SISTEM PERVASIF UNTUK FITUR LAMPU RUMAH CERDAS MENGGUNAKAN *UNIVERSAL PLUG AND PLAY*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Maystya Tri Handono
NIM: 115060900111021

PROGRAM STUDI TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

SISTEM PERVASIF UNTUK FITUR LAMPU RUMAH CERDAS MENGGUNAKAN UNIVERSAL PLUG AND PLAY

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Maystya Tri Handono
NIM: 115060900111021

Skripsi ini telah diuji dan dinyatakan lulus pada
14 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizkiqa Akbar, S.T., M.Eng	Achmad Basuki, S.T., M.MG, Ph.D
NIK: 19820809 201212 1 004	NIP: 19741118 200312 1 002

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M. T
NIP: 19670801 199203 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 didalam 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, 14 Januari 2015

Maystya Tri Handono

NIM: 115060900111021

KATA PENGANTAR

Puji syukur kami ucapkan kehadirat Tuhan Yang Maha Esa atas rahmat dan hidayahNya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Sistem Pervasif untuk Fitur Lampu Rumah Cerdas".

Penulis menyadari bahwa penyusunan makalah ini tidak akan berjalan lancar tanpa berbagai masukan yang bermanfaat dari dosen pembimbing serta teman-teman penulis. Oleh karena itu penulis mengucapkan terima kasih kepada pihak-pihak tersebut yang telah bersedia untuk memberikan arahan dan saran demi kelancaran penyusunan makalah ini diantaranya:

1. Bapak dan Ibu yang saya cintai, yang telah memberikan dukungan, semangat serta do'a.
2. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng dan Bapak Achmad Basuki, ST., MMG., Ph.D selaku dosen pembimbing yang telah banyak memberikan ilmu, saran, dan motivasi untuk menyelesaikan laporan ini.
3. Bapak Adharul Muttaqin ST., MT selaku dosen penasihat akademik penulis.
4. Bapak Barlian Henryranu, S.T., M.T. selaku Kepala Laboratorium Sistem Komputer dan Robotika Program Teknologi Informasi dan Ilmu Komputer.
5. Aninditya Nugroho, S.T (Mas Didit) selaku laboran Kepala Laboratorium Sistem Komputer dan Robotika Fakultas Ilmu Komputer yang selalu membantu & menyediakan alat-alat skripsi untuk penulis.
6. Teman-teman seperjuangan skripsi labsiskombot, terima kasih telah memberikan semangat dan dukungan dalam pengerjaan skripsi ini.
7. Teman-teman RoboTIIK yang telah memberikan dukungan serta semangat dalam mengerjakan skripsi ini sampai selesai.
8. Segenap dosen dan karyawan PTIIK Universitas Brawijaya yang telah membantu pelaksanaan skripsi ini.
9. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Pada akhirnya, penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun. Penulis berharap semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi.

Malang, 1 Januari 2015

Penulis

ABSTRAK

Sistem pengontrolan lampu merupakan bagian dari rumah cerdas. Saat ini banyak metode pengontrolan lampu, akan tetapi metode tersebut akan menemui kesulitan untuk diintegrasikan ke sistem lainnya pada satu *smart home. UPnP* adalah standar teknologi untuk menghubungkan perangkat dalam jaringan dengan otomatis tanpa memerlukan konfigurasi yang rumit dari pengguna.

Penggunaan *UPnP* akan memudahkan integrasi perangkat lain yang terdapat pada *smart home,* sesuai dengan standar yang telah dikeluarkan oleh *UPnP* Forum. Untuk membuat lampu yang dapat dikontrol melalui protokol *UPnP* diperlukan *device description* dan *service description*, kedua hal tersebut diperlukan oleh pengguna untuk memanipulasi *service* yang telah disediakan. Pada skripsi ini, *UPnP device* berjalan pada *raspberry pi*, lampu dikontrol oleh raspberry pi menggunakan pin GPIO dengan memanipulasi nilai *PWM* setiap pin. Lampu yang dibuat pada skripsi ini memiliki fitur pengaturan saklar, pengaturan tingkat intensitas cahaya, dan pengaturan warna lampu.

UPnP yang dibuat pada skripsi ini menggunakan versi 1.0. Pada hasil pengujian *UPnP device validator* dan hasil filter *SSDP* menggunakan *wireshark*, *UPnP device* yang dibuat pada skripsi ini memenuhi standar *UPnP* versi 1.0. Warna lampu dapat berubah sesuai dengan nilai PWM dengan rentang nilai 0 - 255 yang diberikan pada pin GPIO. Untuk mengontrol saklar lampu, *control point* memerlukan waktu rata – rata 69,5 ms. Untuk *service dimmer* dan pengaturan warna lampu, *control point* masing-masing memerlukan waktu rata-rata 98 ms dan 85 ms.

Kata Kunci: perangkat *UPnP*, sistem rumah cerdas, Raspberry Pi

ABSTRACT

Lighting control system is part of a smart home. Recently, there are many ways of controlling the lights, but that many ways will hard to integrate into other device in a smart home. UPnP is a technology standard for connecting devices in the network automatically without requiring complicated configuration of users.

UPnP will facilitate the integration of other devices contained in the smart home, in accordance with the standards issued by the UPnP Forum. To make the lamp can be controlled via UPnP protocol required device description and the service description. In this thesis, UPnP devices running on raspberry pi, the lights controlled by the raspberry pi using GPIO pin, by manipulating the PWM value of each pin. Lamps made in this thesis have a switch setting, setting the level of light intensity and light color settings.

UPnP made in this paper uses version 1.0. The result of test UPnP device SSDP validator and wireshark data, UPnP device made in this paper meets the UPnP standard version 1.0. The color of the light can be change based on the value of PWM with a range of values 0-255 given the GPIO pin. To control the light switch, control point takes the average of time 69.5 ms. For service dimmer and color settings, control point require an average of time 98 ms and 85 ms.

Keyword: *UPnP device*, *smart home system*, Raspberry Pi

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Perkembangan teknologi saat ini telah menghasilkan berbagai aplikasi dan peralatan yang canggih dan cerdas yang mungkin akan mengubah gaya hidup manusia di masa mendatang. *Smart building* atau bangunan yang cerdas merupakan impian untuk meninggalkan interaksi dengan peralatan elektronik secara konvensional atau manual. *Smart home* atau rumah cerdas adalah sistem yang menghubungkan semua peralatan elektronik pada rumah dengan teknologi pintar atau cerdas atau otomatis. Rumah cerdas terdiri dari tiga bagian yaitu jaringan, pengontrolan, dan *otomation* (Kumar & Tiwari, 2015).

Sistem lampu yang cerdas adalah bagian yang penting pada rumah cerdas. Sistem tersebut dapat memberikan efek pencahayaan yang berbeda pada tiap ruang di dalam rumah, tidak hanya memberikan orang kualitas hidup yang lebih baik tetapi juga menyelesaikan masalah penghematan energi (Li & , 2013). Tetapi sebagian orang masih manggunakan saklar sebagai kontrol lampu, akibatnya kelalaian dari pengguna dapat menyebabkan pemborosan energi. Dengan teknologi saat ini, sudah terdapat berbagai metode untuk pengontrolan lampu secara otomatis. Metode tersebut antara lain menggunakan RF (*Radio Frequency*), Jaringan GSM (Li & , 2013), *web service* (Khan & Mouftah, 2011), dan *Bluetooth* (Cheng, et al., 2014). Dengan banyaknya metode tersebut muncul masalah baru, yaitu sulit untuk mengintegrasikan metode yang ada.

Universal Plug and Play (UPnP) merupakan teknologi yang mendukung sistem jaringan dan pendeteksian otomatis peralatan elektronik dari berbagai macam pabrikan, termasuk sistem cerdas, peralatan *wireless*, dan komputer dalam berbagai bentuk. Dengan *UPnP* peralatan elektronik dapat secara dinamis tergabung dalam jaringan, mendapatkan alamat IP, menyatakan kemampuannya, dan mempelajari kehadiran dan kemampuan peralatan lain secara otomatis (Yiqin, et al., 2009). Sistem pencahayaan mempunyai peranan penting di bangunan modern, dikarenakan kondisi pencahayaan mempengaruhi kenyamanan visual seseorang (Baniya, et al., 2014). Fitur yang penting untuk dimiliki sistem pencahayaan adalah kendali status lampu, fungsi untuk mengatur tingkat kecerahan cahaya, dan fitur untuk mengubah warna pencahayaan (Higuera, et al., 2015).

Berdasarkan latar belakang tersebut, penelitian ini merancang sistem pengontrolan lampu menggunakan Universal Plug and Play (*UPnP*). *UPnP* mampu mengintegrasikan metode-metode pengontrolan lampu dan sistem pencahayaan dengan standar yang telah dibuat oleh *UPnP* Forum.

1.2 Rumusan masalah

Berdasarkan latar belakang di atas, dapat dirumuskan beberapa masalah sebagai berikut:

1. Bagaimana langkah-langkah implementasi protokol *UPnP* pada fitur lampu pada rumah cerdas.
2. Bagaimana implementasi fitur saklar, *dimmer,* dan perubahan warna pada sistem lampu rumah cerdas.
3. Bagaimana waktu respon sistem ketika mendapatkan input dari pengguna.

1.3 Tujuan

Tujuan dari skripsi ini yaitu mengimplementasikan protokol *UPnP* untuk mengatur saklar, *dimmer* dan perubahan warna lampu pada rumah cerdas serta untuk mengetahui respon sistem ketika mendapatkan input dari pengguna.

1.4 Manfaat

Manfaat yang didapat dari sistem ini adalah dengan implementasi protokol *UPnP* pada rumah cerdas dapat dipergunakan sebagai contoh pengenalan sistem secara pervasif tanpa melibatkan konfigurasi detail dari pengguna. Serta implementasi *UPnP* pada rumah cerdas memudahkan *device* yang memiliki fitur lain dapat berkomunikasi dengan mudah, hal ini sesuai dengan arah penelitian *internet of things* yang meramalkan akan banyak *smart device* dari berbagai vendor.

1.5 Batasan masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka penelitian ini dibatasi dalam hal:

1. Penelitian ini mengabaikan aspek keamanan jaringan.
2. Sistem lampu yang dirancang tidak menggunakan *Auto IP*.
3. Standar *UPnP* yang dipakai adalah versi 1.0.
4. Pengontrolan lampu menggunakan *Universal UPnP Controller*.
5. Fitur lampu yang dikontrol adalah saklar, tingkat intensitas cahaya, dan perubahan warna cahaya.
6. Sistem berjalan pada jaringan lokal.
7. *UPnP device* tidak mengimplementasikan fitur *presentation*.

1.6 Sistematika pembahasan

Sistematika penulisan penelitian ditunjukkan untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, sebagai berikut.

BAB I **Pendahuluan**

Menguraikan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan.

BAB II **Landasan Kepustakaan**

Menguraikan kajian pustaka dan dasar teori yang mendasari teknologi *UPnP*, sistem pengontrolan lampu yang sekarang ada.

BAB III **Metodologi**

Menguraikan tentang metode dan langkah kerja yang terdiri dari studi literatur, analisis kebutuhan simulasi, perancangan sistem, implementasi dan analisis serta pengambilan kesimpulan.

BAB IV **Perancangan dan Implementasi**

Menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem.

BAB V **Pembahasan**

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan

BAB VI **Penutup**

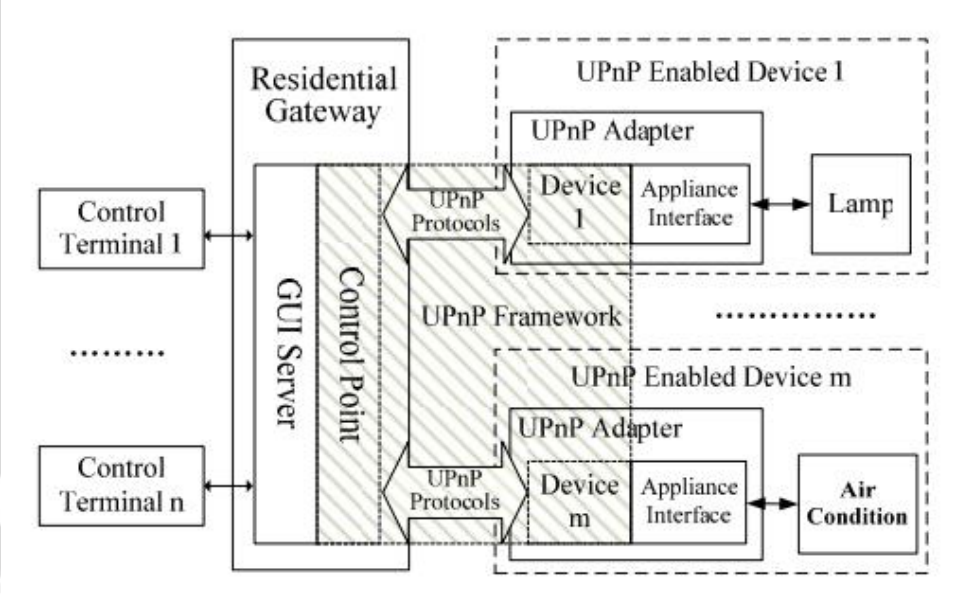
Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian program, serta saran-saran untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tinjauan pustaka yang meliputi tinjauan pustaka dan dasar teori yang diperlukan untuk penelitian. Tinjauan pustaka membahas penelitian yang telah ada dan yang diusulkan. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan.

2.1 Tinjauan Pustaka

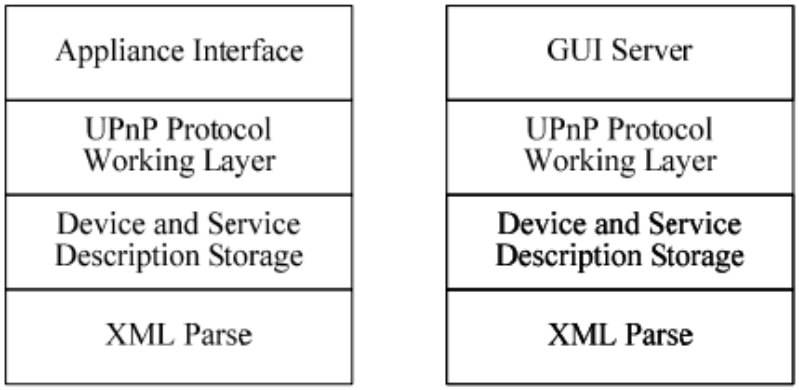
Sebuah jurnal internasional yang berjudul *"Home Networking and Control based on UPnP: An Implementation"* oleh Lu Yiqin pada tahun 2009, berisikan tentang bagaimana merancang sebuah sistem *UPnP* pada sistem *home network.* Pada jurnal tersebut peneliti merancang alat yaitu *UPnP Adapter*, alat tersebut dibuat dengan tujuan untuk menerjemahkan peralatan rumah yang belum mendukung *UPnP* dapat diterjemahkan ke dalam protokol *UPnP* menggunakan *UPnP control message*. *UPnP Adapter* dibangun menggunakan *framework software* dari Intel SDK.



