IMPLEMENTASI SENSOR BRIDGE UNTUK PENGENALAN PERVASIVE PERANGKAT SENSOR NON-IP DENGAN UPNP

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Donny Wicaksono
NIM: 115060900111017

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

PENGESAHAN

IMPLEMENTASI SENSOR BRIDGE UNTUK PENGENALAN PERVASIVE PERANGKAT SENSOR NON-IP DENGAN UPNP
SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Donny Wicaksono
NIM: 115060900111017

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

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika Akbar, S.T.,M.Eng.	Achmad Basuki, S.T., MMG., Ph.D
NIP: 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 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, 21 Januari 2016

Donny Wicaksono

NIM: 115060900111017

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, berkat rahmat dan karunia-Nya penulis dapat menyelesaikan penyusunan skripsi dengan baik. Shalawat serta salam semoga senantiasa terlimpahkan kepada Nabi Muhammad SAW.

Penulisan skripsi ini diajukan untuk memenuhi salah satu syarat untuk mendapatkan gelar Sarjana pada Program Studi Sistem Komputer, Fakultas Ilmu Komputer, Universitas Brawijaya Malang. Judul skripsi yang penulis ajukan adalah "IMPLEMENTASI SENSOR BRIDGE UNTUK PENGENALAN PERVASIVE PERANGKAT SENSOR NON-IP DENGAN UPNP".

Dalam penyusunan dan penulisan skripsi ini tidak terlepas dari bimbingan serta dukungan dari berbagai pihak. Oleh karena itu penulis menyampaikan banyak terimakasih kepada yang terhormat:

1. Kedua orang tua yang selalu memberikan doa, motivasi, dukungan moril dan materil sebagai penyemangat selama masa studi.
2. Bapak Adharul Muttaqin, S.T., M.T. selaku Kepala Program Studi Sistem Komputer Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Barlian Henryranu, S.T., M.T. selaku Kepala Laboratorium Sistem Komputer dan Robotika Program Teknologi Informasi dan Ilmu Komputer.
4. Bapak Sabriansyah Rizqika Akbar, S.T., M.Eng. selaku pembimbing skripsi yang telah membimbing dan memberi masukan penulis dalam penyusunan skripsi.
5. Bapak Achmad Basuki, S.T., MMG., Ph.D selaku pembimbing skripsi yang telah membimbing dan memberi masukan penulis dalam penyusunan skripsi.
6. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya selama menempuh pendidikan di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.
7. Teman-Teman Tekom A yang telah menemani selama 4,5 tahun ini menjalani masa perkuliahan.
8. Teman-teman Siskom 2011 yang selalu mendukung secara moral untuk menyelesaikan penulisan skripsi.
9. Teman-teman satu perjuangan skripsi: Maystia, Mbah, Pamungkas, Azmil, Ananto, Jenar, Ericko, Hadist, Yudha, Afif, Labib, Yanuar, Gibrun, Dimel, Kiki, Vynska, Ibrahim, Sukri dan Abah yang telah memberikan bantuan baik ilmu, moral, dan material dalam proses pengerjaan skripsi dan berjuang bersama sepanjang hari.
10. Kepada Reza Laili Choirilia yang telah memberikan dukungan dan semangat selama ini.

11. Kepada seluruh teman teman Asisten Laboratorium Sistem Komputer dan Robotika.
12. Kepada seluruh teman teman Komunitas RoboTIIK.
13. Rekan-rekan mahasiswa Program Studi Sistem Komputer baik angkatan 2011, 2012, 2013, dan 2014 yang selalu berbagi dukungan dan motivasi selama masa perkuliahan.
14. Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu penulis dalam penyusunan skripsi 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, 25 Januari 2016

Penulis

Dhonie56@gmail.com

ABSTRAK

Universal plug and play (UPNP) merupakan sebuah protokol yang menjembatani antar perangkat yang berbeda *vendor* agar mampu diakses melalui jaringan yang sama tanpa perlu melakukan konfigurasi jaringan terlebih dahulu dan juga mampu berkomunikasi satu sama lain tanpa adanya perangkat tambahan. Akan tetapi untuk saat ini kebanyakan perangkat yang dipakai tidak semuanya memiliki IP *address* ataupun *support* untuk terhubung kedalam sebuah *network*. Maka dari itu dibuatlah sebuah *bridge* untuk menghubungkan perangkat non-ip kedalam jaringan yang *support* dengan UPNP.

Sensor Bridge bekerja dengan cara perangkat perangkat non-IP (disini yang digunakan adalah sensor) dihubungkan dengan *Sensor Bridge* melalui XBee. Masing masing sensor tersebut akan mengirim datanya tiap detik ke *Sensor Bridge* sehingga *end user* cukup melakukan akses pada *Sensor Bridge* untuk mengakses servis yang dimiliki semua *device* yang terhubung pada *Sensor Bridge* tersebut.

