhadap cahaya. Sehingga sensor dapat memberikan *output* pada lampu sesuai dengan penangkapan cahaya yang didapat.

4. *Relay*

   Relay dapat menjalakan fungsinya sebagai *switch* (sakelar) untuk menyambung atau memutuskan tegangan AC ke lampu.

5. Kabel Jumper

   Kabel jumper dibutuhkan untuk menghubungkan antara komponen-komponen yang digunakan pada sistem yang dibuat.

6. Laptop

   Laptop dalam kebutuhan berfungsi sebagai alat media pendukung untuk membantu membuat program untuk mikrokontroler, dan juga berfungsi sumber daya perangkat keras dari sistem yang akan dibuat. Adapun spesifikasi dari laptop yang digunakan yaitu sebagai berikut :

   - Model perangkat : Asus A455L win 10
   - Prosesor : Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz 2.19 GHz
   - Sistem Operasi : Window 10 Pro 64-bit Operating System, x64-based processor

c. Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang dibutuhkan oleh sistem yang dibuat yaitu seperti berikut ini:

1. Raspbian

   Raspbian merupakan sistem operasi yang berbasis *Linux* atau *Debian GNU* yang dioptimalkan dan dikhususkan untuk mikrokontroler Raspberry pi. Raspbian mencakup seluruh kerja untuk menjalankan program yang dibutuhkan dalam pembuatan program untuk penelitian sistem yang akan dibuat.

4.3 Batasan Desain Sistem

Dalam pengaplikasian sistem otomasi lampu menggunakan sensor dan aktuator dengan protokol UPnP maka terdapat batasan agar pembuatan sistem ini tidak keluar dari aspek lingkup pembahasan, Perancangan dan pengimplementasian. Batasan-batasan tersebut adalah sebagai berikut :

1. Sistem hanya berkerja sebagai pengirim notifikasi dan otomasi lampu dimana terdapat 2 *device* dan 1 *control point*.
2. Perangkat yang digunakan untuk sensor dan aktuator adalah sensor PIR dan sensor LDR, aktuator sendiri berupa lampu.

3. Perangkat hanya bisa terkoneksi dengan wifi 802.11b/g/n yang sudah dikenali sebelumnya yang berasal dari *hostpot* laptop peneliti.
4. Untuk tahap pengenalan perangkat, layanan, dan kontrol yang dimiliki berdasarkan protokol UPnP hanya diterapkan pada jaringan lokal (*WLAN*).
5. Tahapan yang diterapkan pada protokol UPnP antara lain berupa *addressing, discovery, description,* dan *control.*
6. Mengabaikan faktor keamanan jaringan pada (*WLAN*).
7. Mengabaikan faktor penggunaan daya yang digunakan pada setiap *device* dan *control point.*
8. Pengambilan data dari sensor LDR hanya berupa ada cahaya dan tidak ada cahaya atau bisa dibilang bernilai satu dan nol tidak secara analog.

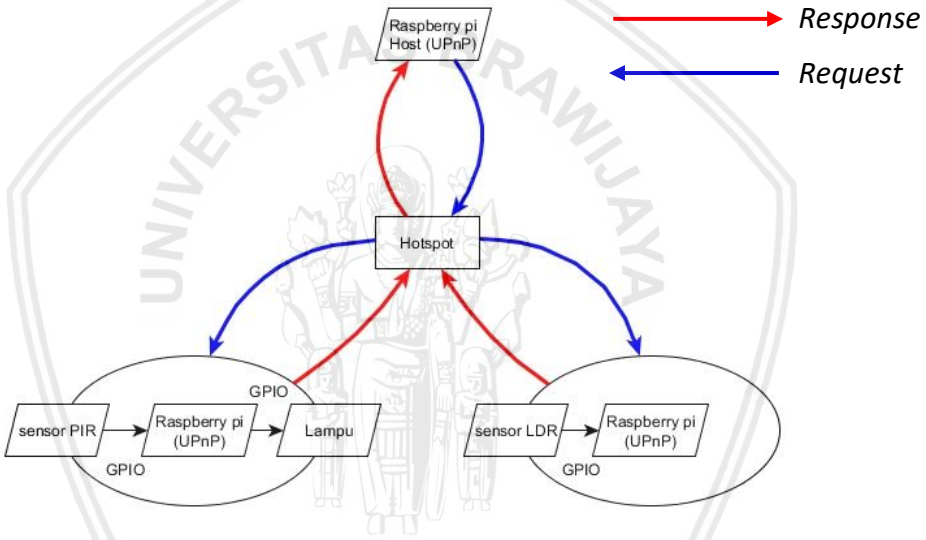
BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan Sistem

Mengenai tahap perancangan bertujuan untuk pembuatan diagram dimana berfungsi sebagai pemodelan sistem yang akan dibuat. Tahap perancangan dibagi menjadi 2 hal yaitu perancangan perangkat keras dan perancangan perangkat lunak.

5.1.1 Perancangan Perangkat Keras

Mengenai tahap perancangan perangkat keras dapat dilihat pada Gambar 5.1 dimana arsitektur tersebut menggunakan perangkat sensor dan aktuator serta untuk komunikasi layanan menggunakan protokol UPnP.

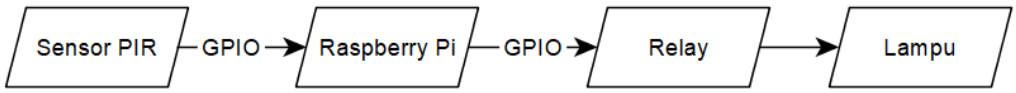


Gambar 5.1 Arsitektur Sistem Secara Umum

Pada Gambar 5.1 dapat dilihat adalah arsitektur sistem secara umum yang mana menggunakan dua *device* dan satu *control point.* Semua perangkat dapat saling berkomunikasi dengan cara terhubung ke dalam jaringan lokal (LAN). Dalam jaringan setiap perangkat dan *control point* akan mendapatkan IP secara otomatis dari pengaturan jaringan lokal (LAN) yang bernama dinamik. Dengan ini *control point* dapat melakukan berbagai hal antara lain seperti mencari perangkat, menemukan perangkat, dan memanfaatkan fungsi dari perangkat sensor dan aktuator itu sendiri. Untuk sistem kontrolnya dapat menggunakan perangkat yang dinamakan *control point* atau *control point* yang mana dengan komunikasi menggunakan jaringan, maka *control point* dapat mengontrol dari jarak jauh selama tetap terhubung dengan jaringan yang terdapat *device*. Komunikasi dalam UPnP mengacu dalam penggunaan *request* dan *response* untuk saling bertukar informasi atau data dari berbagai perangkat yang terdapat dalam jaringan tersebut.

5.1.1.1 Perangkat Sensor Pergerakan

Untuk penyusunan dari perangkat sensor pergerakan dapat dilihat pada diagram blok Gambar 5.2.



Gambar 5.2 Diagram Blok Perangkat Sensor Pergerakan

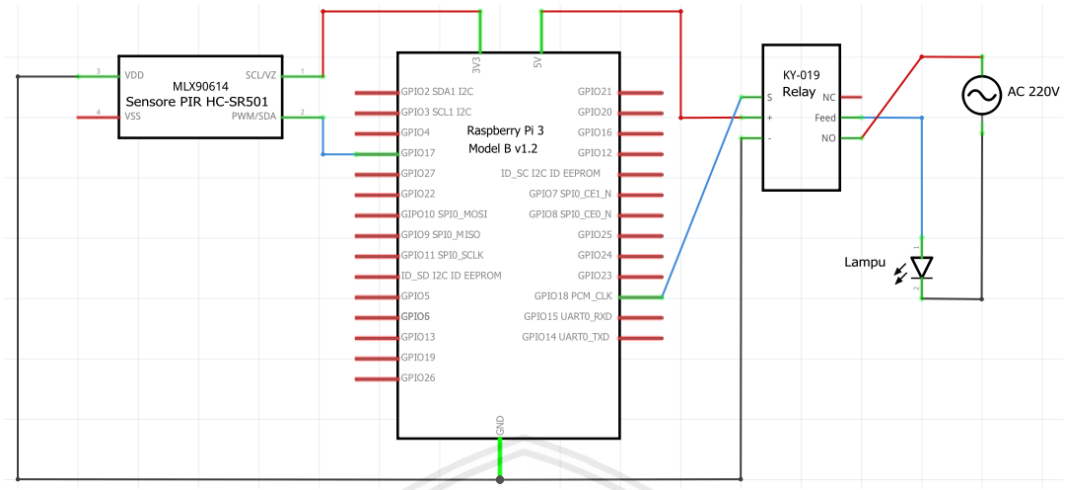
Dapat dilihat pada Gambar 5.2 dimana komponen untuk masukan sendiri adalah sensor PIR, Raspberry pi sebagai pengendali dari satu rangkaian perangkat sensor pergerakan, dan lampu sebagai keluaran atau penanda adanya pergerakan yang ditangkap. Daya dari Raspberry pi diambil dari penggunaan adaptor 5V. Penyusunan perangkat sensor pergerakan dirangkai dengan menggunakan pengkabelan yang dapat dilihat pada Tabel 5.1.

Tabel 5.1 Skema Pengkabelan Perangkat Sensor Pergerakan

Raspberry Pi	Sensor PIR	*Relay*	AC 220V	Lampu
Pin 3,3V	*VDD*	-	-	-
GPIO.0	*SIG*	-	-	-
Pin ground 0V	*GND*	*GND*	-	-
GPIO.1	-	*IN*	-	-
Pin 5V	-	*VCC*	-	-
-	-	*No*	*1.a*	-
-	-	*Common*	-	*2.a*
-	-	-	*1.b*	*2.b*

Untuk penggunaan daya pada sensor PIR karena daya yang dapat digunakan pada sensor sebesar 3-5V, maka untuk pin *VDD* akan dihubungkan pada pin 3.3V dan untuk pin *GND* dihubungkan pada pin *ground* (0V) yang terdapat pada Raspberry pi. Untuk pin *SIG* sendiri berfungsi sebagai masukan dimana jika menangkap pergerakan akan mengirimkan sinyal berupa *high* dan sinyal *low* jika tidak menangkap pergerakan. Untuk itu pin *SIG* akan dihubungkan pada pin GPIO.0 yang terdapat pada Raspberry pi.

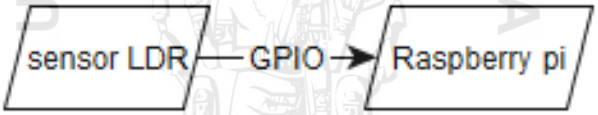
Untuk penggunaan lampu yang berfungsi sebagai keluaran maka dihubungkan dengan *relay.* Untuk masukan dari *relay* pin *IN* dihubungkan pada pin GPIO.1. Untuk pin *NO* dihubungkan pada AC 220V. Untuk bagian 2.a dari lampu dihubungkan pada *Common (Feed)* sedangkan bagian 2.b dihubungkan pada tegangan AC 220V. Untuk lebih jelasnya dapat dilihat pada rangkaian skematik yang terdapat pada Gambar 5.3 berikut.



Gambar 5.3 Diagram Skematik Perangkat Sensor Pergerakan

5.1.1.2 Perangkat Sensor Cahaya

Penyusunan perangkat sensor cahaya dapat dilihat diagram blok yang terdapat pada Gambar 5.4.