Gambar 2.1 Implementasi *UPnP* Adapter

Sumber: Yiqin, Fang, & Wei Liu (2009)

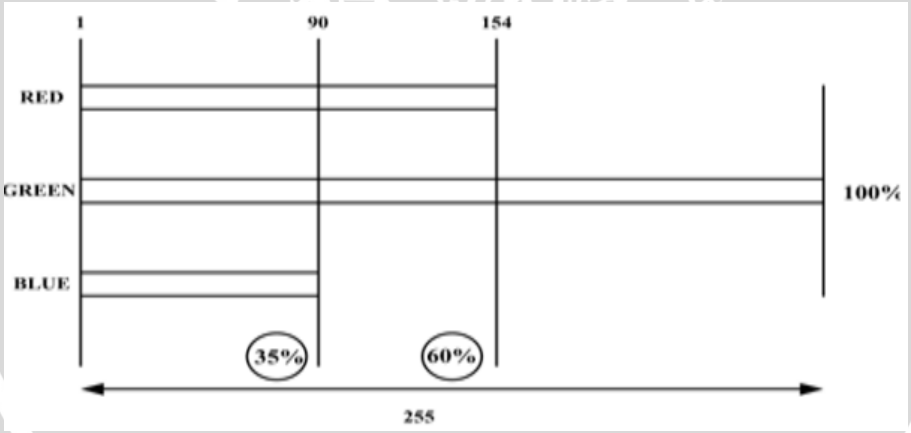
Gambar 2.1 menunjukkan arsitektur *UPnP Adapter.* Dalam arsitektur tersebut terdapat empat komponen fisik, yaitu: *control terminal, residential gateway, UPnP adapter,* dan perangkat elektronik yang terdapat pada rumah. Pada penelitian ini, *control point* yang dipakai adalah *residential gateway* dan *UPnP adapter* bertindak sebagai *UPnP device*. Gambar 2.2 menunjukkan struktur perangkat lunak *UPnP adapter*. Untuk menerjemahkan dokumen XML yang dipakai untuk *device description* dan *service description,* mereka menggunakan XML API dari Intel SDK dan menyimpannya sebagai *structure* melalui STL (*Standard Template Library).*



Gambar 2.2 Struktur Perangkat Lunak *UPnP Device*.

Sumber: Yiqin, Fang, & Wei Liu (2009)

Sumber jurnal lain yang berjudul *"Development of Wireless RGB LED Dimming Control Technology Using SmartPhone"* oleh Yu-Shan Cheng pada tahun 2014 membahas tentang teknik pengontrolan LED RGB. Untuk dapat mengatur tingkat kecerahan *(dimming)* LED, peneliti pada jurnal ini menggunakan *digital signal controller (DSC)* dsPIC30F2020 dan menggunakan metode perbandingan waktu. Pada Gambar 2.3 metode perbandingan waktu digunakan untuk mengatur tingkat intensitas LED. Nilai *dimming* akan dibandingakan dengan waktu untuk mengatur tingkat intensitas cahaya .



Gambar 2.3 Metode pengatur cahaya led

Sumber: Cheng, Chen, Liu, & Wang (2014)

2.2 Dasar Teori

Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan. Dasar teori akan dibahas disubbab 2.2.1 sampai 2.2.3.

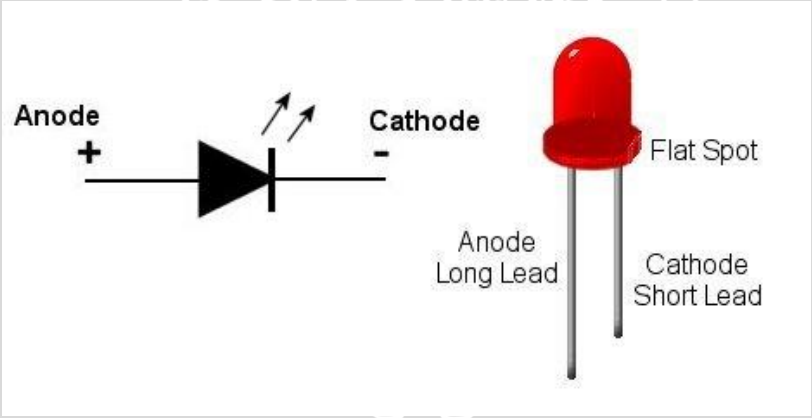
2.2.1 *Light-emitting Diode* (LED)

Dioda cahaya atau lebih dikenal dengan sebutan LED (*light-emitting diode*) adalah suatu *semikonduktor* yang memancarkan cahaya *monokromatik* yang

tidak *koheren* ketika diberi tegangan maju. Chip LED mempunyai kutub positif dan negative (p-n) dan hanya akan menyala bila diberi tegangan arus maju. Ini dikeranakan LED terbuat dari bahan *semikonduktor* yang hanya akan mengizinkan arus listrik mengalir ke satu arah dan tidak ke arah sebaliknya. Bila LED diberi arus terbalik, hanya akan ada sedikit arus yang melewati chip LED. Ini menyebabkan chip LED tidak akan mengeluarkan emisi cahaya.

Gambar 2.4 Ragam warna LED

Sumber: http://www.rayennur.com (2015)



Gambar 2.5 Bentuk dan Simbol LED

Sumber: www.mainbyte.com (2015)

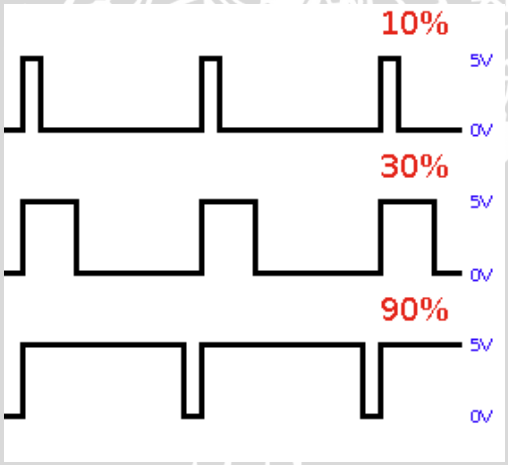
Salah satu contoh LED adalah RGB LED yang terdiri dari 3 buah LED. Masing-masing LED mempunyai satu warna yaitu satu merah, satu hijau, dan satu biru. Tiga buah warna LED tersebut mampu memproduksi sekitar 16 juta warna.

Gambar 2.6 RGB LED

Sumber: Wikipedia (2015)

2.2.2 *Pulse Width Modulation* (PWM)

Pulse Width Modulation (PWM) secara umum adalah sebuah cara memanipulasi lebar sinyal yang dinyatakan dengan pulsa dalam suatu perioda, untuk mendapatkan tegangan rata-rata yang berbeda. Beberapa contoh aplikasi PWM adalah pemodulasian data untuk telekomunikasi, pengontrolan daya atau tegangan yang masuk ke beban, regulator tegangan, *audio effect* dan penguatan, serta aplikasi-aplikasi lainnya. Aplikasi PWM berbasis mikrokontroler biasanya berupa pengendalian kecepatan motor DC, pengendalian servo motor, pengaturan nyala terang LED (Marzuki, 2013).



Gambar 2.7 Ilustrasi pengaturan PWM

Pada Gambar 2.7, semakin rapat periode antar pulsa, maka frekuensi yang dihasilkan semakin tinggi. PWM mempunyai dua jenis dalam pembangkitan sinyalnya yaitu secara analog dan digital. Pada metode digital setiap perubahan PWM dipengaruhi oleh resolusi dari PWM itu sendiri. Misalkan PWM digital 8 bit berarti PWM tersebut memiliki resolusi 2^8 = 256, maksudnya nilai keluaran PWM ini memiliki 256 variasi, variasinya mulai dari 0-255 yang mewakili *duty-cycle* 0%-100% dari keluaran PWM tersebut.

2.2.3 *Universal Plug and Play* (*UPnP*)

Arsitektur *UPnP* didesain untuk menghubungkan perangkat yang terhubung dalam jaringan, dan mendefisinikan set standar baku untuk semua perangkat agar dapat mendeskripsikan spesifikasinya dan layanan yang diberikan. *UPnP* menggunakan standar yang ada seperti *IP, TCP, UDP, HTTP* dan *XML*. Seperti pada internet, kontak didasarkan pada *XML* yang telah dideklarasikan dan komunikasi melalui *HTTP*. Teknologi *UPnP* dapat berjalan hampir di semua sistem operasi yang ada dan bekerja dengan baik hampir di semua jenis tipe fisik media komunikasi. Teknologi *UPnP* menyediakan berbagai macam fitur, antara lain (Jeronimo & Weast , 2003):

1. *Device Connectivity*, arsitektur mengatur protokol setiap perangkat untuk berkomunikasi dengan perangkat lainnya.
2. *Ad-Hoc Networking*, perangkat *UPnP* dapat bergabung secara dinamik, tanpa melalui infrastruktur khusus untuk mengaturnya. Jaringan *ad-hoc* terbuat secara langsung tanpa perlu konfigurasi manual.
3. *Zero-Configuration Network*, arsitektur *UPnP* mendukung *zero configuration networking* dimana pengguna tidak perlu mengatur konfigurasi jaringan pada perangkat.
4. *Standart Based Architecture,* arsitektur *UPnP* didasarkan pada *open standart*.
5. *Platform Independence,* arsitektur *UPnP* adalah suatu set protokol bukan suatu API.
6. *Media and Device Independende*, *UPnP* mampu berjalan di medium apapun selama terdapat IP.
7. *Programmatic and Manual Device Control*, arsitektur *UPnP* mengizinkan suatu aplikasi yang terprogram untuk mengatur perangkat dalam *home network*.

Keuntungan dari arsitektur *UPnP* yaitu (Jeronimo & Weast , 2003):

1. *UPnP* dapat berjalan di berbagai macam lingkungan jaringan.
2. *UPnP* dapat diimplementasikan pada semua software platform dengan berbagai macam bahasa pemrograman.

3. *UPnP* dibuat dari berbagai macam teknologi yang terkenal seperti *IP*, *TCP, UDP, HTTP* dan *XML*.
4. Fungsi dan layanan yang terdapat pada *UPnP* dapat dengan mudah dimodifikasi.

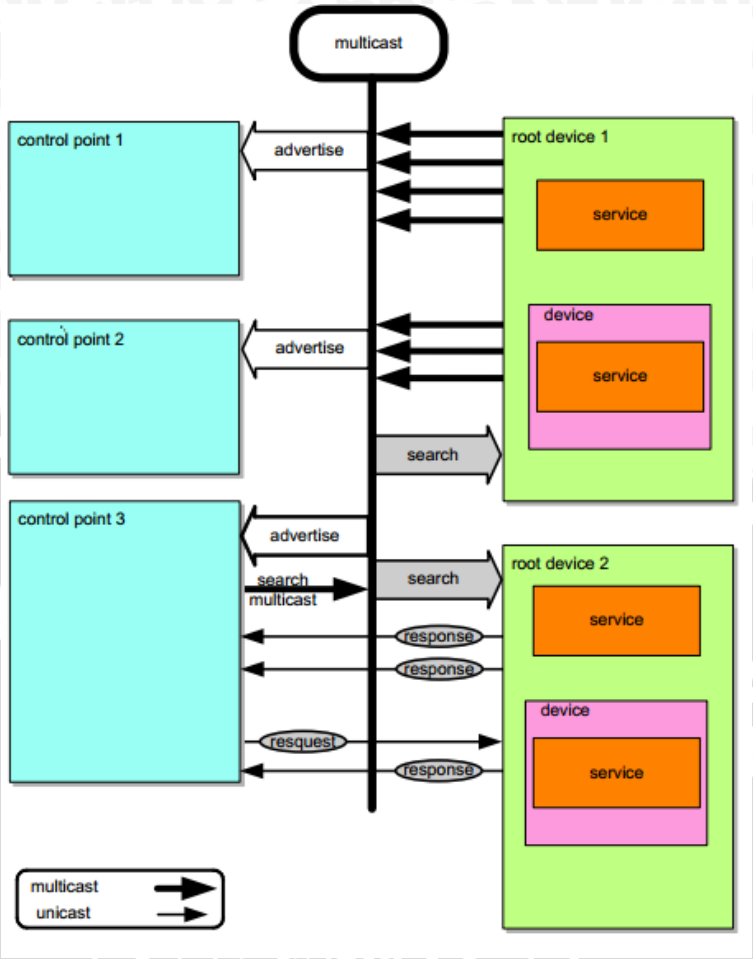
Komponen utama dari *UPnP* adalah satu *control point* dan satu atau lebih perangkat. *Control point* mengenali dan mengontrol perangkat untuk dapat menjalankan layanan yang tersedia dalam perangkat tersebut. Setiap perangkat mengizinkan *control point* untuk mengenalinya, dan menerima pesan kontrol yang dikirim oleh *control point*. Operasi *UPnP* terdiri dari lima langkah, yaitu *discovery, description, control, event,* dan presentasi, kelima langkah tersebut dapat dijelaskan sebagai berikut:

1. Discovery

   Ketika perangkat pertama kali terhubung ke dalam jaringan, d*iscovery* mengizinkan perangkat untuk mengumumkan layanan yang terdapat di dalamnya kepada *control point.* Untuk *control point* ketika terhubung ke dalam jaringan, protokol *discovery UPnP* membuat *control point* mampu untuk menemukan perangkat yang terdapat pada jaringan. Untuk dapat melakukan hal tersebut, semua *UPnP device* mempunyai perilaku sebagai berikut:

   - Ketika *device* baru muncul di dalam jaringan, *device* tersebut harus melakukan *multicast* beberapa paket *discovery message* untuk mengumumkan *service* yang dimiliki.
   - *Device* harus selalu mendengarkan ke alamat *multicast* untuk *discovery message*. Setelah menerima *discovery message*, *device* mengirim sebuah pesan respon kepada pengirim.
   - Ketika *device* menghapus keberadaannya, *device* tersebut harus mengirimkan *discovery message* yang memberitahukan bahwa *service* sudah tidak tersedia.

   Untuk *control point* harus memiliki kriteria yaitu ketika *control point* ditambahkan ke dalam jaringan, *control point* harus mengirimkan sebuah pesan *discovery* secara *multicast* untuk mencari *device* dan *service* yang sesuai dengan yang diinginkan. Terdapat dua jenis *discovery message*, yaitu *advertisement* dan *search*. Pesan *search* dikirimkan oleh *control point* sedangkan pesan *advertisement* dikirimkan oleh *control point* dan *device*. *Discovery message* dikirimkan menggunakan protokol *SSDP*. SSDP didesain sebagai solusi yang mudah untuk sumber daya yang didasarkan HTTP dalam jaringan lokal tanpa perlu konfigurasi, manajemen, dan administrasi (Forum, 2008). Gambar 2.8 merupakan proses *discovery* yang dilakukan oleh *control point* dan *device.*



Gambar 2.8 Arsitektur Discovery

Sumber: *UPnP* Device Architecture v2.0 (2014)

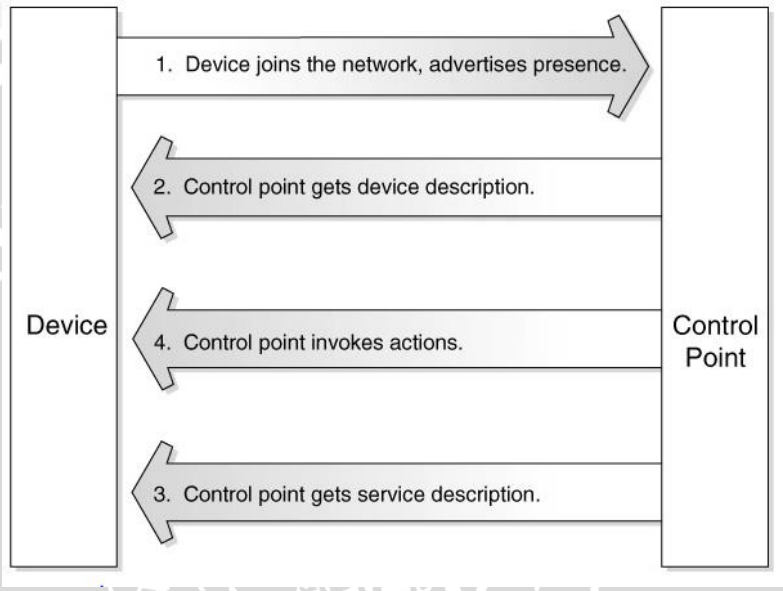
2. *Description*

Description adalah penghubung antara *discovery* dan *control*. Pada *discovery* perangkat mengumumkan keberadaannya ke *control point,* sedangkan *control point* mencari perangkat. Setelah *control point* menemukan perangkat yang sesuai, *control point* akan menerima dokumen deskripsi perangkat dan layanan yang tersedia. Setelah *control point* menerima dan mengerti dokumen yang diberikan perangkat, maka *control point* mampu untuk mengontrol perangkat (Forum, 2008). Dokumen *description* ditulis dengan format XML. Dokumen *description* pada *UPnP* dibagi dua bagian, yaitu:

- *Device description* adalah sebuah dokumen yang berisikan:
  - Nama *vendor* pembuat, informasi *manufacture*(nomor seri, nama pebrik pembuat dll)
  - Daftar *service* yang terdapat pada *device* tersebut. Setiap element dalam daftar tersebut mempunyai tipe *service*,

sebuah nama, sebuah alamat *URL* ke *service description*, sebuah *URL* untuk mengontrol dan sebuah *URL* untuk melakukan *eventing.*

- *Service description* adalah dokumen yang berisikan:
  - Daftar *action* yang tersedia serta argumen dari *action* tersebut.
  - Daftar dari variabel yang tersedia dan tipe datanya, ukuran nilainya, serta karakteristik *event*



Gambar 2.9 Proses pengiriman dan penerimaan dokumen deskripsi

Sumber: Michael Jeronimo (2003)

3. Control

   Protokol yang digunakan antara perangkat dengan *control point* adalah *Simple Object Access Protocol (SOAP)*. Pengontrolan perangkat dilakukan setelah *control point* menerima deskripsi perangkat, *control point* akan mengirim pesan kontrol ke URL device (Forum, 2008).

4. Event

   Layanan pada perangkat harus dapat memperbarui status keadaannya, perangkat membuat *event notification* yang terbuat dari *General Event Notification Architecture (GENA)* (Forum, 2008).

5. Presentasi

   Presentasi digunakan untuk memudahkan pengguna dalam pengontrolan perangkat. Setiap perangkat menyediakan alamat (URL) presentasinya. Pengguna dapat mengakses keadaan perangkat dari

URL tersebut. Operasi pengontrolan ini dapat dilakukan menggunakan metode HTTP GET (Jeronimo & Weast , 2003).

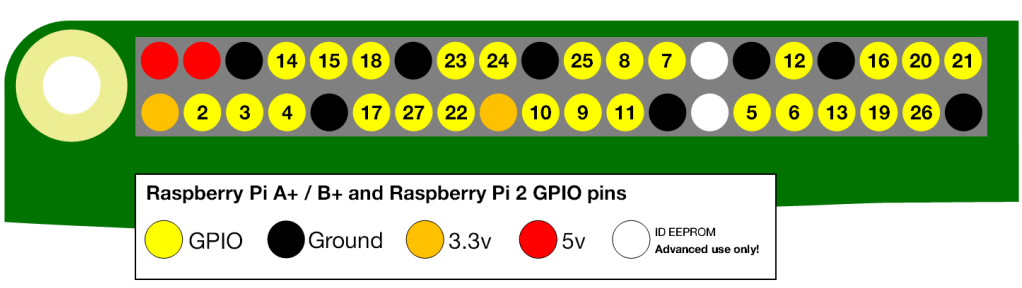
2.2.4 Raspberry Pi

Raspberry pi adalah sebuah perangkat *single-board* komputer memiliki besar seukuran kartu kredit, dikembangkan di Inggris oleh *Raspberry Pi Foundation*. Pengembangan *raspberry pi* bertujuan sebagai bentuk pembelajaran ilmu komputer dasar pada sekolah (Sandeep, et al., 2015).

Gambar 2.10 Bentuk mini komputer Raspberry Pi 2

Sumber: Persson (2015)

Raspberry Pi 2 merupakan generasi kedua dari *Raspberry Pi, Raspberry Pi 2* memiliki gambar fisik ditujukkan oleh gambar 2.10. Dibandingkan dengan seri sebelumnya yaitu *Raspberry Pi 2 Model B+*, generasi kedua memiliki kelebihan yaitu memakai CPU ARM Cortex-A7 dengan 900MHz dan memiliki kapasitas RAM sebesar 1GB. Salah satu fitur yang terdapat pada *Raspberry Pi* adalah *pin GPIO(General Purpose Input/Output).*



Gambar 2.11 *Pin GPIO Raspberry Pi 2*

Sumber: Raspberry Pi(2015)

Gambar 2.11 menunjukkan penomoran pin pada Raspberry Pi 2. *Pin-pin* tersebut merupakan penghubung *Raspberry Pi* dengan dunia fisik. Dalam

Raspberry Pi terdapat 40 *pin*, 26 adalah *pin GPIO* dan sisanya adalah sumber tegangan atau *ground* (Pi, 2015).

2.2.5 *Library GUPnP*

Untuk mempercepat pembuatan suatu aplikasi, *developer* menggunakan *library.* Pada skripsi ini, *GUPnP* digunakan untuk mempermudah pembuatan *UPnP device*. *GUPnP* merupakan *open source framework* yang ditulis memakai bahasa C menggunakan *GObject dan libsoup*. Fitur utama pada *GUPnP* adalah sebagai berikut (Georg, et al., 2015):

- Mengimplementasikan *GSSDP* sebagai sumber daya untuk *discovery* dan *announcement* melalui *SSDP*.
- *GUPnP* mengimplementasikan spesifikasi *UPnP*, seperti: *announcement* dan *discovery*, *description, control, event notification* dan *presentation.*
- Pada *GUPnP-AV* banyak koleksi *helpers* untuk membuat aplikasi *AV(audio/video).*

Library GUPnP masih terus berkembang dan mempunyai *update* setiap muncul *libsoup* versi terbaru.

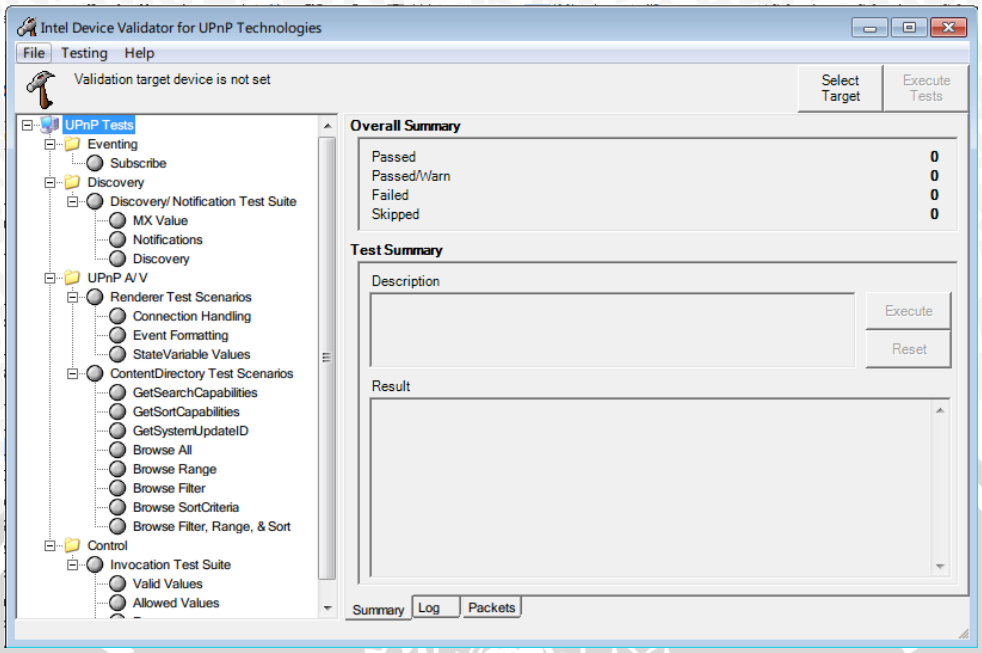
2.2.6 *Library WiringPi*

WiringPi adalah *library* yang digunakan untuk mengakses pin GPIO pada BCM2835 yang digunakan oleh *Raspberry Pi*. *Library wiringpi* ditulis dengan bahasa C dan memiliki lisensi *GNU LGPLv3. Wiringpi* memiliki banyak fitur yang digunakan untuk mengatur *pin GPIO*, antara lain (Henderson, 2009):

- Pengaturan PWM
- Memiliki *Serial library*
- Memiliki I2C *library*
- Memiliki SPI *library*
- Memiliki Shift *library*

Pada skripsi ini, *wiringpi* digunakan untuk mengatur PWM pada *pin GPIO Raspberry Pi.*

2.2.7 *UPnP Device Validator*



Gambar 2.12 *UPnP Device Validator*

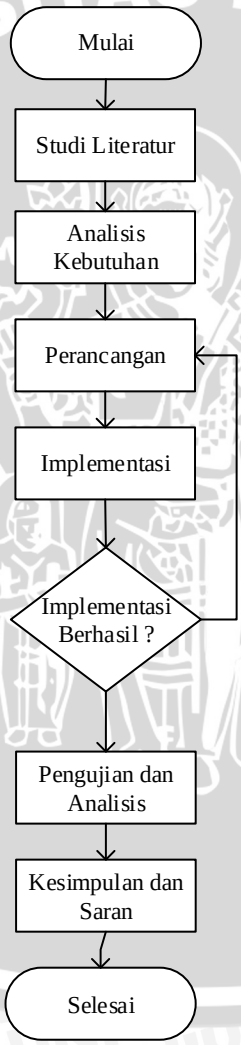
UPnP device validator adalah perangkat lunak yang dibuat oleh Intel, tujuan perangkat lunak ini adalah untuk membantu menjamin bahwa *UPnP device* yang dibuat telah memenuhi standar *UPnP Device Architecture v1.0*. Perangkat lunak ini juga menguji tingkat respon dari *UPnP device,* yaitu dengan melakukan banyak *subscription* dan atau *action invocation*. *UPnP device validator* memiliki dua panel utama dalam tampilannya, ditujukkan oleh Gambar 2.12. Pada panel kiri mempunyai struktur *tree* yang berisi macam-macam pengujian untuk *UPnP device.* Untuk melakukan pengujian *UPnP device*, terlebih dahulu kita memilih *device* yang akan diuji dengan memilih pada tombol *"Select Target"*, kemudian klik tombol *"Execute Tests"* untuk memulai pengujian. Panel sisi kanan menunjukkan statistik hasil pengujian. Jika pengujian berhasil maka titik warna abu-abu pada panel kiri akan menjadi warna merah, namun jika gagal akan berwarna merah.

BAB 3 METODOLOGI

Bab ini akan berisi serta menjelaskan metode apa yang akan digunakan pada penelitian. Selain hal tersebut dalam bab ini akan menjabarkan tujuan dan cara dari tiap - tiap langkah yang akan dilakukan dalam penelitian.

3.1 Metodologi Penelitian

Dalam metode penelitian akan menjabarkan tujuan serta tata cara langkah dalam penelitian. Beberapa hal yang akan dibahas didalamnya antara lain meliputi studi literatur, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis, kesimpulan dan saran. Langkah-langkah penelitian ini akan berjalan seperti diagram alir pada Gambar 3.1.



Gambar 3.1 Flowchart Runtutan Pengerjaan Penelitian

3.1.1 Studi Literatur

Studi literatur mempelajari mengenai penjelasan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut diperoleh

dari buku, jurnal, *e-book*, dokumentasi, dan penelitian sebelumnya yang berkaitan tentang topik tugas akhir ini. Referensi utama yang diperlukan dalam penulisan ini adalah *forum UPnP*, protokol SSDP, dan *wiringPi*.

3.1.2 Analisa Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan oleh sistem yang akan dibangun dan diuji. Analisis kebutuhan dilakukan dengan mengidentifikasi kebutuhan sistem dan siapa saja yang terlibat didalamnya. Dalam kebutuhan sistem akan terjadi proses pengindentifikasi perangkat yang digunakan seperti perangkat keras dan perangkat lunak. Dengan adanya pengindetifikasi maka nantinya dapat membantu serta mempermudah dalam mendesain hingga membuat sistem.

3.1.2.1 Kebutuhan Perangkat Keras

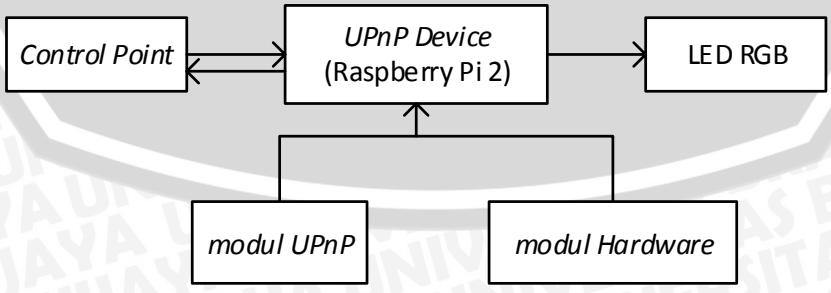
Sistem pervasif untuk mengontrol lampu menggunakan *UPnP* akan dibangun menggunakan Raspberry Pi 2 sebagai *UPnP device*. Lampu yang akan digunakan disimulasikan dengan LED RGB.

3.1.2.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk membangun sistem pervasive untuk mengontrol lampu mengunakan *UPnP* adalah perangkat lunak yang bersifat *free* dan *open source*. Perangkat lunak yang dibutuhkan antara lain *Universal Control Point*, pustaka wiringPi, dan pustakan *gupnp*. Perangkat lunak yang berupa pustaka akan dipasang pada Rasberrypi 2 yang bertindak sebagai *UPnP device*. *Universal Control Point* akan dipasang pada perangkat lain seperti laptop atau PC yang nantinya akan melakukan proses pengontrolan terhadap *device.*

3.1.3 Perancangan Sistem

Setalah dilakukan analisis kebutuhan sistem, maka langkah selanjutnya yaitu perancangan sistem. Untuk memudahkan pemahaman peracangan sistem secara keseluruhan dapat dijelaskan menggunakan diagram blok. Diagram blok sistem pada penelitian ini menjelaskan desain sistem keseluruhan. Yang digambarkan sebagai berikut:



Gambar 3.2 Diagram blok sistem

Setiap diagram yang terdiri dari beberapa fungsi akan disatukan menjadi diagram blok sistem pada penelitian ini. Dari diagram tersebut akan dijelaskan

lebih dalam lagi menggunakan *flowchart* untuk mengetahui tahapan aktivitas proses dalam sistem ini.