Gambar 5.4 Diagram blok perangkat sensor cahaya

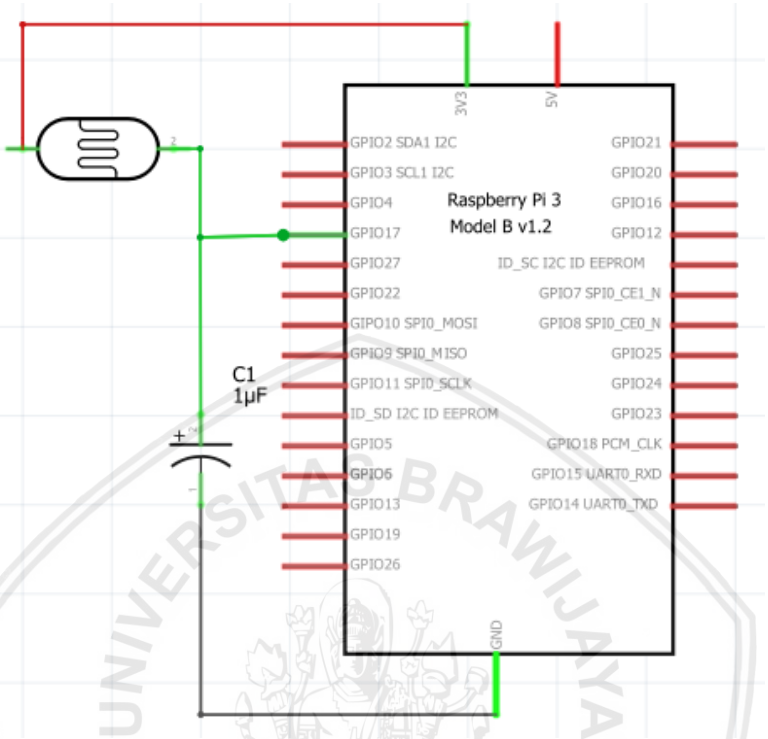
Dari Gambar 5.4 dapat dilihat jika untuk penyusunan perangkat sensor cahaya digunakan sensor LDR sebagai masukan dan Raspberry pi sebagai pusat kendali dari perangkat sensor cahaya. Daya yang digunakan Raspberry pi menggunakan adaptor 5V. Penyusunan dari perangkat sensor cahaya menggunakan pengkabelan yang dapat dilihat pada skema yang tercantum pada Tabel 5.2.

Tabel 5.2 Skema Pengkabelan Perangkat Sensor Cahaya

Raspberry Pi	Sensor LDR
Pin 3,3V	*Wire Terminal*
GPIO.0	*Wire Terminal*

Seperti informasi yang didapat dari *datasheet* untuk penggunaan daya pada sensor cahaya ialah 3-5V, maka untuk satu kaki dari sensor LDR dihubungkan pada pin 3.3V dan untuk kaki satunya yang mana adalah masukan jika sensor menangkap cahaya maka akan bernilai *low* dan jika sensor LDR tidak menangkap cahaya sama sekali maka sensor akan mengirimkan nilai *high* dan pin ini akan

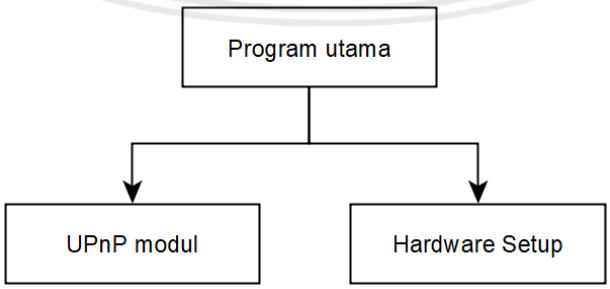
dihubungkan pada pin GPIO.0. Untuk diagram skematik dapat dilihat pada Gambar 5.5.



Gambar 5.5 Diagram Skematik Perangkat Sensor Pergerakan

5.1.2 Perancangan Perangkat Lunak

Pada tahap ini perancangan perangkat lunak dibagi menjadi tiga bagian yaitu program utama dan UPnP modul. Untuk mempermudah perancangan dan pengimplementasiannya maka langkah pertama yaitu membuat diagram blok yang berdasar dari analisis kebutuhan. Berikut adalah Gambar 5.6 mengenai arsitektur perangkat lunak.

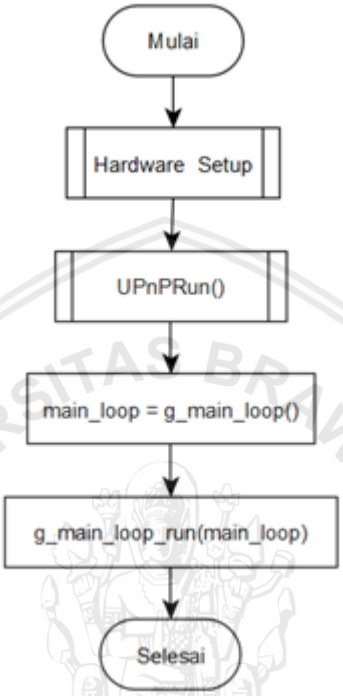


Gambar 5.6 Arsitektur Perangkat Lunak

Dapat dilihat pada Gambar 5.6 program utama berfungsi sebagai *main program,* sedangkan untuk UPnP Modul dan *hardware setup* berfungsi sebagai pendukung yang nantinya akan digunakan oleh sistem utama. Selanjutnya adalah penjelasan dari setiap bagian yang ada pada perancangan perangkat lunak.

5.1.2.1 Program Utama

Program utama atau bisa dibilang sebagai *main program* adalah program yang berfungsi mengeksekusi dari tiap sub program saat sistem dijalankan. Untuk menjalankan setiap fungsi yang terdapat pada sub sistem maka dapat dilihat pada Gambar 5.7 flowchart dari program utama.



Gambar 5.7 Flowchart Program Utama

Dari Gambar 5.7 dapat dilihat saat setelah program utama berjalan diikuti oleh *hardwareSetup,* UPnP*Run* dan *main_loop* akan melakukan perulangan tanpa henti sampai adanya perintah dari pengguna untuk mematikan program utama tersebut. Pada kode sumber program utama juga berisikan *hardware setup* untuk mempermudah pemanggilannya.

5.1.2.2 UPnP Modul

Pada UPnP modul memiliki peran untuk menghubungkan antara UPnP *control* dan UPnP *device.* Oleh sebab itu agar dapat mendukung kinerja program utama maka UPnP Modul harus memiliki fungsi-fungsi seperti *advertising, service,* dan *control.* Selanjutnya agar *service* dapat berjalan maka data yang berasal dari UPnP *control* harus diteruskan kepada *hardware setup.* Sedangkan untuk format dari pembuatan UPnP modul sendiri yaitu menggunakan aturan yang terdapat pada UPnP forum yang bernama *BinaryLight1.xml.* berikut adalah spesifikasi dari *device* yang telah dibuat.

Tabel 5.3 Spesifikasi UPnP *Device*

Elemen	Tipe Data	Nama
DeviceType	*String URI*	urn:schemas-UPnP-org:device:Pir_Led:1
friendlyName	*String*	Jahary
Manufacturer	*String*	OpenedHand
modelName	*String*	Pir_Led
UDN	*Single URI*	uuid:cc93d8e6-6b8b-4f60-87ca-228c36b5b0e8

```
. . . . .
 <device>
   <deviceType> ... </deviceType>
   <friendlyName> ... </friendlyName>
   <manufacturer> ... </manufacturer>
   <modelName> ... </modelName>
   <UDN> ... </UDN>
   <serviceList>
     <service>
       <serviceType> ... </serviceType>
       <serviceId> ... </serviceId>
       <SCPDURL> ... </SCPDURL>
       <controlURL> ... </controlURL>
       <eventSubURL> ...  </eventSubURL>
     </service>
   </serviceList>
 </device>
</root>
```

Gambar 5.8 T*emplate Device* Dan *Service Description*

Dapat dilihat pada Tabel 5.3 dimana *device description* memiliki dua bagian, pada bagian pertama berisikan informasi yang berfungsi sebagai spesifikasi dari *device* sedangkan untuk bagian kedua yaitu *service* atau layanan yang tersedia pada *device* itu sendiri. Dapat dilihat pada gambar 5.8 adalah contoh dari sebuah service yang berisikan nantinya akan berisikan layanan yang dapat dilakukan oleh *device* tersebut. Rancangan pada penelitian ini menggunakan satu tipe *servicelist* yang bernama *switchpower* yang dapat dilihat pada Tabel 5.4.

Tabel 5.4 Rancangan *servicelist* untuk StatusPir_SwitchLed

Elemen	Tipe Data	Nama
serviceType	*String URI*	urn:schemas-UPnP-org:service:StatusPir_SwitchLed:1
ServiceID	*Single URI*	urn:UPnP-org:serviceId: StatusPir_SwitchLed :1
SCPDURL	*Single URI*	/ StatusPir_SwitchLed1.xml
controlURL	*Single URI*	/ StatusPir_SwitchLed /Control
eventSubURL	*Single URI*	/ StatusPir_SwitchLed /Event

Tabel 5.5 Rancangan *servicelist* untuk StatusLdr

Elemen	Tipe Data	Nama
serviceType	*String URI*	urn:schemas-UPnP-org:device:StatusLdr:1
ServiceID	*Single URI*	urn:UPnP-org:serviceId:StatusLdr:1
SCPDURL	*Single URI*	/StatusLdr1.xml
controlURL	*Single URI*	/StatusLdr/Control
eventSubURL	*Single URI*	/StatusLdr/Event

Untuk tahap selanjutnya membuat infromasi mengenai layanan yang dapat dilakukan *device* untuk dimasukkan ke dalam *template* program yang telah disediakan oleh UPnP Forum. Pada Tabel 5.5 adalah rancangan dari *servicelist* untuk *StatusLdr* yang terdapat pada *control point.* Informasi *service* dari *device* sendiri harus secara detail. Berikut adalah gambar dari *template* yang tersedia pada UPnP Forum.

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns="urn:schemas-UPnP-org:device-1-0">
  <specVersion>
    <major> .... </major>
    <minor> .... </minor>
  </specVersion>
 . . . . .
```

Gambar 5.9 *Template Header Service Description*

Pada Gambar 5.9 adalah *template* yang secara urutan program adalah sebagai *header. Service description* memiliki ciri dengan adanya penggunaan <*scpd*>. Atribut pada *scpd* sendiri harus berupa *xmlns* yang berasal dari *urn:schemas-*UPnP-*org:service-1-0.* Setelah bagian *header* ada bagian selanjutnya yang

bernama *actionlist.* pembuatan dari *actionlist* sendiri juga merupakan *template* yang berasal dari UPnP forum dan dapat dilihat pada Gambar 5.10.