3.1.4 Implementasi

Implementasi sistem ini akan dilakukan sesuai dengan perancangan sistem yang telah dibuat sebelumnya. Pada bagian ini terdapat beberapa proses implementasi yaitu, implementasi *UPnP device*, implementasi *UPnP service* dan implementasi *modul* program untuk mengendalikan lampu. Impelementasi *UPnP device* menggunakan pustakan g*UPnP* versi 0.20.12 dan menggunakan bahasa pemrograman C. Untuk perancangan *device description* dan *service description* digunakan bahasa XML sesuai standar pembutan dari *UPnP* Forum. Kemudian untuk implementasi program pengendalian lampu menggunakan pustaka wiringPi dan menggunakan bahasa pemrograman C.

3.1.5 Pengujian

Pengujian pada skripsi ini dilakukan agar dapat menunjukkan bahwa sistem telah mampu bekerja dengan baik sesuai spesifikasi kebutuhan yang melandasinya. Terdapat tiga pengujian yang dilakukan dalam penelitian ini, yaitu:

1. Pengujian pengaruh PWM terhadap perubahan warna LED.
2. Pengujian validasi *UPnP device* menggunakan *Device Validator* milik Intel
3. Pengujian sistem secara keseluruhan menggunakan *control point*.

3.1.6 Analisis

Untuk mengukur kinerja dari sistem pengontrolan lampu menggunakan *UPnP*, dilakukan analisis untuk mengetahui hasil yang akan digunakan untuk menarik kesimpulan dari penelitian yang dilakukan. Hasil analisa digunakan untuk mengetahui kelayakan dari sistem yang telah dibuat.

3.1.7 Kesimpulan

Pengambilan hasil dilakukan setelah semua tahapan perancangan, implementasi, dan pengujian sistem telah dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahap akhir pada penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta memberikan pertimbangan atas hasil yang telah dilakukan.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

Dalam bab ini akan membahas tentang peracangan, analisis kebutuhan dan pengujian, serta implementasi dari kontrol lampu menggunakan *UPnP* sehingga dapat menunjukkan bahwa sistem dapat diimplementasikan dengan baik.

4.1 Perancangan

Tahap perancangan akan memfokuskan pembahasan tentang perancangan dari sistem dengan tujuan untuk memudahkan dalam proses implementasi sistem secara keseluruhan. Dalam peracangan sistem dibagi menjadi beberapa tahap yang harus dilakukan, diantaranya:

4.1.1 Analisis Kebutuhan Sistem

Kebutuhan sistem dalam skripsi ini adalah lingkungan implementasi untuk *UPnP device* dan *control point*. Lingkungan implementasi diperlukan agar sistem dapat berjalan dengan baik. Analisis kebutuhan sistem tersebut adalah:

Pada *UPnP device*:

1. Sistem dapat mengkonfigurasi IP dan *port* yang digunakan untuk *host server UPnP*.
2. Sistem dapat mengumumkan keberadaan dirinya pada jaringan.
3. Sistem dapat melakukan aksi pengontrolan fitur lampu berdasarkan *request* dari *control point.*

Pada *control point*:

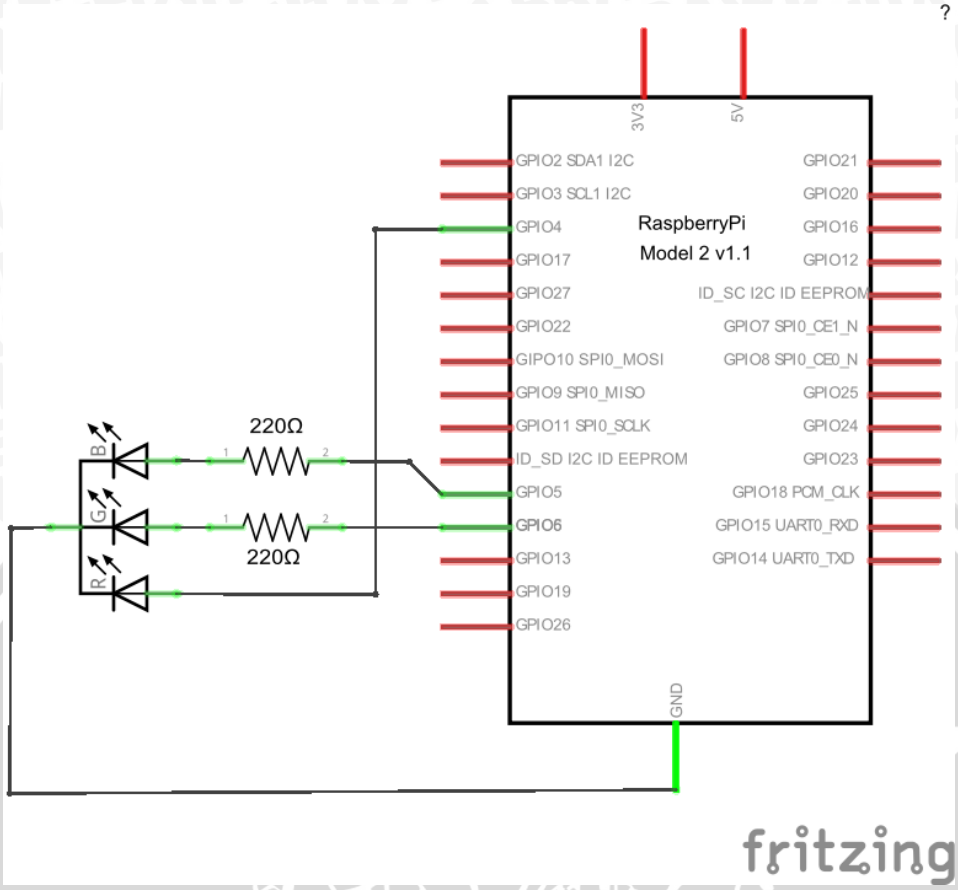
1. Sistem dapat mengenali perangkat *UPnP* pada jaringan.
2. Sistem dapat melakukan *subscribe* pada *UPnP* yang ditemukan.
3. Sistem dapat melukukan pengontrolan *service* yang tersedia pada *device*.
4. Sistem dapat menerima notifikasi *event* dari *UPnP device.*

4.1.2 Perancangan

Tahap perancangan sistem bertujuan untuk memudahkan pengimplementasian sistem. Tahap peracancangan sistem dibagi menjadi dua bagian, yaitu peracangan perangkat keras dan peracangan perangkat lunak.

4.1.2.1 Peracangan Perangkat Keras

Sistem yang akan dirancang pada penelitian ini, akan diimplementasikan pada rangkaian komponen fisik yang sesuai dengan gambar berikut:

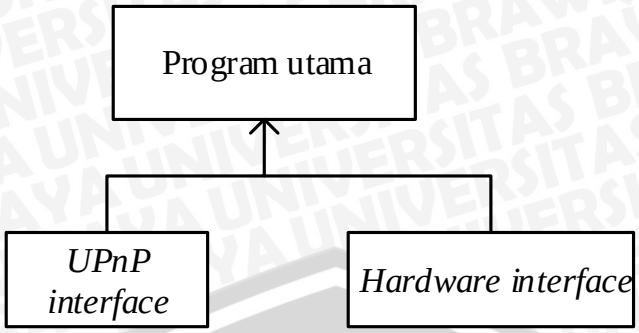


Gambar 4.1 Rancangan Perangkat Keras

Pada komponen perangkat keras, raspberry pi digunakan sebagai *interface* untuk mengendalikan lampu atau LED. LED dihubungkan ke raspberry pi menggunakan *pin GPIO* yang terdapat pada raspberry pi. Pin yang dipakai pada gambar adalah *pin* 4 untuk warna merah, *pin* 5 untuk warna biru, dan *pin 5* untuk warna hijau. Resistor digunakan ntuk membatasi tegangan yang dilewatkan ke LED, pin GPIO raspberry pi memiliki *output* 3,3 V. Pin LED warna hijau dan biru membutuhkan tegangan 2.66 V sehingga resistor yang dipakai bernilai 220 Ω (Leach, 2013).

4.1.2.2 Perancangan Perangkat Lunak

Perangkat lunak yang dirancang pada sistem ini terbagai menjadi *hardware modul*, *UPnP modul* dan program utama. Untuk memudahkan implementasi terlebih dahulu dibentuk diagram blok sistem berdasarkan analisis kebutuhan.

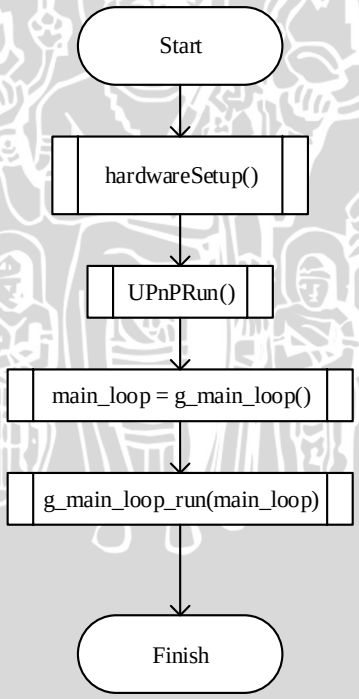


Gambar 4.2 Arsitektur Perangkat Lunak

Pada gambar 4.2 program utama bertindak sebagai *main program*, sedangkan *UPnP Modul* dan *hardware modul* berisikan fungsi-fungsi yang nantinya digunakan oleh sistem. Berikut penjelasan untuk sub sistem tersebut:

1) Program utama

Seperti dijelaskan sebelumnya, program utama yang merupakan *main program* dan yang akan dieksekusi ketika program dijalankan. Dengan menjalankan fungsi-fungsi yang disediakan oleh sub sistem. Berikut flowchart program utama:



Gambar 4.3 Flowchart Program Utama

2) *UPnP Modul*

UPnP modul bertugas sebagai jembatan antara *UPnP device* dan *UPnP control*. Oleh karena itu *UPnP modul* harus dapat melakukan fungsi-fungsi

yang terdapat pada *UPnP,* seperti *advertising service, control,* dan *eventing*. Kemudian data dari *UPnP control* harus dapat diteruskan ke *hardware modul* agar *service* dapat dijalankan. Pada *UPnP modul* ini pembuatan *UPnP device* mengikuti aturan dari spesifikasi *BinaryLight1.xml* yang terdapat pada *UPnP Forum*. Spesifikasi *device* yang dibuat dalam penelitian ini adalah:

Tabel 4.1 Spesifikasi *UPnP Device*

Elemen	Tipe Data	Nama
deviceType	*Single URI*	urn:schemas-upnp-org:device:BinaryLight:1
friendlyName	*String*	LUPnP
manufacturer	*String*	Probelmtech
modelName	*String*	Virtual Light
UDN	*Single URI*	uuid:cc93d8e6-6b8b-4f60-87ca-228c36b5b0e8

```
1.  . . . . . .
2.
3.      <?xml version="1.0"?>
4.      <root xmlns="...">
5.      ...
6.        <specVersion>
7.        <device>
8.          ...
9.          <serviceList>
10.           <service>
11.           <serviceType> …… </serviceType>
12.           <serviceId> …… </serviceId>
13.           <SCPDURL> …… </SCPDURL>
14.           <controlURL> …… </controlURL>
15.           <eventSubURL> …… </eventSubURL>
16.           </service>
17.         </serviceList>
18.       </device>
19.     </root>
20. . . . . .
```

Gambar 4.4 Rancangan daftar *service* pada *device description*

Dokumen *device description* terdapat dua bagian utama, pertama berisikan informasi spesifikasi *device* yang ditunjukkan oleh Tabel 4.1 dan yang kedua adalah layanan atau *service* yang disediakan oleh *device* tersebut. Gambar 4.4 menujukkan spesifikasi minimal yang terdapat pada *device description*. Pada penelitian ini, pada *device description* memuat tiga *serviceList* dengan rancangan ditunjukkan oleh tabel 4.2 – 4.4.

Tabel 4.2 Rancangan *switch serviceList* pada *device description*

Elemen	Tipe Data	Nama
serviceType	*Single URI*	urn:schemas-upnp-org:service:SwitchPower:1
serviceID	*Single URI*	urn:upnp-org:serviceId:SwitchPower:1
SCPDURL	*Single URI*	Resource/SwitchPower1.xml
controlURL	*Single URI*	Resource/SwitchPower/Control
eventSubURL	*Single URI*	Resource/SwitchPower/Event

Tabel 4.3 Rancangan *dimming serviceList* pada *device description*

Elemen	Tipe Data	Nama
serviceType	*Single URI*	urn:schemas-upnp-org:service:Dimming:1
serviceID	*Single URI*	urn:upnp-org:serviceId:Dimming:1
SCPDURL	*Single URI*	Resource/Dimming1.xml
controlURL	*Single URI*	Resource/Dimming/Control
eventSubURL	*Single URI*	Resource/Dimming/Event

Tabel 4.4 Rancangan *colorchange serviceList* pada *device description*

Elemen	Tipe Data	Nama
serviceType	*Single URI*	urn:schemas-upnp-org:service:ColorChange:1
serviceID	*Single URI*	urn:upnp-org:serviceId:ColorChange:1
SCPDURL	*Single URI*	Resource/ColorChange1.xml
controlURL	*Single URI*	Resource/ColorChange/Control
eventSubURL	*Single URI*	Resource/ColorChange/Event

Langkah berikutnya yaitu mendefisinikan *service* yang tersedia pada *device* yang akan dibuat. *Service description* berisikan informasi detail mengenai *service*. Pembuatan *service description* mengacu pada *template* yang dibuat oleh *UPnP Forum*. Pada *template* yang disediakan kita hanya perlu untuk mengisi informasi pada tempat yang telah disediakan.

```
1.    <?xml version="1.0"?>
2.    <scpd xmlns="urn:schemas-upnp-org:service-1-
3.    0">
4.      <specVersion>
5.        <major>1</major>
6.        <minor>0</minor>
7.      </specVersion>
8.      ….
9.    </scpd>
```

Gambar 4.5 *Template header service description*

Gambar 4.5 merupakan *header* dari dokumen *service* yang akan dibuat. Pada baris 9 *<scpd>* menunjukkan bahwa dokumen tersebut merupakan sebuah *service description*. Elemen *scpd* harus beratribut *xmlns* dari *urn:schemas-upnp-org:service-1-0.* Pada setiap *service description* memuat *actionList* yang terdapat pada *service* tersebut. Pembuatan *actionList* didasarkan pada *template* ditunjukkan oleh Gambar 4.6 yang telah disediakan oleh *UPnP Forum.*

```
1.    <scpd>
2.       …
3.      <actionList>
4.        <action>
5.          <name> … </name>
6.            <argumentList>…</argumentList>
7.        </action>
8.        ……
9.      </actionList>
10.   <scpd>
```

Gambar 4.6 *Template action list*

Pada setiap dokumen *service description* memiliki *action list* yang berbeda. Setiap *action* mempunyai *argument List*. Setiap argument memiliki arah sebagai *input* atau sebagai *output.* Setiap argument harus berhubungan dengan sebuah *state variable*. *State variable* digunakan untuk memodelkan status dari *service* yang berjalan.