```
. . . . .
<actionList>
    <action>
      <name> ... </name>
      <argumentList>
        <argument>
          <name> ... </name>
          <relatedStateVariable> ... </relatedStateVariable>
          <direction> ... </direction>
        </argument>
      </argumentList>
    </action>
  </actionList>
 . . . . .
```

Gambar 5.10 *Template actionLlist*

Action list adalah sebuah *action* atau pergerakan dimana setiap pergerakan akan memiliki sebuah *argument. Argument* sendiri berfungsi untuk mengarahkan *action* terhadap arah nilai sebagai *input* ataupun *output.*

Tabel 5.6 Rancangan *actionList*

actionList	***argumentList***		
	name	***relatedStateVariable***	***direction***
SetTarget	*newTargetValue*	*Target*	*in*
GetTarget	*RetTargetValue*	*Target*	*Out*
GetStatus	*ResultStatus*	*Status*	*Out*

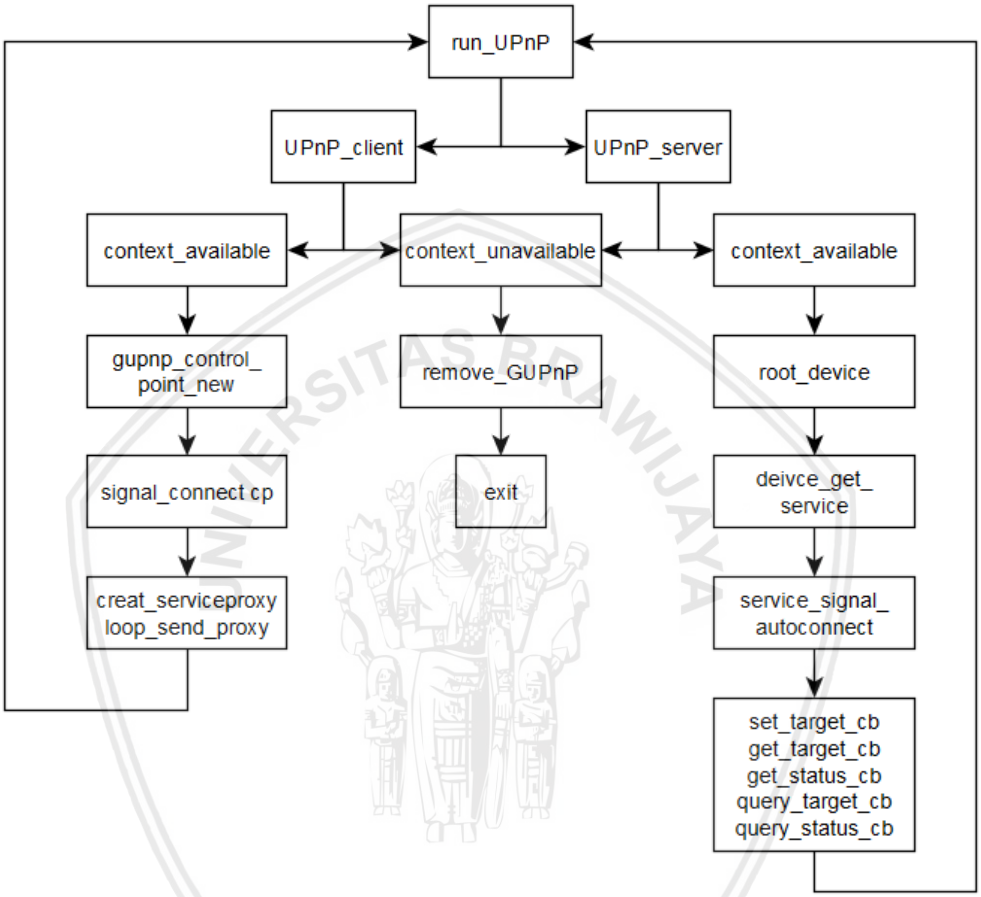
Tabel 5.7 Rancangan *serviceStateTable*

serviceStateTable			
name	***dataType***	***defaultValue***	***sendEvents***
Target	*Boolean*	*0*	*no*
Status	*Boolean*	*0*	*yes*

Dapat dilihat pada Tabel 5.6 dan Tabel 5.7 adalah sebuah rancangan untuk *actionList* dan *serviceStateTable* yang terdapat pada *service description* pada *StatusPir_SwitchLed* dan *StatusLdr*. Tipe data yang digunakan adalah *Boolean*

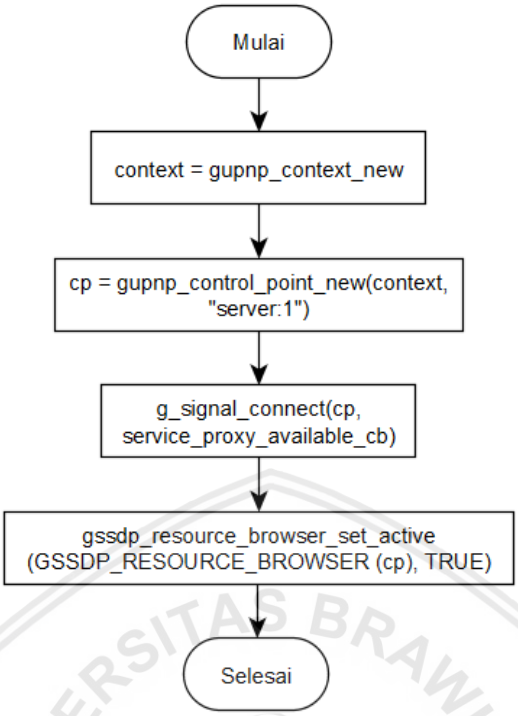
karena seperti karakteristiknya *Boolean* memiliki dua macam data yaitu satu dan nol, seperti halnya dengan sakelar yang memiliki dua macam *switch control* dari nyala ke mati ataupun sebaliknya.

Tahap selanjutnya adalah perancangan program UPnP modul untuk mengimplementasikan protokol UPnP yang menggunakan *library G*UPnP. Untuk diagram UPnP modul dapat dilihat pada Gambar 5.11.



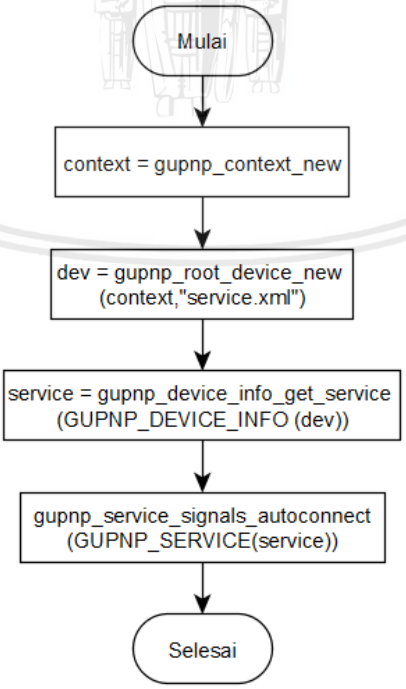
Gambar 5.11 Diagram Siklus Upnp Modul

Dapat dilihat pada Gambar 5.11 adalah urutan dari siklus UPnP modul yang dimulai dari pertama kali *device* dijalankan yaitu *run*_UPnP. Kemudian proses selanjutnya yaitu terdapat pilihan antara menjalankan *client* maupun *server.* selanjutnya menentukan apakah *context_unavailable* atau *context_available* yang mana merupakan pemindaian apakah *context* pada setiap *device* memilikinya. *s*aat ada *context* yang terbaca dan itu adalah menjalankan setiap perintah yang terdapat pada setiap mekanisme yang berada pada *device* maupun *server.*



Gambar 5.12 Diagram Alir Upnp Modul *Client* (*Control Point)*

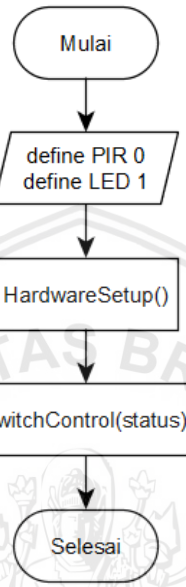
Pada *control point* akan memindai *service* apa yang terdapat pada *device* tersebut. Selanjutnya mencocokan *service* yang dimiliki oleh *server* dan *control point* apakah ada kecocokan antar *service* tersebut. Setelah ada kecocokan maka *action* dapat berjalan. Diagram alir UPnP Modul ditunjukkan pada Gambar 5.12.



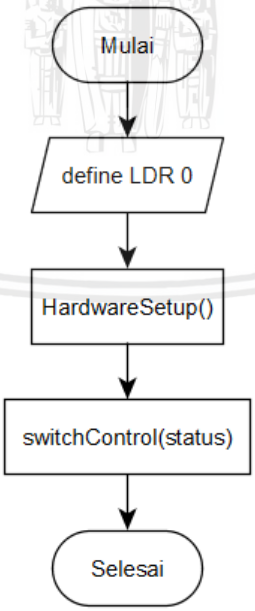
Gambar 5.13 Diagram Alir Upnp Modul *Server*

5.1.2.3 *Hardware Setup*

Pada tahap perancangan perangkat keras ini menggunakan lampu, sensor PIR, dan sensor LDR yang mana untuk dapat menjalankan fungsi dari lampu dan sensor yaitu menggunakan pustaka yang bernama WiringPi. WiringPi adalah sebuah pustaka untuk mengaktifkan pin GPIO pada Raspberry pi. Untuk diagram alir dari *hardware Setup* dapat dilihat pada Gambar 5.14 dan Gambar 5.15.



Gambar 5.14 Diagram Alir *Hardwardsetup* Perangkat Sensor Pergerakan



Gambar 5.15 Diagram Alir *Hardwaresetup* Perangkat Sensor Cahaya

5.2 Implementasi Sistem

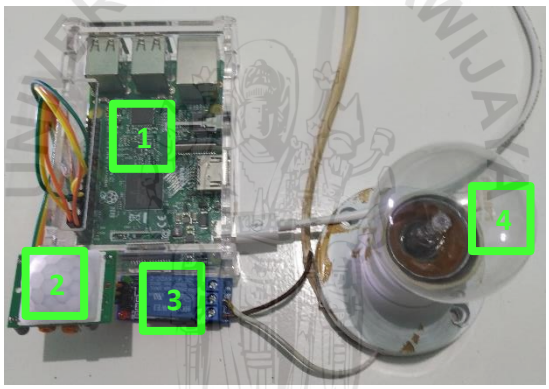
Pada sub bab implementasi yaitu membahas mengenai tahapan yang berdasarkan dari perancangan yang telah dirancang sebelumnya dan dibagi menjadi dua bagian yaitu implementasi perangkat keras dan perangkat lunak.

5.2.1 Implementasi perangkat keras

Pada tahap implementasi perangkat keras ini adalah untuk menyusun dan merangkai komponen yang telah dirancang pada blok diagram pada tahap perancangan.

5.2.1.1 Perangkat sensor pergerakan

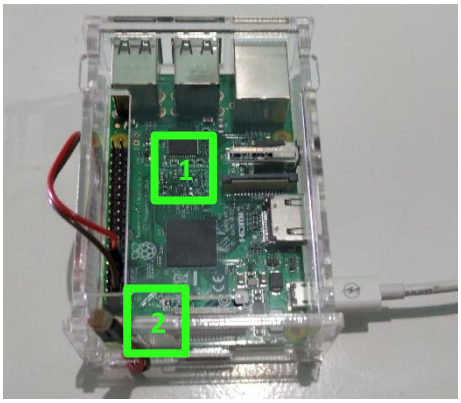
Pada Gambar 5.16 merupakan hasil implementasi dari perancangan perangkat sensor pergerakan. Sebagai pusat kontrol perangkat yaitu Raspberry pi (1). Untuk sensor PIR (2) diletakkan pada pojok kiri bawah dengan menghubungkan setiap pin dari sensor PIR ke pin GPIO . Sedangkan untuk *relay* (3) berada ditengah bawah sebagai penghubung antara raspberry pi dengan lampu (4). Lampu sendiri berfungsi sebagai aktuator.



Gambar 5.16 Implementasi Perangkat Sensor Pergerakan

5.2.1.2 Perangkat sensor cahaya

Pada Gambar 5.17 adalah hasil implementasi dari perancangan perangkat sensor cahaya. Sebagai pusat kontrol menggunakan Raspberry pi (1). Untuk sensor LDR (2) diletakkan pada bagian kiri bawah dan setiap pin pada sensor LDR dihubungkan pada pin GPIO yang berada pada Raspberry pi.



Gambar 5.17 Implementasi Perangkat Sensor Cahaya

5.2.2 Implementasi Perangkat Lunak

Pada implementasi perangkat lunak mengacu pada perancangan dari program utama dan UPnP modul. Pada setiap *device* akan memiliki program utama dan UPnP modul sendiri.

5.2.2.1 Implementasi Program Utama

Program utama adalah sebuah kumpulan kode sumber yang mana bila dieksekusi akan memanggil fungsi dari UPnP modul dan menjalankan *hardware setup* untuk ikut dieksekusi. Program utama berjumlah tiga yang mana setiap *device* memiliki program utama sendiri. Berikut adalah penjelasan dari program utama pada setiap *device.*

1. Program utama pada perangkat sensor pergerakan

Tabel 5.8 Kode Sumber Program Utama Perangkat Sensor Pergerakan

Fungsi Utama UPnP dan Penghubung dengan UPnP Modul