Tabel 4.5 Rancangan *actionList* dokumen *switch service*

actionList	*argumentList*		
	Nama	*relatedStateVariable*	*direction*
SetTarget	*newTargetValue*	*Target*	*in*
GetTarget	*RetTargetValue*	*Target*	*out*
GetStatus	*ResultStatus*	*Status*	*out*

Tabel 4.6 Rancangan *serviceStateTable* dokumen *switch service*

serviceStateTable	Tipe data	*Default value*	*event*
Target	*boolean*	*0*	Tidak
Status	*boolean*	*0*	ya

Tabel 4.5 dan tabel 4.6 merupakan perancangan dokumen *service description* untuk *switch service* pada penelitian ini. Penggunaan variabel *boolean* disebabkan saklar hanya mempunyai nilai benar ketika nyala dan salah ketika mati. *SetTarget* merupakan masukan untuk mengganti nilai saklar maka mempunyai *direction in*.

Tabel 4.7 Rancangan *actionList* dokumen *dimming service*

actionList	*argumentList*		
	Nama	*relatedStateVariable*	*direction*
SetLoadLevelTarget	newLoadlevelTarget	LoadLevelTarget	*in*
GetLoadLevelTarget	retLoadlevelTarge	LoadLevelTarget	*out*
GetLoadLevelStatus	retLoadlevelStatus	LoadLevelStatus	*out*

Tabel 4.8 Rancangan *serviceStateTable* dokumen *dimming service*

serviceStateTable	Tipe data	*Default value*	*Allowed value*		*event*
			Minimum	*maximum*	
LoadLevelTarget	ui1	*0*	0	255	Tidak
LoadLevelStatus	ui1	*0*	0	255	ya

Tabel 4.7 dan tabel 4.8 merupakan rancangan *service description* untuk *dimming service*. Tipe data yang digunakan adalah *ui1* atau *unsigned integer* dan mempumyai rentang nilai 0 – 255 sesuai dengan nilai minimal dan maksimal masukan dari PWM.

Tabel 4.9 Rancangan *actionList* dokumen *color service*

actionList	*argumentList*		
	Nama	*relatedStateVariable*	*direction*
SetColorChangeTarget	newRedTarget	ColorRedTarget	*in*
	newGreenTarget	ColorGreenTarget	*in*
	newBlueTarget	ColorBlueTarget	*in*
GetColorChangeTarget	retRedTarget	ColorRedTarget	*out*
	retGreenTarget	ColorGreenTarget	*out*

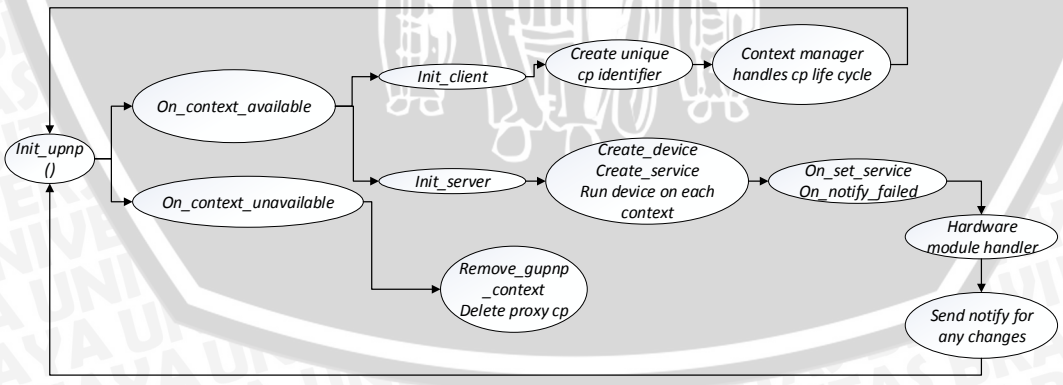
	retBlueTarget	ColorBlueTarget	*out*
GetColorChangeStatus	RedStatus	ColorRedStatus	*out*
	GreenStatus	ColorGreenStatus	*out*
	BlueStatus	ColorBlueStatus	*out*

Tabel 4.10 Rancangan *serviceStateTable* dokumen *color service*

serviceStateTable	Tipe data	*Default value*	*Allowed value*		*event*
			Minimum	*maximum*	
ColorRedTarget	ui1	0	0	255	Tidak
ColorGreenTarget	ui1	0	0	255	Tidak
ColorBlueTarget	ui1	0	0	255	tidak
ColorRedStatus	ui1	0	0	255	Ya
ColorGreenStatus	ui1	0	0	255	Ya
ColorBlueStatus	ui1	0	0	255	ya

Pembuatan dokumen *color service* pada tabel 4.9 dan tabel 4.10 adalah dengan memodifikasi *template* dari *dimming service*. Hal ini dilakukan karena *UPnP Forum* belum mengeluarkan standar untuk *color service* pada *BinaryLight device*.

Setelah perancangan *device* dan *service selesai*, maka dibuat perancangan pengimplementasian *device* menggunakan pustaka G*UPnP*. Program *UPnP* modul mempunyai diagram siklus sebagai berikut:

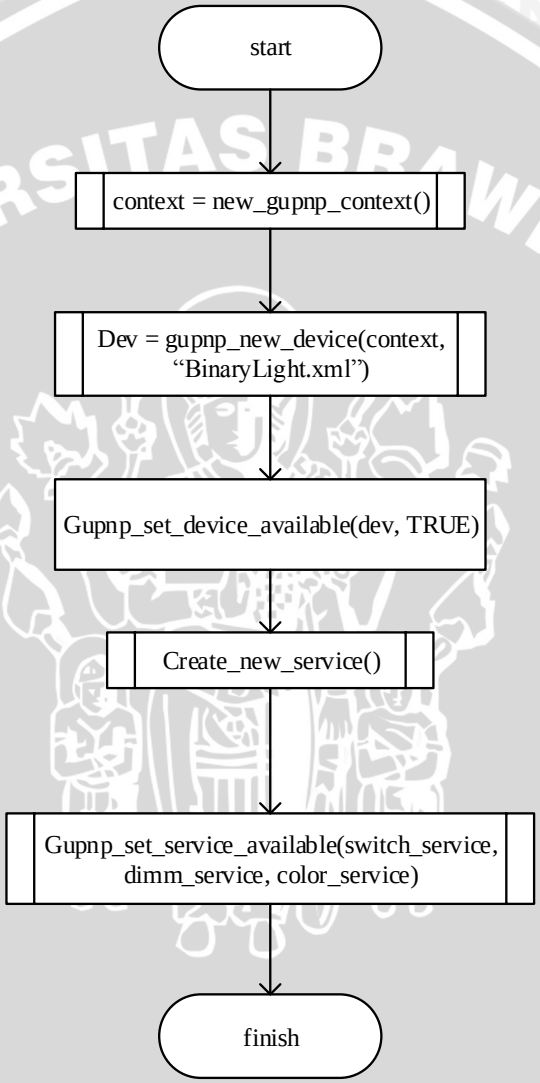


Gambar 4.7 Diagram siklus *UPnP Module*

Pada Gambar 4.7, semua fungsi pada *UPnP module* akan berjalan pada *thread* menggunakan fasilitas dari *library glib*. Siklus tersebut terjadi mulai *device* mulai dinyalakan, proses *on_context_available* dan *on_context_unavailble* adalah untuk

mengecek *IP* pada *interface device UPnP*. Ketika ada *interface* baru memiliki *IP,* program akan melakukan proses *init_client* dan *init_server.* Kemudian *device* akan membuat sebuah *proxy* untuk masing-masing *context* yang dibuat. Tujuan pembuatan *proxy* ini adalah untuk menangani proses *invoke* dari masing-masing *interface,* dikarenakan pada *device* yang dibuat terdapat lebih dari satu *network interface*.

Diagram alir pada gambar 4.8 merupakan bagian dari pembuatan setiap *proxy* pada proses *init_server.* Diagram alir program *UPnP module*:

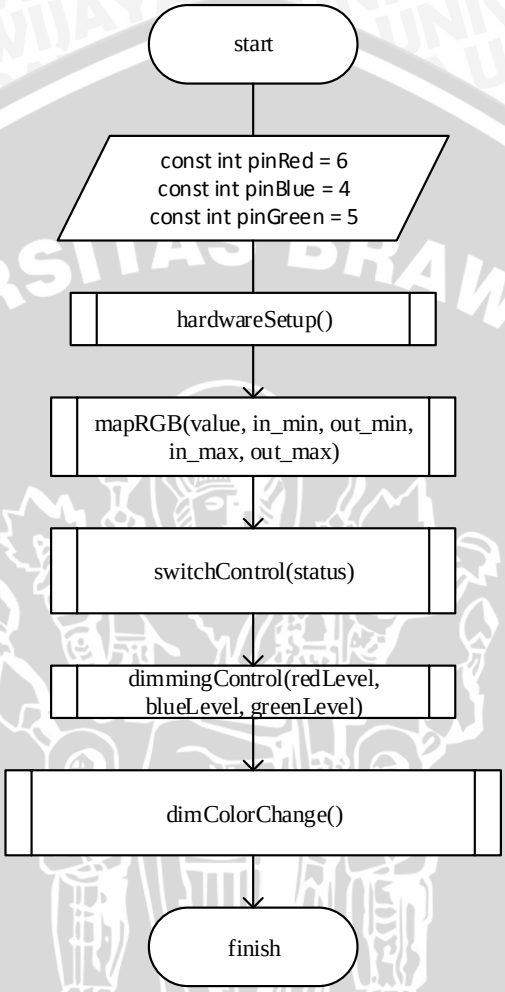


Gambar 4.8 Diagram Alir *UPnP Modul*

3) *Hardware Modul*

Pada bagian perancangan perangkat keras digunakan LED RGB. LED RGB agar dapat menghasilkan tingkat warna dan intensitas yang berbeda maka perlu dibuat antar muka untuk mengendalikan proses tersebut. PWM dapat digunakan untuk mengatur tingkat warna dan intensitas cahaya LED. Pada *hardware modul* proses pengaturan PWM dibantu

dengan menggunakan pustaka *wiringPi*. *wiringPi* memiliki fitur lengkap untuk memanipulasi GPIO pada *Raspberry Pi*, salah satunya adalah *softPWM*. Dengan *softPWM*, semua pin GPIO pada *raspberry pi* dapat diatur nilai PWM-nya. Untuk dapat mengatur tingkat intensitas cahaya tanpa mengubah warna LED, dapat digunakan perbandingan. Diagram alir *hardware modul* ditunjukkan oleh gambar 4.9.



Gambar 4.9 Diagram alir *hardware modul*

4.2 Implementasi

Sub bab ini membahas mengenai tahapan impelementasi berdasarkan perancangan yang telah dibuat sebelumnya.

4.2.1 Implementasi Perangkat Keras

Implementasi perangkat keras sesuai dengan perancangan pada bab sebelumnya menggunakan *raspberry pi 2*, LED RGB, dan resistor. Led RGB dihubungkan ke pin GPIO *raspberry pi.* Pin LED warena merah dihubungkan dengan pin GPIO 6, pin LED warna biru ke pin GPIO 4, pin LED warna hijau dihubungkan dengan pin GPIO 5, dan pin *ground* LED dihubungkan dengan pin

ground yang terdapat pada *raspberry pi.* Hasil implementasi perangkat keras pada penelitian ini dapat dilihat pada Gambar 4.10.

Gambar 4.10 Impelementasi Perangkat Keras

4.2.2 Implementasi Perangkat Lunak

Impelementasi perangkat lunak pada penelitian ini meliputi implementasi *main program, UPnP modul* dan *hardware modul.*

4.2.2.1 Impelementasi *main program*

main program merupakan *source code* yang dieksekusi ketika program dijalankan. *Main program* memanggil fungsi dari *UPnP modul* dan *hardware modul* untuk dijalankan ketika program dieksekusi. *Source code main program* ditujukkan pada Gambar 4.11.

```
1.  #include <stdio.h>
2.  #include "../Header/HardwareHandler.h"
3.  #include "../Header/UPnPCore.h"
4.
5.
6.  int main()
7.  {
8.      /**/
9.      //by default lamp is off
10.     switchStatus = FALSE;
11.
12.     g_print("Default Status %s.\n", switchStatus ? "on"
13. : "off");
14.
15.     hardwareSetup();
16.     UPnPDump();
17.
18.     GMainLoop *mainLoop;
19.     mainLoop = g_main_loop_new(NULL, FALSE);
20.     g_main_loop_run(mainLoop);
21.
22.     return 0;
23.
24. }
25.
26.
```

Gambar 4.11 *Source Code main program*

Penjelasan proses pada Gambar 4.11 yaitu:

1) Baris 1 – 3 proses inisialisasi *library* yang dipakai oleh program.
2) Baris 10 adalah inisialisasi awal *default* status lampu yaitu mati, dalam program nilai switchStatus adalah FALSE atau mati.
3) Baris 15 adalah proses inisialisasi pin GPIO pada *raspberry pi.*
4) Baris 16 adalah proses inisialisasi *UPnP*.
5) Baris 20 adalah inisialisasi data struct *main event loop* untuk aplikasi Glib, GMailLoop digunakan untuk memanajemen semua sumber daya untuk Glib dan GTK+. Penggunaan *data struct* dikarenakan pustaka G*UPnP* menggunakan GLib sebagai dasar pembuatan programnya.
6) Baris 21 adalah pembuatan struktur GMainLoop baru.
7) Baris 22 adalah menjalankan mainLoop.

4.2.2.2 Implementasi *UPnP Modul*

UPnP modul adalah kumpulan fungsi-fungsi untuk menangani proses *UPnP* dengan bantuan pustaka *GUPnP*. *UPnP modul* didalamnya juga terdapat *device architecture* dan *service description*. Setiap *service* yang akan diimplementasikan dibuat fungsi yang berbeda-beda. Tahapan implementasi *service* dimulai dengan

pembuatan *service description* yang berformat dokumen XML. Untuk *service description* lengkap dapat dilihat pada halaman lampiran.

```
. . . . . .

<actionList>
    <action>
      <name>SetTarget</name>
      <argumentList>
        <argument>
          <name>NewTargetValue</name>

<relatedStateVariable>Target</relatedStateVariable>
          <direction>in</direction>
        </argument>
      </argumentList>
    </action>
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="no">
      <name>Target</name>
      <dataType>boolean</dataType>
      <defaultValue>0</defaultValue>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>Status</name>
      <dataType>boolean</dataType>
      <defaultValue>0</defaultValue>
    </stateVariable>
  </serviceStateTable>

. . . . . .
```

Gambar 4.12 Controh *service description Switch Control*

Pada Gambar 4.12 merupakan *service description* untuk *service* kontrol status lampu, pada *service description* hal yang paling untuk diperhatikan adalah bagian *actionList* dan *serviceStateTable*. Variabel pada *actionList* digunakan sebagai *sharing* variabel antara *control point* dan *device* dan menjelaskan variabel tersebut sebagai *input* atau *output* kepada *device.* Kumudian pada bagian *serviceStateStable* terdapat pengaturan variabel untuk dikirim statusnya pada jaringan atau tidak, pada *UPnP* bagian ini berperan pada saat *Eventing*. Dokumen *device description* dan *service description* dilampirkan pada halaman lampiran. *Berikutnya*, setelah kita membuat fungsi program untuk menangani proses *UPnP*. Fungsi dibuat untuk setiap *service*, karena setiap *service* memiliki *action* yang berbeda-beda.

```
. . . . . .

G_MODULE_EXPORT
void        set_target_cb(GUPnPService        *service,
GUPnPServiceAction *action, gpointer user_data) {

    gboolean target;
    gUPnP_service_action_get(action,  "NewTargetValue",
G_TYPE_BOOLEAN, &target, NULL);
```

```
11.         switchControl(target);
12.
13.         gUPnP_service_notify(service,                "Status",
14.     G_TYPE_BOOLEAN, target, NULL);
15.         gUPnP_service_action_return(action);
16.
17.     }
18.
19.     . . . . . .
```

Gambar 4.13 Fungsi *switchService* untuk *service Switch Power*

Berikut penjelasan fungsi pada Gambar 4.13:

1) Baris 3 digunakan untuk mendeklarasikan fungsi diekspor sebagai modul.
2) Baris 7 adalah variabel yang digunakan untuk menyimpan status dari *Control Point*.
3) Baris 8 adalah proses untuk menangkap *action* yang dilakukan oleh *control point*.
4) Baris 11 digunakan untuk melakukan *action* oleh *device* dengan mengirim nilai *target* kepada fungsi *switchControl* yang dimiliki oleh *hardware modul*.
5) Baris 13 digunakan untuk mengirim notifikasi *event* kepada *control point*.
6) Baris 15 mengindentifikasikan *service* berhasil diimplementasikan

Untuk fungsi *service dimmer* dan perubahan warna memiliki kesamaan struktur kode yang terdapat pada fungsi *switch service*. Pertama dibuat variabel untuk menyimpan input dari *control point*, variabel tersebut dapat berupa *boolean*, *integer* atau *character*. Kedua, memanggil fungsi untuk mendapatkan *input* dari *control point* yaitu `gUPnP_service_action_get(gUPnP_service_action, . . .)`. Ketiga, memanggil fungsi untuk mejalankan *service*. Pada penelitian ini nilai yang didapatkan dari *control point* akan dipakai sebagai parameter pengendalian lampu. Keempat, mengirim *event* kepada *control point* dengan menggunakan fungsi `gUPnP_service_notify()`. Terakhir, memanggil fungsi `gUPnP_service_action_return` untuk mengindetifikasikan *service* berhasil dilakukan.

4.2.2.3 Impelementasi *Hardware Modul*

Impelementasi *hardware modul* dibuat dengan bantuan pustaka *wiringPi*. *Hardware modul* berisi fungsi-fungsi untuk menangani proses interaksi dengan perangkat keras. Untuk dapat menggunakan *wiringpi* dalam program, terlebih dahulu dilakukan proses inisialisasi pada awal program dijalankan. Proses inisialisasi ditujukkan oleh Gambar 4.14.

```
1.  void hardwareSetup() {
2.
3.      wiringPiSetup();
4.      softPwmCreate(pinRed, 0, 100);
5.      softPwmCreate(pinGreen, 0, 100);
6.      softPwmCreate(pinBlue, 0, 100);
7.
8.      softPwmWrite(pinRed, 0);
9.      softPwmWrite(pinGreen, 0);
10.     softPwmWrite(pinBlue, 0);
11. }
```

Gambar 4.14 Fungsi inisialisasi GPIO *wiringpi*

Penjelasan proses pada Gambar 4.14 yaitu:

1) Baris 3 proses inisialisasi *library* yang dipakai oleh program, *wiringpiSetup()* adalah fungsi untuk memakai konfigurasi pin milik *wiringpi*.
2) Baris 4-5 adalah proses inisialisasi *pin GPIO* yang dipakai. Pada skripsi ini *pin GPIO* yang dipakai adalah untuk mengatur PWM, maka dipakai fungsi *softPwmCreate().* Didalam fungsi tersebut dinisialisasi nomor *pin* yang dipakai, nilai PWM paling kecil yaitu 0, dan nilai PWM terbesar yaitu 100. *Library wiringpi* mengatur nilai PWM sampai 100 dikarenakan menggunakan *pin GPIO* biasa, sehingga pembangkit sinyal dari pin tersebut menggunakan waktu .
3) Baris 8-10 adalah untuk memberikan nilai awal pada pin *GPIO* yang dipakai. Fungsi *softPwmWrite()* diisikan pin *GPIO* yang akan ditulis nilainya, serta nilai PWM untuk pin tersebut.

```
1.  long mapRGB(int value, long in_min, long in_max, long
2.  out_min, long out_max){
3.        return (value - in_min) * (out_max - out_min) /
4.  (in_max - in_min) + out_min;
5.  }
```

Gambar 4.15 Fungsi *mapping* nilai

Pada Gambar 4.15 adalah fungsi untuk menghitung nilai perbandingan tingkat intensitas dan warna pada lampu. Fungsi *mapRGB* dibutuhkan untuk mengubah tingkat intensitas tanpa mengubah warna lampu. Hasil kembalian fungsi *mapRGB* adalah sebagai nilai PWM baru untuk lampu.

```
1.  void switchControl(bool setStatus) {
2.
3.      switchStatus = setStatus;
4.      printf("Status Lampu: %s\n", switchStatus ? "on" :
5.  "off");
6.      if (setStatus) {
7.          redDump = 100;
8.          greenDump = 100;
9.          blueDump = 100;
10.
11.         softPwmWrite(pinRed, redDump);
12.         softPwmWrite(pinGreen, greenDump);
13.         softPwmWrite(pinBlue, blueDump);
14.         printf("%d   %d   %d\n",   redDump,   greenDump,
15. blueDump);
16.     }
17.     else {
18.         softPwmWrite(pinRed, 0);
19.         softPwmWrite(pinGreen, 0);
20.         softPwmWrite(pinBlue, 0);
21.     }
22. }
```

Gambar 4.16 Fungsi *switch service*

```
1.  void dimmingControl(int _dimValue) {
2.
3.      dimValue = mapRGB(_dimValue, 0, 255, 0, 100);
4.
5.      if(switchStatus){
6.          dimColorChange();
7.      }
8.      else
9.          printf("Turn on Lamp First\n");
10.
11. }
```

Gambar 4.17 Fungsi Pengatur Intensitas Cahaya

```
1.  void  colorControl(int  redLevel,  int  greenLevel,  int
2.  blueLevel) {
3.
4.      redDump = mapRGB(redLevel, 0, 255, 0, 100);
5.      greenDump = mapRGB(greenLevel, 0, 255, 0, 100);
6.      blueDump = mapRGB(blueLevel, 0, 255, 0, 100);
7.
8.      if(switchStatus) {
9.        dimColorChange();
10.     }
11.     else
12.         printf("Turn on Lamp First\n");
13. }
```

Gambar 4.18 Fungsi Pengatur Warna Cahaya

Impelementasi service terlihat pada fungsi pada Gambar 4.16 sampai pada Gambar 4.18. Ketiga fungsi tersebut memiliki kesamaan, yaitu menerima variabel

dari fungsi service yang terdapat pada modul UPnP. Hubungan antar modul dapat dilihat pada bab perancangan sebelumnya.

```
1.      void dimColorChange(){
2.        int dimm = dimValue;
3.
4.        int redDimm = mapRGB(redDump, 0, 100, 0, dimm);
5.        int greenDimm = mapRGB(greenDump, 0, 100, 0, dimm);
6.        int blueDimm = mapRGB(blueDump, 0, 100, 0, dimm);
7.
8.        softPwmWrite(pinRed, redDimm);
9.        softPwmWrite(pinGreen, greenDimm);
10.       softPwmWrite(pinBlue, blueDimm);
11.
12.       printf("Status %s. R: %d G: %d B: %d\n", switchStatus
13.     ? "on" : "off", redDimm, greenDimm, blueDimm);
14.
15.     }
```

Gambar 4.19 Fungsi Pengatur Nilai PWM

Pada Gambar 4.19adalah fungsi untuk memberi nilai PWM pada pin *GPIO raspberry pi*. Nilai PWM didapatkan dari *client* atau *control point* yang memberi nilai pada *service* pengaturan intensitas cahaya dan warna cahaya. Sebelum dituliskan ke pin GPIO, nilai PWM dihitung terlebih dahulu dengan perbandingan yang terdapat pada fungsi *mapRGB*. Nilai yang dibandingkan adalah nilai maksimal PWM yang dapat diatur oleh *softPWM* yang dimiliki oleh pustaka *wiringPi* yaitu 100 dengan nilai intensitas cahaya yang diperoleh dari *dimming service* pada modu *UPnP*. Dengan menggunakan perbandingan dapat diperoleh nilai tengan PWM yang tepat tanpa mengubah intensitas atau warna pada lampu. Nilai yang diperoleh dari perhitungan tersebut dituliskan ke pin GPIO *raspberry pi* dengan fungsi *softPwmWrite(pin, value)*.

BAB 5 PEMBAHASAN

Pada bab ini dilakukan proses pengujian dan analisis dari sistem yang telah dibuat. Tujuan dari dilakukannya pengujian ini adalah untuk mengetahui bahwa semua kebutuhan fungsional maupun non-fungsional yang dirancang sebelumnya terpenuhi. Hal-hal yang diuji pada sistem ini yaitu mengenai fungsionalitas dari perangkat keras dan perangkat lunak yang digunakan dalam penelitian ini.

Pengujian sistem dilakukan dengan beberapa perubahan program sesuai dengan pengujian yang dibutuhkan. Pada setiap pengujian dipersiapkan scenario dari pengujian.

Analisis dari sistem dilakukan dengan membandingkan data pada pengujian dibandingkan dengan hipotesa dan ditarik kesimpulan dari setiap pengujian yang ada pada setiap sub bab.

5.1 Pengujian

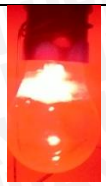
Pengujian dalam penelitian ini terdiri dari beberapa proses uji coba sostem dikarenakan untuk mengetahui apakah semua sub-sistem dapat bekerja sesuai dengan yang diinginkan. Pengujian yang dilakukan adalah pengujian fungsionalitas dan pengujian akurasi. Pengujian fungsionalitas adalah pengujian mengenai fungsionalitas dari perangkat keras dan perangkat lunak yang digunakan pada sistem tersebut, sedangkan pengujian akurasi adalah pengujian yang mengacu pada tingkat kesesuaian dan kebenaran suatu hasil dari perangkat keras atau perangkat lunak dari sistem.

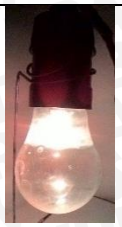
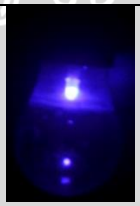
Berdasarkan metode penelitian dan perancangan pada penelitian maka dilakukan pengujian pada sistem untuk dapat menganalisis hasil yang didapatkan. Penjelesan hasil pengujian dan analisis sistem sebagai berikut.

5.1.1 Pengujian LED RGB

Led RGB merupakan *output* yang mewakili lampu pada *smart home*. Led ini dapat diatur warnanya dan intensitas cahayanya. Pengujian Led dilakukan untuk mengetahui pengaruh nilai PWM terhadap warna yang dihasilkan oleh LED. Pengujian dilakukan dengan memberikan nilai PWM yang telah ditentukan. Warna yang dipilih berdasarkan spektrum warna RGB dasar.

Tabel 5.1 Hasil Pengujian LED RGB

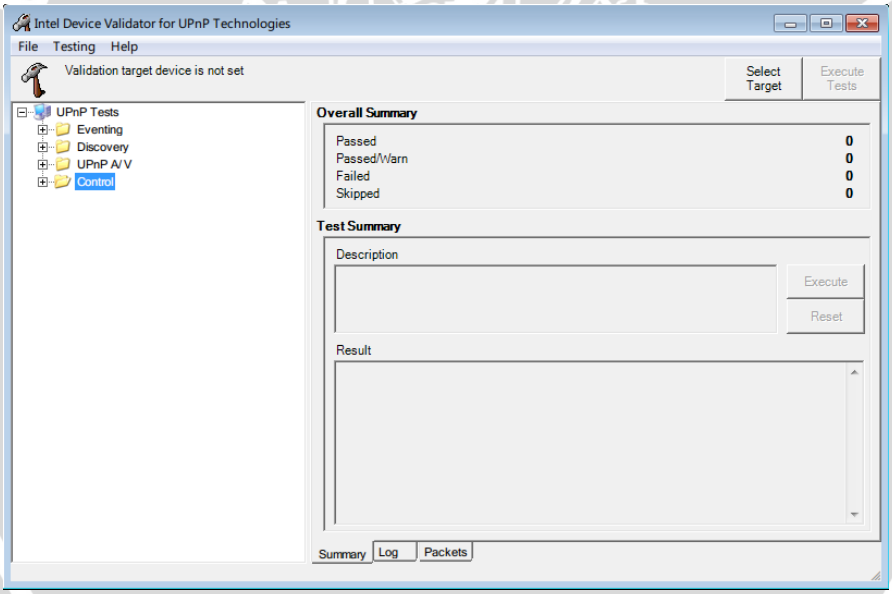
No.	R	G	B	Map R	Map G	Map B	Hasil	Warna Asli
1.	255	0	0	100	0	0		

2.	255	128	0	100	50	0		
3.	255	255	0	100	100	0		
4.	128	255	0	50	100	0		
5.	0	255	0	0	100	0		
6.	0	255	255	0	100	100		
7.	0	128	255	0	50	100		
8.	0	0	255	0	0	100		
9.	128	0	255	50	0	100		

10.	255	0	255	100	0	100		
11.	255	0	128	100	0	50		

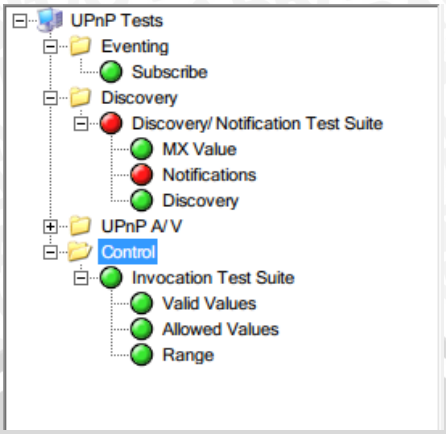
5.1.2 Pengujian *UPnP Device*

Tujuan pengujian *UPnP* device adalah untuk mengetahui *device* yang dibuat sesuai dengan *UPnP* Device Architectur 1.1. Pengujian dilakukan dengan menggunakan aplikasi *Device Validator* milik Intel. *Device Validator* juga menguji respon dari *device* untuk melihat seberapa mampu *device* menangani proses *subscription* dan *invocations*.