```
1.  #include <libgUPnP/gUPnP.h>
2.  #include <stdlib.h>
3.  #include <gmodule.h>
4.  #include <wiringPi.h>
5.
6.  #define PIR 0
7.  #define LED 1
8.
9.  . . . . .
10.
11. int
12. main (G_GNUC_UNUSED int argc, G_GNUC_UNUSED char **argv)
13. {
14.   wiringPiSetup ();
15.
16.   . . . . .
17.
18.   dev = gUPnP_root_device_new (context, "Pir_Led1.xml", ".",
19. &error);
20.   if (error != NULL) {
21.     g_printerr ("Error creating the GUPnP root device: %s\n",
22.                 error->message);
23.
24.     g_error_free (error);
25.
26.     return EXIT_FAILURE;
27.   }
28.   gUPnP_root_device_set_available (dev, TRUE);
29.
30.   service = gUPnP_device_info_get_service
31.     (GUPNP_DEVICE_INFO (dev), "urn:schemas-UPnP-
32. org:service:StatusPir_SwitchLed:1");
33.   if (!service) {
34.     g_printerr ("Cannot get Pir_Led1 service\n");
35.
36.     return EXIT_FAILURE;
37.   }
38.
39.   cp = gUPnP_control_point_new (context,
40.                 "urn:schemas-UPnP-org:service:NotifPir:1");
41.
```

Fungsi Utama UPnP dan Penghubung dengan UPnP Modul

```
42.   g_signal_connect (cp,
43.                     "service-proxy-available",
44.                     G_CALLBACK (service_proxy_available_cb),
45.                     NULL);
46.
47.   gssdp_resource_browser_set_active (GSSDP_RESOURCE_BROWSER
48. (cp), TRUE);
49.
50. . . . . .
51.
52.   return EXIT_SUCCESS;
53. }
```

Tabel 5.8 adalah kode sumber untuk program utama dari perangkat sensor pergerakan. Pertama program menginisialisasi dari *Library* untuk mendukung program utama (baris 1-4). Melakukan inisialisasi GPIO pada Raspberry pi untuk sensor PIR dan lampu (baris 6-7). Mengaktifkan pin GPIO agar dapat melakukan *input* maupun *output* dari Raspberry pi (baris 14). Selanjutnya melakukan permintaan akses ke dalam *root* untuk mengakses dokumen bernama *"Pir_Led1.xml"* dan notifikasi jika akses *root* tidak dapat berjalan (baris 18-27).

Kemudian meminta akses untuk membaca dokument yang berisikan daftar *service* dari *device* dan memberikan notifikasi jika dokument tersebut tidak ada maupun tidak bisa diakses (baris 30-37).vMembuat sebuah *control point* dan mencari *service "NotifPir"* yang terdapat dalam jaringan lokal tersebut (baris 39-40). Dalam jaringan lokal jika menemukan *service* yang dicari maka akan meminta terhubung dan meminta pemberitahuan jika koneksi dengan *service* sudah terjalin (baris 42-45). Menjalankan GSSDP di dalam jaringan dan melakukan perulangan terus-menerus sampai ada perintah mematikan dari pengguna (baris 47).

Tabel 5.9 Kode Sumber *Void Set_target_cb*

Menerima Fungsi *Switch* dari *Control point*

```
1.  G_MODULE_EXPORT void
2.  set_target_cb (GUPnPService          *service,
3.                 GUPnPServiceAction    *action,
4.                 G_GNUC_UNUSED gpointer user_data)
5.  {
6.    gUPnP_service_action_get (action,
7.                              "newTargetValue",      G_TYPE_BOOLEAN,
8.  &target,
9.                              NULL);
10.
11.   if (target != status) {
12.      status = target;
13.      gUPnP_service_notify (service,
14.                            "Status", G_TYPE_BOOLEAN, status,
15.                            NULL);
16.      if (!quiet)
17.      {
18.        g_print ("Host switch the light to %s.\n", status ? "on" :
19. "off");
20.
21.      }
22.         if (status == TRUE){
```

Menerima Fungsi *Switch* dari *Control point*	
23.	digitalWrite (LED, HIGH);
24.	} else {
25.	digitalWrite (LED, LOW);
26.	}
27.	}
28.	gUPnP_service_action_return (action);
29.	}

Pada Tabel 5.9 dapat dilihat untuk nama dari *void* tersebut adalah *set_target_cb* (baris 2). Fungsi dari *void* ini antara lain menerima nilai dari *target* yang dikirim dari *device* lain berupa data yang bertipe *Boolean* yang mana data tersebut salah satu dari daftar dari *optionContext.* Isi dari data tersebut adalah *on* atau *off.* jika isi dari data berbeda dari *status* yang sekarang dimiliki, maka nilai dari *status* akan berubah dan akan menukan kondisi dari lampu saat itu juga. Bila nilai dari data yang dikirim sama dengan *status* maka tidak akan ada perubahan pada notifikasi dan lampu.

Tabel 5.10 Kode Sumber *Void Send_cmd*

Membaca dan Mengirim Data Sensor kepada *Control point*	
1.	static void
2.	send_cmd (GUPnPServiceProxy *proxy)
3.	{
4.	delay (2000);
5.	if (digitalRead(PIR)) {
6.	mode = ON;
7.	digitalWrite (LED, HIGH);
8.	g_print ("The light is now on.\n");
9.	} else {
10.	mode = OFF;
11.	digitalWrite (LED, LOW);
12.	g_print ("The light is now off.\n");
13.	}
14.	GError *error = NULL;
15.	target = mode;
16.	if (!gUPnP_service_proxy_send_action (proxy, "SetTarget", &error,
17.	"newTargetValue", G_TYPE_BOOLEAN, target,
18.	NULL,
19.	NULL)) {
20.	g_print ("==Host Down== \n");
21.	return;
22.	}
23.	return;
24.	}

Pada Tabel 5.10 dapat dilihat nama untuk *void* yaitu *send_cmd* (baris 2). Pembacaan sensor LDR untuk mengetahui apakah sensor menangkap cahaya ataupun tidak yang selanjutnya digunakan untuk menentukan *mode* dan memberikan masukan pada kondisi lampu sesuai masukan dari sensor PIR (baris 5-13). *Mode* di masukkan ke dalam *target* (baris 15). Untuk menentukan data yang dikirim kepada *control point* diterima atau tidak makan akan ada notifikasi berupa *"Host Down"* jika *control point* mengalami kegagalan penerimaan data (baris 16-23).

2. Proram utama perangkat sensor cahaya

Program utama pada perangkat sensor cahaya hampir sama dengan perangkat sensor pergerakan. Untuk perbedaannya dapat dilihat Tabel 5.8 pada baris 6-7 yang sebelumnya deklarasi mengenai sensor PIR dan lampu dirubah dengan deklarasi pin GPIO.0 dengan nama LDR. Selanjutnya mengganti nama berkas untuk *device description* dan *service description* yang sebelumnya *Pir_Led1.xml* menjadi *Ldr1.xml* (Baris 18) dan *StatusPir_SwitchLed* menjadi *StatusLdr* (baris 32). Perubahan status dari sensor yang nantinya dikirimkan pada *control point* yang sebelumnya *NotifPir* menjadi *NotifLdr* (baris 40).

Pada Tabel 5.9 Untuk perangkat sensor cahaya tidak menggunakan kode sumber pada baris 20-24 yang berfungsi untuk menyalakan atau mematikan lampu. Selanjutnya pada Tabel 5.10 yang membedakannya adalah variabel PIR ganti dengan LDR (baris 5). Menghilangkan baris 7-8 dan 11-12 yang berfungsi menyalakan atau mematikan lampu yang mengikuti kondisi dari sensor.

3. Program utama pada *Control point*

Pada *control point* memiliki tiga program utama dimana untuk dua program utama sebagai notifikasi yang diterima dari *device* yang membedakannya adalah ada *device* yang menggunakan sensor PIR dan sensor LDR. Perbedaan lainnya berupa nama dokumen dan isi dari *service.* Untuk keluaran dari program utama ini sama-sama berupa notifikasi apakah sensor menangkap pergerakan manusia atau menangkap cahaya. Untuk program utama yang terakhir sebagai *switch power* pada lampu yang berada pada *device* perangkat sensor pergerakan.

Tabel 5.11 Kode Sumber Program Utama *Control point Notifldr*

Fungsi Utama UPnP dan Penghubung dengan UPnP Modul	
1.	`#include <libgUPnP/gUPnP.h>`
2.	`#include <stdlib.h>`
3.	`#include <gmodule.h>`
4.	
5.	`. . . . .`
6.	
7.	`int`
8.	`main (G_GNUC_UNUSED int argc, G_GNUC_UNUSED char **argv)`
9.	`{`
10.	
11.	`  . . . . .`
12.	
13.	`  dev = gUPnP_root_device_new (context, "Ldr1.xml", ".", &error);`
14.	`  if (error != NULL) {`
15.	`    g_printerr ("Error creating the GUPnP root device: %s\n",`
16.	`                error->message);`
17.	
18.	`    g_error_free (error);`
19.	
20.	`    return EXIT_FAILURE;`
21.	`  }`
22.	`  gUPnP_root_device_set_available (dev, TRUE);`
23.	
24.	`  service = gUPnP_device_info_get_service`
25.	`    (GUPNP_DEVICE_INFO          (dev),          "urn:schemas-UPnP-`
26.	`org:service:NotifLdr:1");`
27.	`  if (!service) {`
28.	`    g_printerr ("Cannot get NotifLdr1 service\n");`

Fungsi Utama UPnP dan Penghubung dengan UPnP Modul	
29.	
30.	return EXIT_FAILURE;
31.	}
32.	
33.	gUPnP_service_signals_autoconnect (GUPNP_SERVICE (service),
34.	NULL, &error);
35.	if (error) {
36.	g_printerr ("Failed to autoconnect signals: %s\n", error-
37.	>message);
38.	g_error_free (error);
39.	
40.	return EXIT_FAILURE;
41.	}
42.	
43.	. . . . .
44.	
45.	return EXIT_SUCCESS;
46.	}

Pada Tabel 5.11 untuk mengawali kode sumber dari program utama adalah inisialisasi *library* pendukung (baris 1-3). Permintaan akses ke dalam *root* untuk mengakses dokumen bernama *"Ldr1.xml"* dan notifikasi jika akses *root* tidak dapat berjalan (baris 13-21). Permintaan akses untuk membaca dokumen yang berisikan daftar *service* dari *device* dan memberikan notifikasi jika dokument tersebut tidak ada maupun tidak bisa diakses (baris 24-30). Otomatis dapat mengakses jaringan dan mengirimkan secara berkala mengenai *service* yang dapat dilakukan oleh *device* tersebut (baris 50-56).

Menjalankan perulangan dan membersihkan *cache* dari setiap perulangan pada *library G*UPnP (baris 33-41). Untuk fungsi pendukung dari program ini yang berupa *void* dimana memiliki kesamaan dengan *void* yang berada pada perangkat sensor pergerakan yang ditunjukkan pada Tabel 5.9. Untuk perbedaannya dari *void* perangkat sensor pergerakan hanya pada Tabel 5.9 pada baris 20-24 tidak ada untuk mengganti kondisi dari lampu.

Untuk kode program pada *NotifPir* memiliki kesamaan dengan kode program *NotifLdr,* yang membedakan dari dua buah program tersebut terletak pada baris 31 berisi *Ldr1.xml* menjadi *Pir1.xml* dan baris 43 untuk *service* yang sebelumnya *NotifLdr* menjadi *NotifPir.*

Tabel 5.12 Kode Sumber Program Utama *Switch Power*

Mengirim Fungsi *Switch* kepada Perangkat Sensor Pergerakan	
1.	#include <libgUPnP/gUPnP.h>
2.	#include <stdlib.h>
3.	
4.	. . . . .
5.	
6.	int
7.	main (int argc, char **argv)
8.	{
9.	
10.	. . . . .
11.	
12.	if (argc != 2) {
13.	usage (optionContext);
14.	return EXIT_FAILURE;

Mengirim Fungsi *Switch* kepada Perangkat Sensor Pergerakan