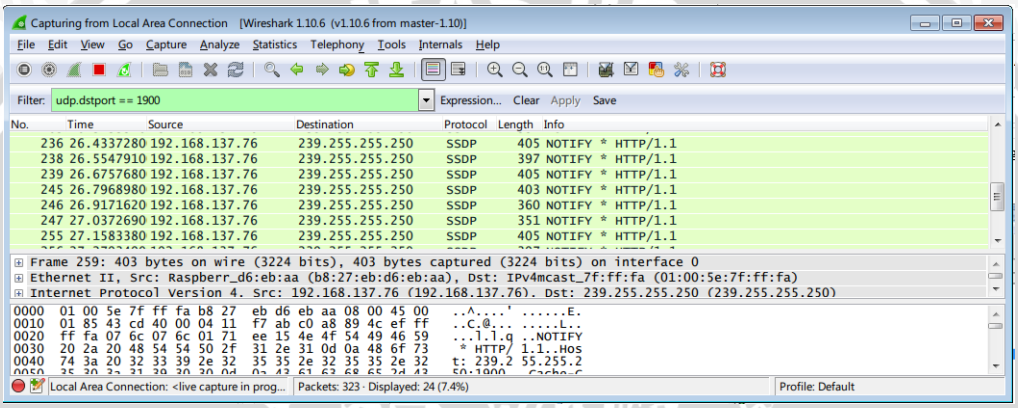


Gambar 5.1 *UPnP Device Validator*

Pada *Device Validator* dilakukan pengujian *Eventing, Discovery* dan *Control*. Hasil dari pengujian dapat dilihat pada kolom *Overall Summary*. Pengujian dilakukan pada *access point WiFi* lokal. Pengujian juga menggunakan perangkat lunak Wireshark, perangkat lunak ini diperlukan jika *device validator* gagal menjalankan tes dapat dilihat *log* paket yang terdapat pada jaringan



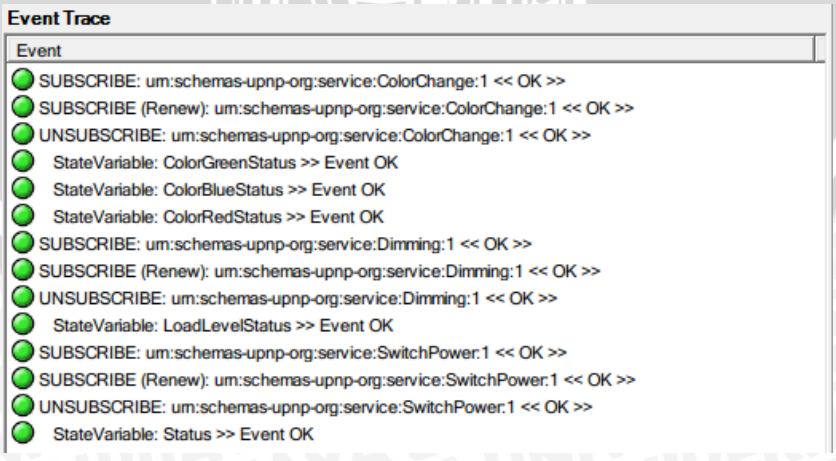
Gambar 5.2 Hasil Pengujian *UPnP Device*



Gambar 5.3 Hasil filter SSDP pada wireshark

Pengujian pada *device validator* dan hasil filter pada gambar 5.3 dapat dianalisa sebagai berikut:

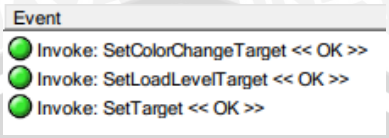
1. Pada pengujian *subscription*, *device validator* melakukan proses *subscription, un-subsription,* pembaruan *subscription* dan mengecek hasil eror.



Gambar 5.4 Hasil tes *subscription*

Dari hasil pengujian yang didapatkan pada gambar 5.4 dapat disimpulan *device* dapat memberikan informasi *event* kepada *control point.*

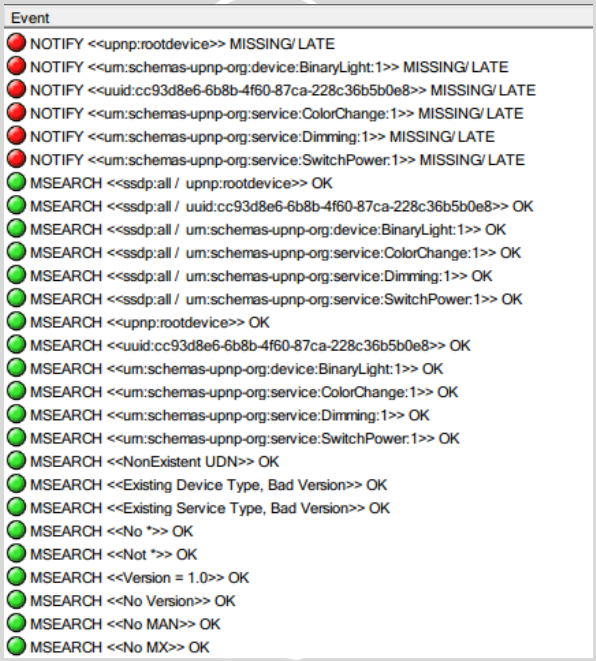
2. Pengujian kontrol dilakukan dengan memberikan nilai yang valid dan tidak valid kepada *device*. Hal tersebut dilakukan untuk mengetahui *device* mampu menerima aksi dari *control point*.



Gambar 5.5 Hasil tes kontrol

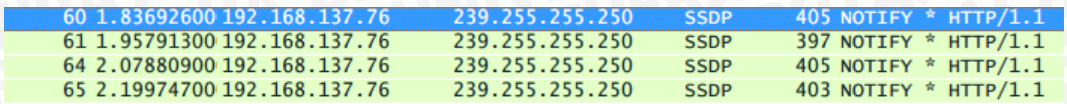
Dari Gambar 5.5 Hasil tes kontrol dapat disimpulkan *device* mampu menerima aksi dari *control point*.

3. Pengujian *discovery* dilakukan dengan menguji M-SEARCH, proses notifikasi dan proses *discovery*.

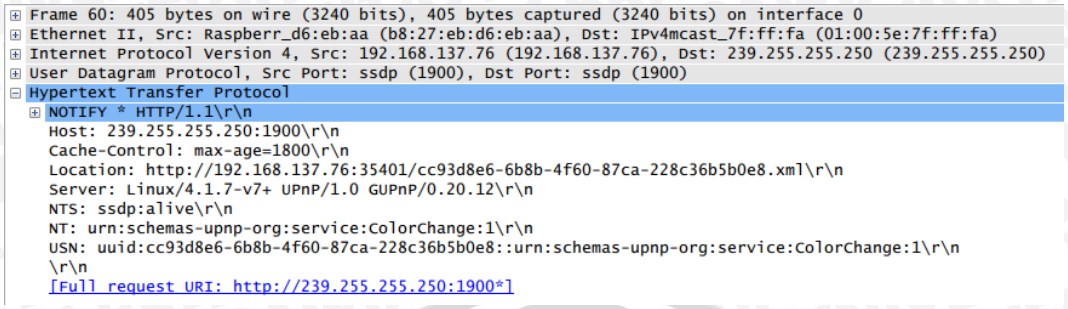


Gambar 5.6 hasil tes *discovery*

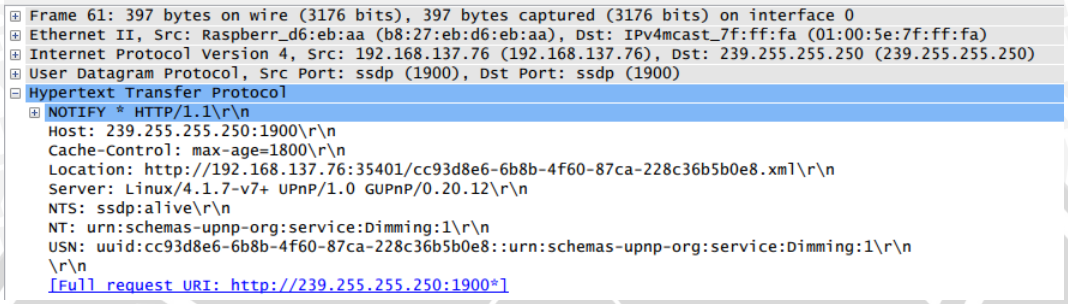
Pada Gambar 5.6 hasil tes *discovery*, *device* tidak mampu untuk melakukan proses *notification*. Proses n*otification* dilakukan *device* dengan mengirimkan paket SSDP menggunakan NOTIFY dengan *ssdp:alive.* Untuk memastikan hal tersebut didunakan *log* paket pada *wireshark*.



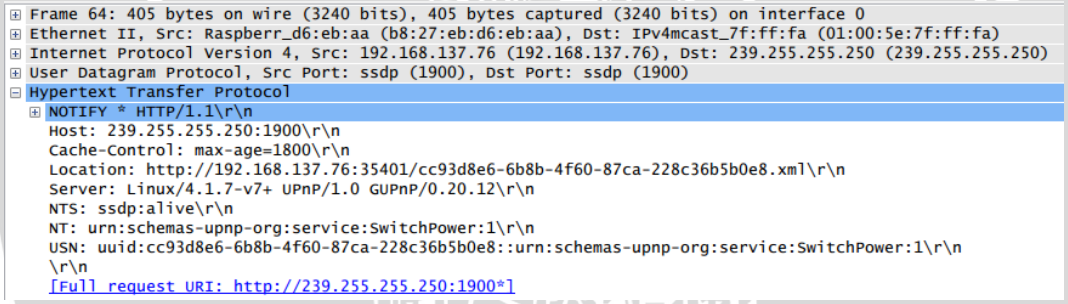
Gambar 5.7 hasil SSDP paket yang ditangkap *wireshark*



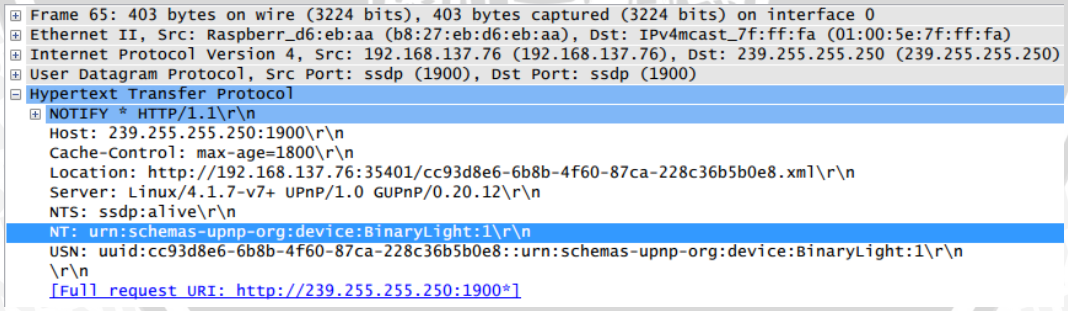
Gambar 5.8 isi SSDP paket nomer 60



Gambar 5.9 isi SSDP paket nomer 61



Gambar 5.10 isi SSDP paket nomor 64

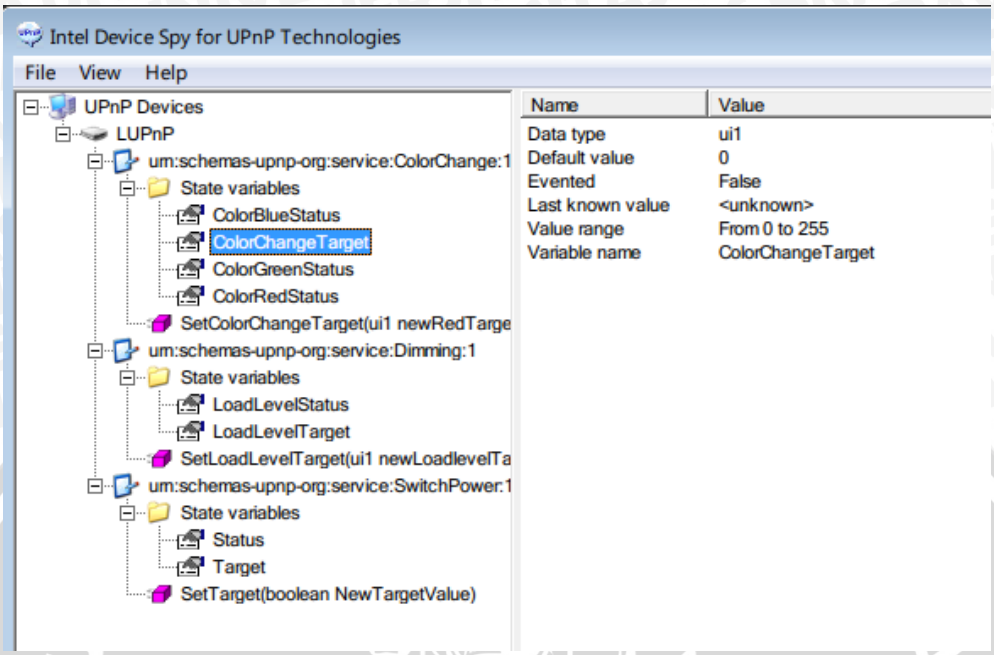


Gambar 5.11 isi paket SSDP paket nomor 65

5.1.3 Pengujian Sistem Keseluruhan

Pengujian sistem keseluruhan adalah pengujian yang dilakukan untuk melihat kesiapan sistem yang dijalankan. Tujuan dari pengujian ini adalah untuk menguji kemampuan sistem, intregrasi sub sistem dan kondisi yang terjadi pada setiap modul. Pengujian sistem secara keseluruhan dilakukan dengan menggunakan

aplikasi *Device Spy* milik Intel. *Device Spy* adalah *universal control point UPnP* yang digunakan untuk menguji *device* dengan *control point*.

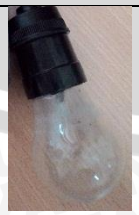


Gambar 5.12 Intel *Device Spy*

Pengujian dilakukan dengan melakukan proses *invoke* untuk masing-masing servis menggunakan *control point.* Pengujian dilakukan pada jaringan lokal menggunakan *access point WiFi.* Waktu respon didapat dari selisih waktu *invocation service* dan *notification* perubahan *service.*

1. *SwitchPower*

Tabel 5.2 Hasil Pengujian Service *SwitchPower*

No.	Status	Hasil	Waktu Respon
1.	Nyala		77 ms
2.	Mati		62 ms

2. *Dimming*

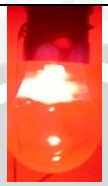
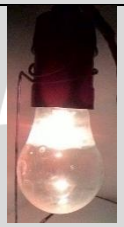
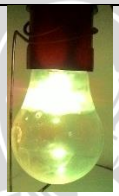
Tabel 5.3 Hasil Pengujian *Service Dimming*

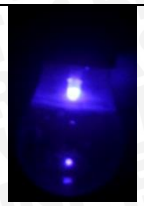
No.	Nilai *Dimming*	Hasil	Waktu Respon
1.	51		106 ms
2.	102		103 ms
3.	153		88 ms
4.	204		51 ms
5.	255		54 ms

3. *ColorChange*

Pengujian warna menggunakan data PWM pada tabel 5.4.

Tabel 5.4 Hasil Pengujian Service *ColorChange*

No.	R	G	B	Hasil	Waktu Respon
1.	255	0	0		58 ms
2.	255	128	0		48 ms
3.	255	255	0		64 ms
4.	128	255	0		104 ms
5.	0	255	0		93 ms
6.	0	255	255		125 ms
7.	0	128	255		113 ms

8.	0	0	255		60 ms
9.	128	0	255		124 ms
10.	255	0	255		66 ms
11.	255	0	128		81 ms

5.2 Analisis

Berdasarkan hasil pengujian LED RGB pada subbab 5.1.1 diperoleh hasil pada tabel 5.1 dan dapat disimpulkan bahwa lampu dapat berubah warnanya tergantung nilai RGB yang diberikan pada PWM. Nilai PWM yang terdapat pada Tabel 5.1 dikonversi menjadi nilai dengan rentang 0-100 sesuai dengan nilai PWM yang dapat diberikan ke pin GPIO raspberry pi.