```
15.    }
16.
17.    if (g_str_equal (argv[1], "on")) {
18.      mode = ON;
19.    } else if (g_str_equal (argv[1], "off")) {
20.      mode = OFF;
21.    } else if (g_str_equal (argv[1], "toggle")) {
22.      mode = TOGGLE;
23.    } else {
24.      usage (optionContext);
25.      return EXIT_FAILURE;
26.    }
27.
28.  . . . . .
29.
30.    cp = gUPnP_control_point_new (context,
31.                         "urn:schemas-UPnP-
32.  org:service:StatusPir_SwitchLed:1");
33.
34.     g_signal_connect (cp,
35.                       "service-proxy-available",
36.                       G_CALLBACK (service_proxy_available_cb),
37.                       NULL);
38.
39.    gssdp_resource_browser_set_active (GSSDP_RESOURCE_BROWSER (cp),
40.  TRUE);
41.
42.    . . . . .
43.
44.    return EXIT_SUCCESS;
45.  }
```

Pada Tabel 5.12 adalah kode sumber dari program utama untuk *switch power.* Program diawali dengan menginsialisasi *library* untuk mendukung kinerja dari program utama (baris 1-2). Selanjutnya memberikan pilihan *input* dimana jika kondisi dari *input* tidak termasuk pada tiga *optionContext* maka akan menampilkan *"option parsing failed"*. Pada sebuah *mode* berasal dari *input* yang diketik pada *terminal* yang disediakan oleh operasi sistem *Linux.* Perubahannya jika kita mengetik *on* maka *mode* juga akan berubah menjadi *on* dan berlaku dengan pilihan lainnya (baris 17-26). Kemudian membuat sebuah *control point* dan mencari *service "StatusPir_SwitchLed"* yang terdapat dalam jaringan lokal tersebut (baris 30-32).

Pada jaringan lokal jika menemukan *service* yang dicari maka akan meminta terhubung dan meminta pemberitahuan jika koneksi dengan *service* sudah terjalin (baris 34-37). Menjalankan GSSDP di dalam jaringan dan melakukan perulangan terus-menerus sampai ada perintah mematikan dari pengguna (baris 39). Selanjutnya adalah penjelasan dari sebagian *void* yang berada pada program utama *switch power*.

Tabel 5.13 Kode Sumber Void Send_cmd Pada *Switch Power*

Notifikasi Keberhasilan Fungsi *Switch*

```
1.  static void
2.  send_cmd (GUPnPServiceProxy *proxy)
3.  {
4.    GError *error = NULL;
```

Notifikasi Keberhasilan Fungsi *Switch*

```
5.    gboolean target;
6.
7.    if (mode == TOGGLE) {
8.      if (!gUPnP_service_proxy_send_action
9.          (proxy, "GetStatus", &error,
10.                                    NULL,
11.                                    "ResultStatus",    G_TYPE_BOOLEAN,
12.  &target, NULL)) {
13.       return;
14.     }
15.     target = ! target;
16.   } else {
17.     target = mode;
18.   }
19.
20.   if (!gUPnP_service_proxy_send_action (proxy, "SetTarget", &error,
21.                                         "newTargetValue",
22.  G_TYPE_BOOLEAN, target, NULL,
23.                                         NULL)) {
24.     return;
25.     goto error;
26.   } else {
27.     if (!quiet) {
28.       g_print ("Set switch to %s.\n", target ? "on" : "off");
29.     }
30.   }
31.
32.  . . . . .
33.
34.  }
```

Tabel 5.13 merupakan program *void* dengan nama *send_cmd* (baris 2). Variabel *target* bertipe *Boolean* (baris 5). Fungsi *mode toggle* yaitu mengganti nilai dari *target* saat ini pada *device* yang akan dirubah pada *mode* lainnya atau bisa diartikan jika keadaan sekarang *on* maka akan dirubah menjadi *off* dan sebaliknya (baris 7-17). Selanjutnya mengatur *target* untuk dapat dikirim pada *device* yang dituju dan jika *device* tersebut merespon dari data yang dikirim maka akan menampilkan notifikasi "*Set switch to on"* atau *"off"* tergantung dari kata yang diketik pada *cmd* (baris 19-28). Penggunaan *switch* bertujuan agar saat kondisi sensor mendapatkan pergerakan namun pengguna tetap ingin mematikan lampu secara manual.

5.2.2.2 Implementasi UPnP Modul

UPnP modul merupakan sistem pendukung untuk menjalankan protokol UPnP. Untuk *library* dari UPnP peneliti menggunakan GUPnP. Fungsi *G*UPnP sendiri sebagian untuk mendeskripsikan dari sebuah *device* dan *service* dengan format yang tersedia pada UPnP forum dengan tipe format ialah XML. Tujuan pembuatan deskripsi pada *device* dan *service* agar pengguna dapat mencari dan menemukan *device* atau *service* yang ingin dijalankan sesuai keinginannya. Komunikasi yang dijalankan untuk mendapatkan deskripsi yaitu menggunakan komunikasi *unicast* dengan memanfaatkan protokol HTTP GET. Berikut adalah penjelasan dari tiap bagian *device description* dan *service description.*

Tabel 5.14 Kode Sumber *Device Description* Dan *Servicelist*

Fungsi Deskripsi Perangkat dan *Service*

```
1.   . . . . .
2.
3.    <device>
4.      <deviceType>urn:schemas-UPnP-org:device:Led:1</deviceType>
5.      <friendlyName>Jahary</friendlyName>
6.      <manufacturer>OpenedHand</manufacturer>
7.      <modelName>Switch Light</modelName>
8.      <UDN>uuid:cc93d8e6-6b8b-4f60-87ca-228c36b5b0e8</UDN>
9.
10.     <serviceList>
11.       <service>
12.         <serviceType>urn:schemas-UPnP-
13. org:service:StatusPir_SwitchLed:1</serviceType>
14.         <serviceId>urn:UPnP-
15. org:serviceId:StatusPir_SwitchLed:1</serviceId>
16.         <SCPDURL>/StatusPir_SwitchLed1.xml</SCPDURL>
17.         <controlURL>/StatusPir_SwitchLed/Control</controlURL>
18.         <eventSubURL>/StatusPir_SwitchLed/Event</eventSubURL>
19.       </service>
20.   . . . . .
```

Pada Tabel 5.14 merupakan tampilan dari sebagian kode sumber *device description* dan *servicelist* dimana memiliki nama *device* "*Led:1",* dengan *servicelist "StatusPir_SwitchLed:1"*. Untuk dokumen *device description* dan *service description* memiliki kesamaan dengan dokumen pada program lainnya. Hanya saja yang membedakan pada bagian penamaan pada *deviceType, modelName, ServiceType, ServiceId, controlURL,* dan *eventSubURL.* Untuk kode sumber yang lengkap dapat dilihat pada lampiran.

Tabel 5.15 Kode Sumber *ActionList*

Fungsi Penentuan I/O

```
1.   . . . . .
2.
3.   </actionList>
4.     <serviceStateTable>
5.       <stateVariable sendEvents="no">
6.         <name>Target</name>
7.         <dataType>boolean</dataType>
8.         <defaultValue>0</defaultValue>
9.       </stateVariable>
10.      <stateVariable sendEvents="yes">
11.        <name>Status</name>
12.        <dataType>boolean</dataType>
13.        <defaultValue>0</defaultValue>
14.      </stateVariable>
15.    </serviceStateTable>
16.   . . . . .
```

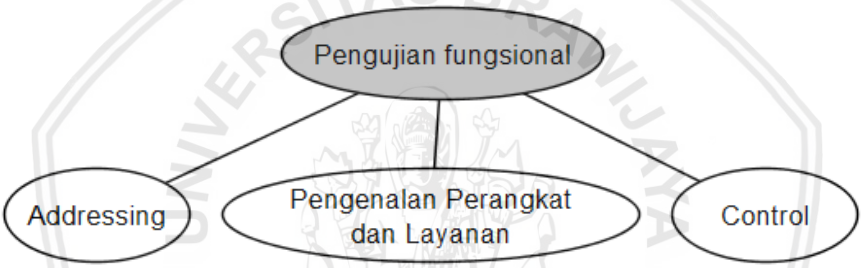
Tabel 5.15 adalah sebagian dari kode sumber *actionlist* dari sebuah *service switch power.* Di dalam *actionlist* terdapat tindakan apakah *service* memiliki fungsi sebagai *input* atau *output* dengan tipe variabel yaitu *Boolean* dengan nama *target.* Untuk *service* pada program utama yang terdapat di perangkat sensor pergerakan, perangkat sensor cahaya, dan *control point* memiliki *actionlist* yang sama karena mengirimkan data berupa *on* dan *off*.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini menjelaskan mengenai proses dalam pengujian dan analisis yang dimulai dengan pengujian pada lingkungan, tujuan pengujian, prosedur pengujian, hasil pengujian. Setelah hasil pengujian didapat maka selanjutnya menganalisis hasil dari pengujian tersebut apakah sudah sesuai dengan implementasi yang mengacu pada rancangan fungsional dan nonfungsional.

6.1 Pengujian Fungsional

Dalam pengujian fungsional ini mengacu pada implementasi fungsional apakah sudah sesuai dengan rancangan fungsional. Isi dari pengujian ini meliputi kemampuan dari UPnP yang terdiri dari *addressing, discovery, description,* dan *control.* Diagram blok pada Gambar 6.1 berfungsi untuk mempermudah proses pengujian.



Gambar 6.1 Diagram Blok Pengujian Fungsional

6.1.1 *Addressing*

6.1.1.1 Tujuan Pengujian

Tujuan dari pengujian *addressing* ialah apakah *device* dan *control point* dapat terhubung ke dalam jaringan lokal (WLAN). Dapat mengkonfigurasi ulang saat *device* dan *control point* tidak dapat terhubung ke dalam jaringan Prosedur Pengujian.

6.1.1.2 Prosedur Pengujian

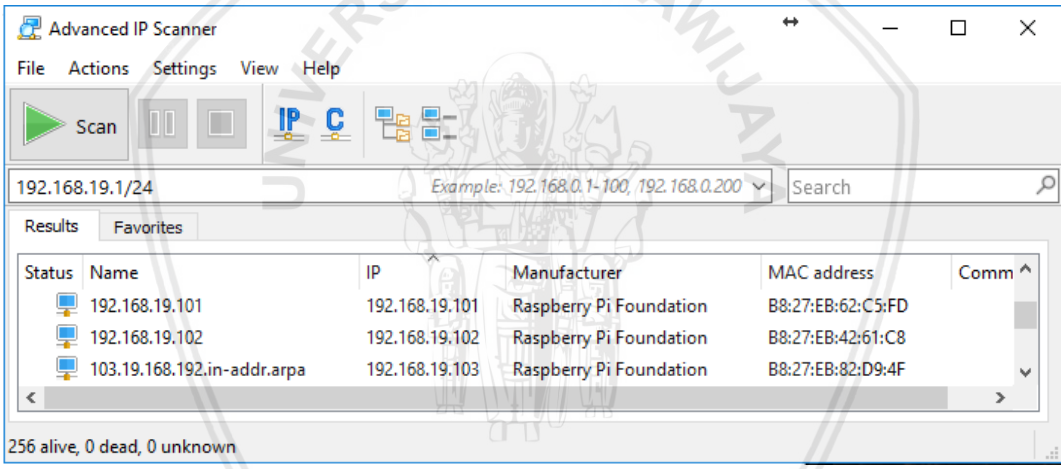
Prosedur pengujian dilakukan dengan menyalakan *device* dan *control point* dan apakah *device* dan *control point* dapat terkoneksi ke dalam jaringan dengan pembuktian melalui sebuah aplikasi yang bernama Advance IP Scanner.

6.1.1.3 Hasil Pengujian

Hasil pengujian ialah tahap dimana *device* dan *control point* diuji dengan berbagai skenario dan selanjutnya diamati apa yang terjadi dari hasil pengujian tersebut. skenario hasil pengujian dapat dilihat pada Tabel 6.1 berikut :

Tabel 6.1 Hasil Pengujian *Addressing*

No.	Skenario	Hasil Pengamatan	Kesimpulan
1.	*Device* dan *control point* tidak mempunyai SSID dan *password*	*Device* dan *control point* tidak dapat terhubung ke dalam jaringan lokal (WLAN)	Sesuai
2.	*Device* dan *control point* mempunyai SSID dan *password*	• *Device* dan *control point* dapat terhubung dengan jaringan lokal (WLAN) • *Device dan control point* mendapatkan IP • SSID dan *password* disimpan pada konfigurasi dan memori	Sesuai



Gambar 6.2 *Screenshot* Program Advanced IP Scanner

Gambar 6.2 memperlihatkan bahwa pada jaringan lokal dari *hostpot, device* dan *control point* dapat terkoneksi dan mendapatkan IP. IP 192.168.19.101 adalah perangkat sensor cahaya, untuk IP 192.168.19.102 sebagai perangkat sensor pergerakan dan untuk IP 192.168.19.103 sebagai *control point.* Konfigurasi dan penyimpanan SSID dan *password* untuk perangkat *smart home* dapat melalui *sudo gedit /etc/wpa_supplicant/wpa_supplicant.conf* pada *terminal.* Dapat dilihat pada Gambar 6.3 untuk tampilan dari konfigurasi SSID dan *password*.

Gambar 6.3 Konfigurasi SSID Dan *Password*

6.1.1.4 Analisis Hasil Pengujian

Dari hasil yang ditampilkan pada Tabel 6.1 dapat dianalisis bahwa perangkat *smart home* dapat terhubung ke dalam jaringan lokal (WLAN). Dapat mengkonfigurasi SSID baru bilamana perangkat *smart home* tidak mengenali jaringan lokal yang baru. Kesimpulannya adalah bahwa perangkat *smart home* dapat berjalan pada jaringan yang berbeda tanpa perlu adanya perubahan di dalam kode program *smart home*.

6.1.2 Pengenalan Perangkat dan Layanan

6.1.2.1 Tujuan Pengujian

Pengenalan perangkat dan layanan diuji untuk mengetahui apakah perangkat *smart home* dapat mengenalkan deskripsi dan mengenalkan layanan apa saja yang dapat dilakukan oleh perangkat *smart home* tersebut ke dalam jaringan lokal (WLAN).

6.1.2.2 Prosedur Pengujian

Perangkat *smart home* terhubung ke dalam jaringan kemudian pada *control point* melakukan pencarian ke dalam jaringan untuk mengetahui *device* dan *service* apa saja yang tersedia dalam jaringan lokal (WLAN) tersebut.

6.1.2.3 Hasil Pengujian

Pada Tabel 6.2 menunjukkan hasil dari pengujian terhadap pengenalan perangkat dan layanan.

Tabel 6.2 Hasil Pengujian Pengenalan Perangkat Dan Layanan

No.	Skenario	Hasil Pengamatan	Kesimpulan
1.	*Control point* melakukan pencarian *device* pada jaringan lokal (WLAN) saat *device* dinyalakan	*Control point* mendapatkan spesifikasi *device* pada setiap *device* yang berada dalam jaringan lokal (WLAN)	Sesuai
2.	*Control point* melakukan pencarian *service* pada jaringan lokal (WLAN) saat *device* dinyalakan	*Control point* mendapatkan *service* pada setiap *device* yang berada dalam jaringan lokal (WLAN)	Sesuai