Hasil pengujian validasi *UPnP device* pada pengujian pada subbab 5.1.2 ditunjukkan oleh gambar 5.7 sampai 5.11 dan dari gambar tersebut *device* mampu untuk melakukan proses *notification* kepada *control point* menggunakan *ssdp:alive*. Proses *notification* adalah respon *device* terhadap proses *discovery* yang dilakukan oleh *control point*. *Control Point* menggunakan M-SEARCH untuk mencari *device,* kemudian *device* membalas menggunakan NOTIFY. Berdasarkan hasil pengujian *UPnP* device dapat disimpulkan bahwa *device* valid karena mampu melakukan dan melayani proses *discovery*, *eventing* dan *control.*

Pengujian sistem secara keseluruhan dilakukan pada subbab 5.1.3 dan didapatkan hasil pengujian *switchPower* dapat berjalan dengan baik dan memerlukan waktu rata-rata 69,5 ms, d*imming service* dapat berjalan dengan baik dengan waktu respon rata-rata 98 ms dan *colorChange service* dapat berjalan dengan baik dengan waktu respon rata-rata 85 ms.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, impelementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Berdasarkan hasil perancangan dan implementasi, untuk mengimplementasikan *UPnP device* diperlukan merancang dan membuat *device description* dan *service description,* membuat fungsi untuk menangani *event* yang terjadi pada *UPnP network*, membuat fungsi untuk mengontrol peralatan fisik berdasarkan status pada *UPnP device* dan diperlukan *control point* untuk melakukan uji coba terhadap *UPnP device* yang dibuat.
2. Berdasarkan hasil implementasi dan pengujian, untuk melakukan proses pengontrolan saklar lampu *control point* melakukan proses *invoke* dari *service switch.* Proses *invoke* mengganti variabel *boolean* untuk menguabah keadaan saklar, *false* untuk mematikan lampu dan *true* untuk menyalakan lampu.
3. Berdasarkan hasil implementasi dan pengujian, untuk melakukan proses pengontrolan intensitas cahaya dan pengontrolan warna *control point* mengubah nilai *integer* sesuai dengan batas atas dan bawah nilai PWM yang diberikan pada *service description*.
4. Berdasarkan hasil pengujian, diketahui bahwa sistem berjalan dengan baik yaitu dapat mengontrol keadaan lampu dengan waktu rata-rata yang dibutuhkan untuk proses *invoke* untuk *service switch* adalah 69,5 ms, *service dimming* 98 ms dan untuk *service color* dibutuhkan waktu 85 ms.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan penelitian ini antara lain :

1. Penelitian ini belum memperhatikan aspek keamanan sistem, penelitian selanjutnya dapat lebih diperhatikan lagi pada aspek keamanan sistem.
2. Implementasi *device* pada penelitian ini berjalan pada lokal *access point*, penelitian selanjutnya diharapkan *device* dapat dikontrol dimana saja.
3. Penelitian ini baru sebatas mengontrol lampu LED, pada penelitian selanjutnya dapat diimplementasikan *UPnP device* lain pada bagian *smart home*.

BAB 7 DAFTAR PUSTAKA

Anon., 2015. [Online] Available at: http://www.rayennur.com/wp-content/uploads/2015/05/led_rainbow_by_steeph_k-d6jb8az.jpg

Anon., 2015. [Online] Available at: http://www.mainbyte.com/ti99/electronics/led.html

Anon., 2015. [Online] Available at: https://upload.wikimedia.org/wikipedia/commons/f/f1/RGB_LED.jpg

Anon., 2015. *Arduino PWM.* [Online] Available at: http://www.almico.com/images/pwmwaves.gif

Baniya, R. et al., 2014. Smart Indoor Lighting Control. *IEEE Conference Publications*, pp. 1745-1750.

Cheng, Y.-S., Chen, J.-H., Liu, Y.-H. & Wang, S.-C., 2014. Development of Wireless RGB LED Dimming Technology Using Smart phone. *IEEE Conference Publications*, pp. 1-4.

Forum, U., 2008. *UPnP™ Device Architecture 1.0.* 1st ed. s.l.:UPnP Forum.

Georg, J., Baayen, J., Burton, R. & Ali (Khattak), Z., 2015. *Project GUPnP*. [Online] Available at: https://wiki.gnome.org/Projects/GUPnP [Accessed 18 January 2016].

Henderson, G., 2009. *Wiring Pi.* [Online] Available at: http://wiringpi.com/ [Accessed 18 Januari 2016].

Higuera, J., Hertog, W., Peralvarez, M. & Carreras, J., 2015. Smart Lighting System ISO/IEC/IEEE 21451 Compatible. *IEEE Sensor Journal*, pp. 2595-2602.

Jeronimo , . J. & Weast , J., 2003. *UPnP Design by Example.* Pertama ed. Hillsboro: Intel Press.

Khan, A. A. & Mouftah, H. T., 2011. Web Service for Indoor Energy Management in a Smart Grid Environment. *IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communication*, pp. 1036-1040.

Kumar, A. & Tiwari, N., 2015. Energy Efficient Smart Home Automation System. *International Journal of Scientific Engineering and Research (IJSER)*, pp. 9-11.

Leach, H., 2013. *Controlling a RGB LED with a Raspberry Pi.* [Online] Available at: http://www.henryleach.com/2013/05/controlling-rgb-led-with-raspberry-pi.html [Accessed 12 11 2015].

Li , H. & D., 2013. Optimazation and Implementation of Lighting System for Smart Home Based on RF Technology and GSM Network. *International Journal of Digital Content Technology and its Application (JDCTA)*, pp. 653-661.

Marzuki, A., 2013. *Pulse Width Modulation (PWM).* [Online] Available at: http://andri_mz.staff.ipb.ac.id/pulse-width-modulation-pwm/
[Accessed 11 2015].

Persson, O., 2015. *flickr*. [Online] Available at: https://www.flickr.com/photos/askella/17784086872/in/photolist-mbH2td-qp1aKq-qaLovv-qQ13Vj-cMQJD9-eJMXna-cHe5CN-t6w6rs-dzXGRu-pra3k2-dRUgcY-rvtPer-qyun1w-encB7w-vmq6TP-dYWANc-dYWDN4-dZ3jMy-dZ3mK9-ygAmGA-e369gD-oH5Avi-bseXmh-dG5cVQ-dUBagJ
[Accessed 18 01 2016].

Pi, R., 2015. *Raspberry Pi.* [Online] Available at: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/README.md
[Accessed 18 January 2016].

Sandeep, V., Gopal, K., Amudhan, . A. & Kumar, L. S., 2015. Globally Accessible Machine Automation Using Raspberry Pi Based on Internet of Things. *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1144-1147.

Yiqin, L., Fang, F. & Wei Liu, 2009. Home Networking and Control based on UPnP: An Implementation. *Second International Workshop on Computer Science and Engineering*, pp. 385-389.

LAMPIRAN A FORMAT DEVICE DESCRIPTION

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns="urn:schemas-UPnP-org:device-1-0">
 <specVersion>
  <major>1</major>
  <minor>0</minor>
 </specVersion>
 <device>
  <deviceType>urn:schemas-UPnP-org:device:BinaryLight:1</deviceType>
  <friendlyName>LUPnP</friendlyName>
  <manufacturer>Probelmtech</manufacturer>
  <modelName>Virtual Light</modelName>
  <UDN>uuid:cc93d8e6-6b8b-4f60-87ca-228c36b5b0e8</UDN>
  <serviceList>
   <service>
    <serviceType>urn:schemas-UPnP-org:service:SwitchPower:1</serviceType>
    <serviceId>urn:UPnP-org:serviceId:SwitchPower:1</serviceId>
    <SCPDURL>Resource/SwitchPower1.xml</SCPDURL>
    <controlURL>Resource/SwitchPower/Control</controlURL>
    <eventSubURL>Resource/SwitchPower/Event</eventSubURL>
   </service>
   <service>
    <serviceType>urn:schemas-UPnP-org:service:Dimming:1</serviceType>
    <serviceId>urn:UPnP-org:serviceId:Dimming:1</serviceId>
    <SCPDURL>Resource/Dimming1.xml</SCPDURL>
    <controlURL>Resource/Dimming/Control</controlURL>
    <eventSubURL>Resource/Dimming/Event</eventSubURL>
   </service>
   <service>
    <serviceType>urn:schemas-UPnP-org:service:ColorChange:1</serviceType>
    <serviceId>urn:UPnP-org:serviceId:ColorChange:1</serviceId>
    <SCPDURL>Resource/ColorChange1.xml</SCPDURL>
    <controlURL>Resource/ColorChange/Control</controlURL>
```

```
    <eventSubURL>Resource/ColorChange/Event</eventSubURL>
   </service>
  </serviceList>
 </device>
</root>
```

LAMPIRAN B FORMAT SERVICE DESCRIPTION

B.1 Switch Service

```
<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-UPnP-org:service-1-0">
 <specVersion>
  <major>1</major>
  <minor>0</minor>
 </specVersion>
 <actionList>
   <action>
    <name>SetTarget</name>
    <argumentList>
     <argument>
      <name>newTargetValue</name>
      <relatedStateVariable>Target</relatedStateVariable>
      <direction>in</direction>
     </argument>
    </argumentList>
   </action>
   <action>
    <name>GetTarget</name>
    <argumentList>
     <argument>
      <name>RetTargetValue</name>
      <relatedStateVariable>Target</relatedStateVariable>
      <direction>out</direction>
     </argument>
    </argumentList>
   </action>
   <action>
    <name>GetStatus</name>
    <argumentList>
     <argument>
```

```
        <name>ResultStatus</name>
        <relatedStateVariable>Status</relatedStateVariable>
        <direction>out</direction>
       </argument>
      </argumentList>
     </action>
    </actionList>
    <serviceStateTable>
     <stateVariable sendEvents="no">
      <name>Target</name>
      <dataType>boolean</dataType>
      <defaultValue>0</defaultValue>
     </stateVariable>
     <stateVariable sendEvents="yes">
      <name>Status</name>
      <dataType>boolean</dataType>
      <defaultValue>0</defaultValue>
     </stateVariable>
    </serviceStateTable>
   </scpd>
```

B.2 Dimming Service

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-UPnP-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>SetLoadLevelTarget</name>
      <argumentList>
        <argument>
          <name>newLoadlevelTarget</name>
          <direction>in</direction>
```

```
            <relatedStateVariable>LoadLevelTarget</relatedStateVariable>
          </argument>
        </argumentList>
      </action>
      <action>
        <name>GetLoadLevelTarget</name>
        <argumentList>
          <argument>
            <name>retLoadlevelTarget</name>
            <direction>out</direction>
            <relatedStateVariable>LoadLevelTarget</relatedStateVariable>
          </argument>
        </argumentList>
      </action>
      <action>
        <name>GetLoadLevelStatus</name>
        <argumentList>
          <argument>
            <name>retLoadlevelStatus</name>
            <direction>out</direction>
            <relatedStateVariable>LoadLevelStatus</relatedStateVariable>
          </argument>
        </argumentList>
      </action>
    </actionList>
    <serviceStateTable>
      <stateVariable sendEvents="no">
        <name>LoadLevelTarget</name>
        <dataType>ui1</dataType>
        <defaultValue>0</defaultValue>
        <allowedValueRange>
          <minimum>0</minimum>
          <maximum>255</maximum>
        </allowedValueRange>
```

```
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>LoadLevelStatus</name>
      <dataType>ui1</dataType>
      <defaultValue>0</defaultValue>
      <allowedValueRange>
        <minimum>0</minimum>
        <maximum>255</maximum>
      </allowedValueRange>
    </stateVariable>
  </serviceStateTable>
</scpd>
```

B.3 Color Service

```
<?xml version="1.0" encoding="utf-8"?>
<scpd xmlns="urn:schemas-UPnP-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>SetColorChangeTarget</name>
      <argumentList>
        <argument>
          <name>newRedTarget</name>
          <direction>in</direction>
          <relatedStateVariable>ColorRedTarget</relatedStateVariable>
        </argument>
        <argument>
          <name>newGreenTarget</name>
          <direction>in</direction>
          <relatedStateVariable>ColorGreenTarget</relatedStateVariable>
        </argument>
        <argument>
```

```
        <name>newBlueTarget</name>
        <direction>in</direction>
        <relatedStateVariable>ColorBlueTarget</relatedStateVariable>
      </argument>
    </argumentList>
  </action>
  <action>
    <name>GetColorChangeTarget</name>
    <argumentList>
      <argument>
        <name>retRedTarget</name>
        <direction>out</direction>
        <relatedStateVariable>ColorRedTarget</relatedStateVariable>
      </argument>
      <argument>
        <name>retGreenTarget</name>
        <direction>out</direction>
        <relatedStateVariable>ColorGreenTarget</relatedStateVariable>
      </argument>
      <argument>
        <name>retBlueTarget</name>
        <direction>out</direction>
        <relatedStateVariable>ColorBlueTarget</relatedStateVariable>
      </argument>
    </argumentList>
  </action>
  <action>
    <name>GetColorChangeStatus</name>
    <argumentList>
      <argument>
        <name>RedStatus</name>
        <direction>out</direction>
        <relatedStateVariable>ColorRedStatus</relatedStateVariable>
      </argument>
```

```
        <argument>
          <name>GreenStatus</name>
          <direction>out</direction>
          <relatedStateVariable>ColorGreenStatus</relatedStateVariable>
        </argument>
        <argument>
          <name>BlueStatus</name>
          <direction>out</direction>
          <relatedStateVariable>ColorBlueStatus</relatedStateVariable>
        </argument>
      </argumentList>
    </action>
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="no">
      <name>ColorRedTarget</name>
      <dataType>ui1</dataType>
      <defaultValue>0</defaultValue>
      <allowedValueRange>
        <minimum>0</minimum>
        <maximum>255</maximum>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>ColorGreenTarget</name>
      <dataType>ui1</dataType>
      <defaultValue>0</defaultValue>
      <allowedValueRange>
        <minimum>0</minimum>
        <maximum>255</maximum>
      </allowedValueRange>
    </stateVariable>
    <stateVariable sendEvents="no">
      <name>ColorBlueTarget</name>
```

```
    <dataType>ui1</dataType>
    <defaultValue>0</defaultValue>
    <allowedValueRange>
      <minimum>0</minimum>
      <maximum>255</maximum>
    </allowedValueRange>
  </stateVariable>
  <stateVariable sendEvents="yes">
    <name>ColorRedStatus</name>
    <dataType>ui1</dataType>
    <defaultValue>0</defaultValue>
    <allowedValueRange>
      <minimum>0</minimum>
      <maximum>255</maximum>
    </allowedValueRange>
  </stateVariable>
  <stateVariable sendEvents="yes">
    <name>ColorGreenStatus</name>
    <dataType>ui1</dataType>
    <defaultValue>0</defaultValue>
    <allowedValueRange>
      <minimum>0</minimum>
      <maximum>255</maximum>
    </allowedValueRange>
  </stateVariable>
  <stateVariable sendEvents="yes">
    <name>ColorBlueStatus</name>
    <dataType>ui1</dataType>
    <defaultValue>0</defaultValue>
    <allowedValueRange>
      <minimum>0</minimum>
      <maximum>255</maximum>
    </allowedValueRange>
  </stateVariable>
```

```
    </serviceStateTable>
</scpd>
```