```
Device available:
        type:     urn:schemas-upnp-org:device:Ldr:1
        location: http://192.168.19.101:45587/cc93d8e6-6b8b-4f60-87ca
Service available:
        type:     urn:schemas-upnp-org:service:StatusLdr:1
        location: http://192.168.19.101:45587/cc93d8e6-6b8b-4f60-87ca
Device available:
        type:     urn:schemas-upnp-org:device:Pir_Led:1
        location: http://192.168.19.102:39209/cc93d8e6-6b8b-4f60-87ca
Service available:
        type:     urn:schemas-upnp-org:service:StatusPir_SwitchLed:1
        location: http://192.168.19.102:39209/cc93d8e6-6b8b-4f60-87ca
```

Gambar 6.4 *Screenshot* Program Pada *Control point*

6.1.2.4 Analisis Hasil Pengujian

Analisis untuk hasil pengujian pengenalan perangkat dan layanan adalah perangkat sensor pergerakan dapat mengirim *device description* dan *service* ke dalam jaringan menggunakan protokol SSDP. Pada Gambar 6.4 dapat dilihat bahwa *device description* dan *service* berstatus tersedia *(available).* Dapat disimpulkan bahwa setelah perangkat mengirimkan *device description* dan *service* selanjutnya dapat dimanfaatkan untuk dikirimkan sebuah perintah dari *control point* untuk menjalankan fungsi yang tersedia.

6.1.3 *Control*

6.1.3.1 Tujuan Pengujian

Pengujian pada *control* adalah untuk mengetahui apakah *device* dapat merespon dan menjalankan tugas yang diberikan oleh kontrol dengan semestinya.

6.1.3.2 Prosedur Pengujian

Pada prosedur pengujian ini meliputi pengiriman dari *device* apakah kondisi lingkungan dari sensor pergerakan mendapatkan pergerakan makhluk hidup dan sensor cahaya mendapatkan cahaya atau tidak. Kemudian mengirimkan perintah dari kontrol untuk mematikan lampu yang terdapat pada perangkat sensor pergerakan.

6.1.3.3 Hasil Pengujian

Tabel 6.3 Hasil pengujian *control*

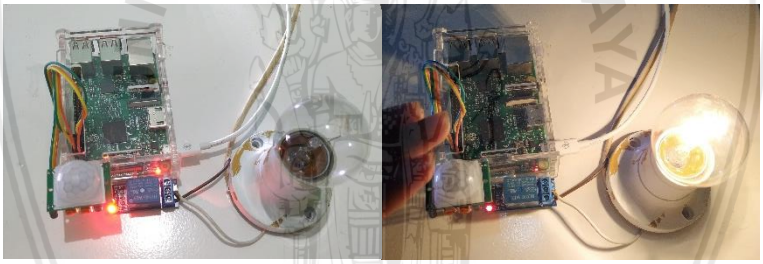
No.	Nama *Method*	Hasil Pengamatan		Kesimpulan
		Device	***Control point***	
1	*StatusPir* (perangkat sensor pergerakan)	Gambar 6.5 sampai Gambar 6.7	Gambar 6.8	Sesuai
	Keterangan : Percobaan pertama adalah dengan memberikan pergerakan dari tangan di depan sensor PIR untuk membaca nilai masukan dari sensor. Percobaan selanjutnya dengan tidak memberikan pergerakan di depan sensor. Hasil dari perangkat			

No.	Nama *Method*	Hasil Pengamatan		Kesimpulan
		Device	***Control point***	
	sensor pergerakan PIR menampilkan di *terminal* pada *device* dengan "*The light is now on*" serta lampu menyala saat perangkat sensor pergerakan menangkap adanya pergerakan dan status "*The light is now off*" untuk keadaan sensor tidak menangkap pergerakan dan lampu dalam keadaan mati. Untuk hasil yang ditampilkan pada *control point* akan memberikan status "*The light is now on*" saat sensor mendapatkan pergerakan dan status "*The light is now off*" saat sensor tidak mendapatkan pergerakan. Percobaan ketiga dengan mematikan *control point* saat *device* sedang mengirimkan nilai sensor dan hasilnya pada *device* menampilkan status "*Host Down*" pada *terminal*.			
2	*StatusLdr* (perangkat sensor cahaya)	Gambar 6.9 sampai Gambar 6.11	Gambar 6.12	Sesuai
	Keterangan : Percobaan pertama dengan memberikan cahaya di depan sensor LDR untuk membaca nilai masukan dari sensor. Percobaan selanjutnya dengan tidak memberikan cahaya di depan sensor. Hasil dari perangkat sensor cahaya LDR menampilkan di *terminal* pada *device* dengan " *The light is now on*" saat perangkat sensor cahaya menangkap adanya cahaya dan status "*The light is now off*" untuk keadaan sensor tidak menangkap cahaya. Untuk hasil yang ditampilkan pada *control point* akan memberikan status "*The light is now on*" saat sensor mendapatkan cahaya dan status "*The light is now off*" saat sensor tidak mendapatkan cahaya. Percobaan ketiga dengan mematikan *control point* saat *device* sedang mengirimkan nilai sensor dan hasilnya pada *device* menampilkan status "*Host Down*" pada *terminal*.			
3	*SwitchLed* (*control point*)	Gambar 6.13 dan Gambar 6.14	Gambar 6.15	Sesuai
	Keterangan : Percobaan dilakukan dengan mengirimkan perintah untuk mematikan lampu pada sensor pergerakan. Setelah program dijalankan dan jika muncul notifikasi berupa "*Set switch to on"* pada *control point* maka *control point* berhasil berubah kondisi lampu pada perangkat sensor pergerakan. Sedangkan pada perangkat sensor pergerakan akan muncul notifikasi *"Host switch the light to on"* dan merubah kondisi lampu.			

Tabel 6.3 menjelaskan mengenai hasil dari pengujian pada *control* yang telah dijelaskan pada prosedur pengujian. Gambar 6.6 sampai Gambar 6.16 adalah hasil pengujian yang diambil berupa *screenshot* program pada *device* dan *control point* serta pengambilan foto pada perangkat sensor pergerakan dan perangkat sensor cahaya.

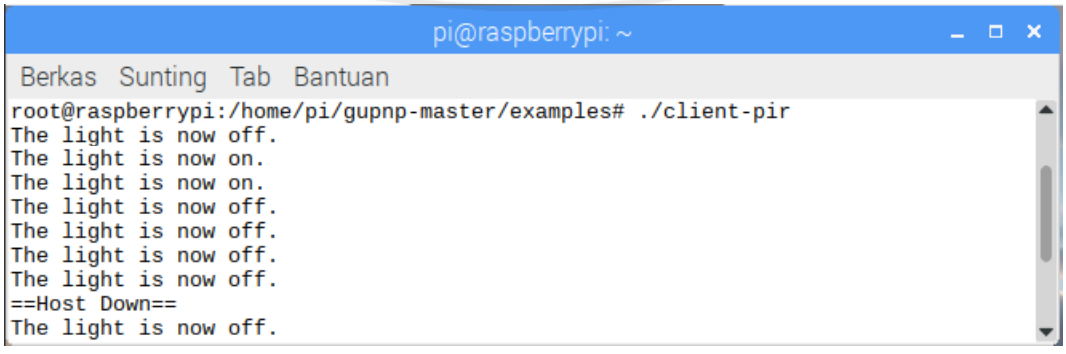
Gambar 6.5 *Screenshot* Program *Client-Pir* Perangkat Sensor Pergerakan

Pada Gambar 6.5 merupakan *screenshot* dari *terminal* yang terdapat pada perangkat sensor pergerakan. Di dalam program tersebut menampilkan kondisi pada sensor sedang menangkap dan tidak menangkap pergerakan berupa inframerah. *On* untuk kondisi sensor PIR penangkap pergerakan dan lampu menyala, sedangkan *off* adalah kondisi sebaliknya.



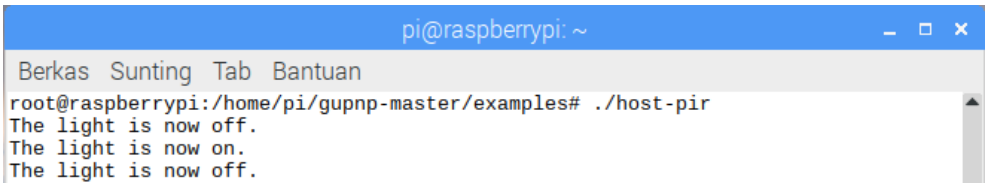
Gambar 6.6 Kondisi Perangkat Sensor Pergerakan *Off* Dan *On*

Pada Gambar 6.6 merupakan foto dari perangkat sensor pergerakan dimana sebelah kiri sensor tidak menangkap pergerakan dan lampu akan mati, sedangkan yang kanan adalah kondisi sensor menangkap pergerakan dari inframerah yang dipancarkan oleh tangan dan membuat kondisi lampu menyala.



Gambar 6.7 *Screenshot* Kondisi *Control point* Mati Pada Perangkat Sensor Pergerakan

Pada Gambar 6.7 menunjukkan notifikasi berupa *"Host Down"* dimana kondisi *control point* mati atau tidak terhubung dengan jaringan dan sensor pergerakan tetap bekerja menangkap pergerakan serta selalu mencoba terhubung kembali kepada *control point.*

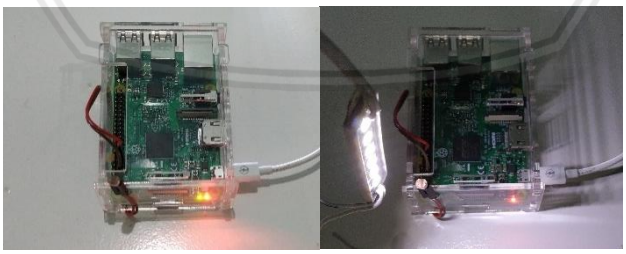


Gambar 6.8 *Screenshot* Program *Host-Pir* Pada *Control point*

Gambar 6.8 merupakan *screenshot* dari program yang terdapat pada *control point* dimana *control point* mendapatkan data atau notifikasi jika perangkat pergerakan sedang menerima pergerakan dari sensor dan tidak menangkap pergerakan dengan menampilkan notifikasi berupa *on* untuk menangkap pergerakan dan *off* untuk sebaliknya.

Gambar 6.9 *Screenshot* Program *Client-Ldr* Perangkat Sensor Cahaya

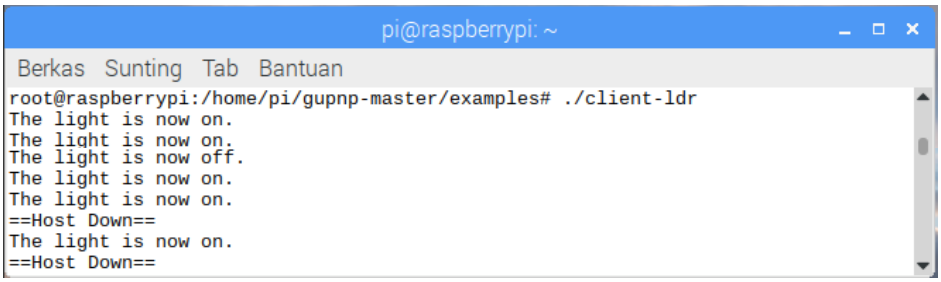
Gambar 6.9 merupakan *screenshot* dari perangkat sensor cahaya dengan menampilkan notifikasi jika sensor menangkap adanya cahaya dan sebaliknya. Perangkat sensor cahaya tersebut melakukan kinerja menangkap cahaya dengan jeda waktu setiap 2 detik.



Gambar 6.10 Kondisi Perangkat Sensor Cahaya *Off* Dan *On*

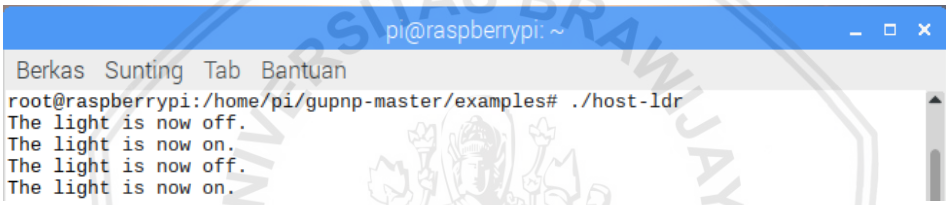
Foto dari perangkat sensor cahaya dapat dilihat pada Gambar 6.10 dimana foto sebelah kiri adalah saat sensor cahaya tidak menangkap cahaya dan notifikasi berupa *off* sedangkan foto sebelah kanan adalah saat sensor cahaya menangkap cahaya dan memberikan notifikasi berupa *on* pada tampilan program di *terminal.*

h



Gambar 6.11 *Screenshot* Kondisi *Control point* Mati Pada Perangkat Sensor Cahaya

Gambar 6.11 adalah tangkapan layar yang menunjukkan kondisi *control point* sedang mati atau tidak terhubung dalam jaringan. Sensor cahaya tetap menjalakan kinerja dengan menangkap cahaya menggunakan sensor LDR sambil mencoba terus untuk dapat kembali terhubung pada *control point.*

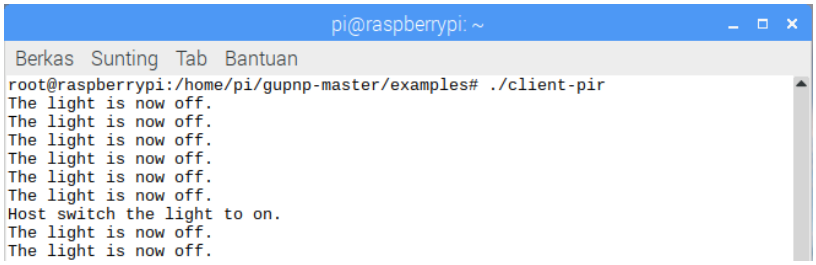


Gambar 6.12 *Screenshot* Program *Host-Ldr* Pada *Control point*

Gambar 6.12 merupakan tangkap layar dari program yang terdapat pada *control point* untuk menerima notifikasi keadaan dari perangkat sensor cahaya. Notifikasi *on* menandakan sensor cahaya menangkap cahaya dan *off* untuk kondisi sebaliknya.

Gambar 6.13 *Screenshot* Program *Switch* Pada *Control point*

Gambar 6.13 adalah tangkap layar dari program yang terdapat pada *control point* dimana berfungsi untuk menyalakan, mematikan, atau merubah kondisi lampu sebaliknya dari kondisi saat ini. Program *switch* berguna hanya pada perangkat sensor pergerakan karena hanya perangkat tersebut yang memiliki lampu.



Gambar 6.14 *Screenshot* Program *Switch* Pada Perangkat Sensor Pergerakan

Pada Gambar 6.14 menampilkan *screenshot* dari program *switch* yang terdapat pada perangkat sensor pergerakan dimana perubahan dari perintah *switch* dijalankan bersama dengan fungsi penangkapan pergerakan menggunakan sensor.

6.1.3.4 Analisis Hasil Pengujian

Dari hasil pengujian dapat dianalisis bahwa perintah dari *control point* dapat berjalan dengan baik. *Device* dapat menjalankan perintah dengan baik dengan mengirimkan kondisi sensor saat ini dan dapat mengganti kondisi lampu pada perangkat sensor pergerakan. Saat *control point* dimatikan akan memberikan notifikasi pada *device* jika *control point* mati dengan notifikasi *"Host Down".* Pengiriman data berupa notifikasi dari kondisi sensor pada perangkat akan berjalan secara terus-menerus kepada *control point.* Dengan pengiriman data secara terus-menerus pada saat sensor mengambil data dengan jeda waktu setiap 2 detik maka dapat disimpulkan bahwa pengiriman notifikasi akan memenuhi kinerja dari *device* maupun *control point* itu sendiri.

Kesimpulan akhir adalah bahwa penggunaan UPnP *control* lebih disarankan untuk tujuan mengganti kondisi lampu pada *device* daripada untuk mengirim maupun menerima status kondisi dari *device.* Untuk menyiasati dari beban yang diterima *device* dan *control point* yaitu dengan cara menambah jeda waktu pengambilan data oleh sensor atau menggunakan UPnP *eventing* dimana hanya saat pengguna menginginkan status kondisi dari *device* maka barulah status kondisi tersebut dikirimkan.

6.2 Pengujian Nonfungsional

Pengujian nonfungsional bertujuan untuk mengetahui apakah hasil implementasi sudah sama dengan kebutuhan nonfungsional. Pengujian nonfungsional meliputi waktu respon dari *device* ke *control point.*

6.2.1 Waktu Respon *Device* ke *Control point*

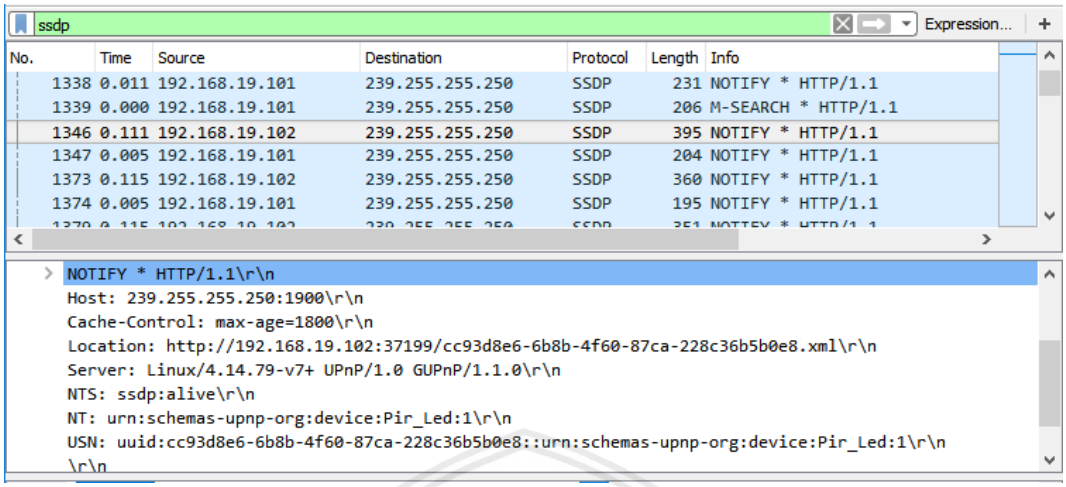
6.2.1.1 Tujuan Pengujian

Pengujian waktu respon *device* ke *control point* bertujuan untuk mengetahui apakah waktu yang dibutuhkan dalam pengirimkan sebuah data atau notifikasi apakah kurang dari batas waktu maksimal yang telah dirancang sebelumnya pada kebutuhan kinerja yaitu sebesar 5 detik.

6.2.1.2 Prosedur Pengujian

Prosedur pengujian dibagi menjadi tiga tahapan yaitu waktu respon pertama kali *device* ke *control point,* waktu respon *device* ke *control point* setelah pengenalan pertama kali, dan terakhir waktu respon pertama kali dua *device* ke *control point* secara bersamaan. Untuk pembacaan waktu respon pada *device* dan *control point* yaitu menggunakan aplikasi Wireshark. Format waktu yang digunakan pada pengujian respon waktu adalah milidetik. Pengujian berulang sebanyak 25 kali untuk mengetahui waktu rata-ratanya.

6.2.1.3 Hasil Pengujian



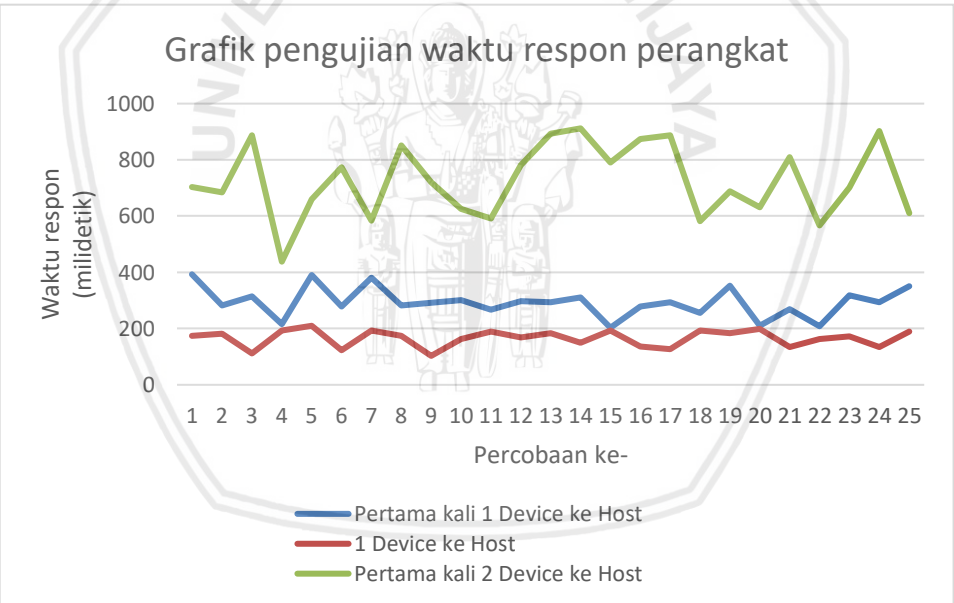
Gambar 6.15 *Screenshot* Program Wireshark

Dapat dilihat pada Gambar 6.15 merupakan tangkapan layar dari aplikasi Wireshark. Dalam aplikasi Wireshark terdapat penjelasan mengenai protokol apa yang digunakan, isi paket yang dikirimkan, dan waktu yang dibutuhkan untuk melakukan fungsi tersebut. Dapat dilihat jika no. 1346 yang memiliki waktu respon sebesar 111 milidetik pada IP 192.168.19.102 (perangkat sensor pergerakan). Cara kerja dari SSDP sendiri yaitu dengan mengirimkan paket "*device:Pir_Led:1"* kepada tujuan *(destination)* sebuah *proxy* yang telah dibuat sebelumnya pada program dengan alamat *proxy* 239.255.255.250. Lalu *control point* akan menerima paket yang telah berada pada alamat *proxy* tersebut.

Tabel 6.4 Waktu Respon *Device* Ke *Control point*

Uji Ke-	Waktu Respon (milidetik)		
	Pertama kali 1 *device* ke *control point*	1 *device* ke *control point*	Pertama kali 2 *device* ke *control point*
1	393	174	704
2	282	181	684
3	314	111	887
4	215	194	438
5	391	210	659
6	279	123	774
7	381	193	583
8	283	174	852
9	291	103	721
10	301	163	625
11	267	189	592
12	298	169	781
13	293	183	893
14	310	149	912
15	203	194	790

Uji Ke-	Waktu Respon (milidetik)		
	Pertama kali 1 *device* ke *control point*	1 *device* ke *control point*	Pertama kali 2 *device* ke *control point*
16	278	137	874
17	294	127	887
18	256	194	582
19	353	184	689
20	210	198	632
21	269	134	809
22	209	163	567
23	318	173	701
24	294	134	903
25	351	190	611
Rata-rata	**293,32**	**165,76**	**726**
	395,027		



Gambar 6.16 Grafik Waktu Respon

6.2.1.4 Analisis Hasil Pengujian

Dapat dilihat pada Tabel 6.4 merupakan hasil dari rata-rata waktu respon *device* ke *control point.* Waktu respon tercepat untuk pertama kali terhubung dari satu *device* ke *control point* adalah percobaan ke 15, dan untuk waktu respon tercepat *device* ke *control point* adalah percobaan ke 9. Untuk waktu respon rata-rata dari setiap percobaan adalah 293,32 untuk pertama kali satu *device* terhubung ke *control point,* untuk *device ke control point* sebesar 165,76 dan untuk waktu respon pertama kali dua *device* ke *control point* sebesar 726.

Dilihat pada Gambar 6.16 yaitu sebuah grafik dari hasil waktu respon. Dari tiga skenario pengujian memiliki waktu respon yang berbeda-beda. Untuk waktu respon pertama kali dari 1 *device* ke *control point* memiliki waktu respon yang lebih lama daripada waktu respon 1 *device* ke *control point* yang sudah pernah dikenali untuk pertama kali. Sedangkan untuk waktu respon pertama kali dari 2 *device* ke *control point* secara bersamaan dapat dilihat jika membutuhkan waktu respon lebih lama dari skenario pengujian yang lain dikarenakan fungsi *control point* mendapatkan deskripsi perangkat dan deskripsi layanan di dalam satu proxy dimana proxy tersebut berisikan informasi yang telah dikirimkan oleh perangkat sensor pergerakan dan perangkat sensor cahaya yang akan meberikan kinerja lebih berat yang setelahnya *control point* secara otomatis membuat komunikasi *unicast* dengan menggunakan TCP. Dengan mengubah jalur komunikasi dengan dua perangkat hampir secara bersamaan berdampak pada kinerja dan memberikan *delay* sepersekian milidetik.

Maka dapat disimpulkan bahwa rata-rata waktu respon dari setiap pengujian telah sepenuhnya berkerja dengan semestinya dan waktu respon dari tiga skenario percobaan kurang dari waktu yang telah ditentukan pada kebutuhan kinerja yaitu sebesar 5 detik, maka dari itu untuk implementasi dari perangkat *smart home* menggunakan protokol UPnP sesuai dengan rancangan yang telah dibuat.

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan dari implementasi protokol *Universal Plug and Play* (UPnP) pada sensor dan aktuator untuk otomasi lampu yang berdasarkan dari rancangan, kebutuhan, dan implementasi yaitu sebagai berikut.

1. Untuk dapat membuat protokol UPnP bekerja dimulai dengan membuat sebuah *device description* dan *service* guna mengenalkan *device* kepada jaringan. Saat pengguna mendapatkan informasi mengenai *device description* dan *service* maka *control point (host)* dapat melakukan permintaan untuk menjalankan *device* sesuai layanannya.
2. Implementasi perangkat *smart home* menggunakan sensor PIR dan *relay* untuk membuat perangkat sensor pergerakan sedangkan untuk perangkat sensor cahaya menggunakan sensor LDR sedangkan untuk pusat pemroses menggunakan raspberry pi 3. Perangkat *smart home* memanfaatkan fungsi yang terdapat pada protokol UPnP yaitu *discovery, description,* dan *control* untuk dapat terhubung dengan perangkat lain yang terdapat di dalam jaringan.
3. Dari hasil pengujian didapatkan bahwa waktu respon pertama kali rata-rata untuk satu *device* ke *control point* sebesar 293,32 ms, untuk waktu respon rata-rata satu *device* ke *control point* yang sudah dikenali sebesar 165,76 ms, dan untuk waktu respon pertama kali dari dua *device* ke *control point* secara bersamaan sebesar 726 ms.
4. Berdasarkan hasil implementasi dan pengujian, waktu respon dari satu perangkat ke *control point* untuk pertama kali lebih lama dari respon selanjutnya. Sedangkan untuk waktu respon dua *device* kepada *control point* lebih lama daripada satu perangkat ke *control point* dikarenakan jika menggunakan dua perangkat sekaligus akan membuat *control point* bekerja untuk menerima paket data dari alamat *proxy* menjadi dua kali dan akan meningkatkan kinerja dari *control point* tersebut.

7.2 Saran

Saran berfungsi untuk dapat mengembangkan penelitian ini agar lebih baik dan lebih sempurna. Untuk saran penelitian selanjutnya adalah sebagai berikut.

1. Dalam penelitian ini mengabaikan faktor keamanan jaringan, penelitian selanjutnya dapat melakukan penambahan keamanan pada jaringan.
2. Penerapan selanjutnya disarankan untuk memperluas koneksi jaringan agar dapat diakses tidak hanya dari jaringan lokal namun dapat diakses dari mana saja.
3. saran penelitian selanjutnya yaitu menambahkan fungsi *low power* agar perangkat lebih efisien dalam penggunaan daya.

DAFTAR PUSTAKA

Aditya, F.G., Hafidudin., & Permana, A.G., 2015. Analisis dan perancangan protype *smart home* dengan sistem *client server* berbasis platform android melalui komunikasi w*ireless.* S1. Universitas Telkom.

Akbar, S.R., Henryranu, B., Handono, M.T., & Basuki, A., 2015. Implementasi puwarupa perangkat rumah cerdas *pervasif* berbasis protokol *Universal Plug and Play* (UPnP) dan Rasberry pi *General Purpose Input/Output* (GPIO). S1. Universitas Brawijaya.

Bachrudin, Z., Widodo, C.E., & Adi, K., 2017. Simulator *input-output* sistem kontrol menggunakan Raspberri Pi. Universitas Diponegoro.

Georg, J., Baayen, J., Burton, R. & Ali (Khattak), Z., 2018. *Project G*UPnP. [Online] Tersedia di: <https//wiki.gnome.org/Projects?GUPnP> [Diakses 12 September 2018]

Handoko, P., 2016. *Automatic light control system based on the number of people in the room using two infrared sensor and arduino uno. R. 3.* Universitas Pembangunan Jaya.

Henderson, G., 2009. *Wiring Pi.* [Online] Tersedia di : <http://wiringpi.com/> [Diakses 12 September 2018]

Lindiawati, W., Pranoto, L.M., Waslaluddin., & Hidayat, J., 2013. Otomatisasi lampu, tirai, dan kipas angin menggunakan mikrokontroler untuk menghemat energi listrik.

Masykur, M., & Prasetiyowati, F., 2016. Aplikasi rumah pintar (*smart home*) pengendali peralatan elektronik rumah tangga berbasis web. Universitas Muhammadiyah Ponorogo.

Muchlas., Sutikno, T., & Sahnan., 2006. Sistem kendali peralatan rumah tangga berbasis HT dan mikrokontroler AT89S51. Universitas Ahmad Dahlan.

PIR (*Passive Infra Red*), 2018, *Passive Infra Red.* [Online] Tersedia di: <https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work> [Diakses Februari 1, 2018].

Pragmawati, K., 2016. Sistem kontrol peralatan elektronik rumah tangga menggunakan sms *gateway.* D3. Universitas Negeri Semarang.

Presser, A., et al., 2015. *Universal Plug and Play Device Architecture.* [Online] Tersedia di: <http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf> [Diakses 1 Februari, 2018].

Putra, G.S., 2016. Sensor gerak (*Passive Infra Red*). Universitas Sriwijaya.

Raspberry Pi, 2018, Raspberry Pi 3 model B. [Online] Tersedia di: <https://www.Raspberrypi.org/products/Raspberry-pi-3-model-b> [Diakses 1 Februari, 2018].

Relay, 2019, 1 channel *relay module (active low control)*. [Online] Tersedia di: <https://www.addicore.com/Single-Channel-Relay-Module-p/ad317.htm> [Diakses 14 mei, 2019].

Ridwanda, H., Triyanto, D., & Brianorman, Y., 2014. Sistem kendali alat listrik berbasis waktu dengan ATMEGA8535. Universitas Tanjungpura.

Saleh, M., & Haryati, M., 2017. Rancang bangun sistem keamanan rumah menggunakan *relay.* Universitas Suryadarma.

Sayuti, A., 2015. Perancangan sistem monitoring suhu menggunakan Raspberry Pi berbasis web dan android pada ruang server Universitas Darma Persada. S1. Universitas Darma Persada.

Setiawan, I., 2009. Sensor dan tranduser. Universital Diponegoro.

Sismomro, H., 2001. Konsep pengalamatan IP dengan DHCP server pada microsoft windows NT server. Jurnal Ilmiah Dasi.

Stephanie., & Sujaini, H., 2015. Sistem otomasi lampu pada bangunan publik dengan metode *forward chaining.* S1. Universitas Tanjungpura.