Pengujian dilakukan dengan melakukan pengaksesan data sensor melalui *control point* dengan *scanning device* servis pada jaringan sehingga diketahui apakah *device* sensor tersebut dapat terbaca pada jaringan yang mendukung UPNP. Setelah *device* ditemukan maka dilakukan akses kepada *device* tersebut untuk menggunakan servis yang disediakan berupa pembacaan nilai sensor, setelah itu hasil data nilai sensor tersebut dicocokan dengan data sensor yang terbaca pada mikrokontroller dan dari *sample* yang diambil datanya adalah sama. Sehinggga dapat disimpulkan implementasi *Sensor Bridge* dapat dilakukan.

Kata kunci: bridge sensor, UPNP, XBee

ABSTRACT

Universal plug and play (UPNP) is a protocol that bridging devices from various vendors to be able accessible and communicate each other from the same network without doing any configuration before. However, not every devices/sensor able to join UPNP network. Therefore, we need a bridge mechanism to add a capability to a devices/sensor to work with UPNP network. We call that capability as bridge sensor.

Bridge sensor works by connecting non-IP device (in this case is sensor) to control point/end user. Each sensor will send data to bridge sensor every second so the end user just access at bridge sensor to access all service of device that connected to bridge sensor.

A test done by accessing data sensor using control point, control point will scanning device service in a network so it will be known that device can be accessed from that network. After that devices is known can be accessed then we will try a test to access that devices service, the service of this devices is provide value number that sensor read, we will match the data that we recive from accesing that sensor from bridge sensor with the original data sensor which is read. From sample that we took there is similarity with perfect accuracy that data from sensor and bridge sensor is same. So can be concluded that bridge sensor can be implemented.

Keywords : Bridge sensor, UPNP, XBee

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Dengan perkembangan jaringan dan teknologi internet banyak sekali peralatan digital dan juga peralatan rumah yang mendukung fungsi *plug and play* pada jaringan rumah tersebut. Peralatan *digital* dan peralatan rumah seperti kamera *digital*, televisi, mesin cuci, kulkas, dll dibuat disesuaikan dengan beragam jenis protokol komunikasi yang telah ada sebelumnya (Shen-Tzong, et al., 2012), salah satunya adalah protokol UPNP.

UPNP (*Universal Plug and Play*) adalah sebuah *open* protokol *auto discovery* dan dapat digunakan pada berbagai macam *device* pada jaringan P2P (Forum, UPNP, 2008). Sehingga memungkinkan berbagai macam vendor terintegrasi menjadi satu tanpa membingungkan keterbatasan protokol yang digunakan. UPNP berbasis *network* terdiri dari beberapa *device* yang dapat dimonitor dan dikontrol oleh satu atau lebih *control point*. Protokol ini memudahkan kita untuk mengakses *device* yang kita miliki melalui jarak jauh.

Namun UPNP sendiri memiliki kelemahan dalam *device* yang digunakan, yaitu harus mendukung protokol UPNP ataupun *device* tersebut harus memiliki IP *address* yang dapat diakses. Permasalahan yang akan muncul adalah apabila *device* tersebut tidak mendukung dengan jaringan UPNP dan juga tidak memiliki IP *address*, hal ini menyebabkan *device* tersebut tidak dapat diakses.

Penelitian sebelumnya membahas tentang menghubungkan *device* yang tidak memiliki IP agar mampu diakses melalui jaringan UPNP. Penelitian tersebut dilakukan oleh Sun-Mi Jun dan Nam-Hoon Park (Jun & Park, n.d.) yang menggunakan *bluetooth* sebagai media penghubung antara *device* yang tidak memiliki ip agar dapat terhubung dan dapat diakses oleh *control point* melalui *bluetooth proxy server*. Namun aplikasi ini hanya dapat menampung 7 buah *device* dan komunikasi dengan *proxy server* dilakukan secara bergantian. Penelitian Sun-Mi Jun dan Nam-Hoon Park masih memiliki beberapa kekurangan, sehingga diperlukan adanya sebuah sistem yang mampu menjembatani antara *device* yang tidak memiliki IP *address* agar *support* dengan protokol UPNP. Selain itu juga sistem tersebut mampu diakses oleh *control point* serta dapat menampung banyak *device* sekaligus dan mampu mengirimkan data secara *realtime*, salah satu cara yang dapat digunakan adalah *Sensor Bridge*.

Sensor Bridge berfungsi untuk menjembatani antara *device* dengan *control point agar device* yang tidak memiliki IP *address support dengan protokol* UPNP dan juga *control point* dapat mengakses data yang dimiliki oleh *device* yang tidak memiliki IP *address* tadi. Sehingga diterapkanlah *Sensor Bridge* sebagai pilihan metode agar *device* yang tidak memiliki IP dapat digunakan.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas maka dapat dirumuskan permasalahan pada skripsi ini sebagai berikut

1. Bagaimana merancang sistem untuk mengakses non-IP device melalui sebuah UPNP *network* ?
2. Apakah *Sensor Bridge* dapat diterapkan sebagai penghubung anatara *device* non-ip dengan UPNP ?

1.3 Tujuan

Tujuan dari pembuatan skripsi ini adalah untuk melakukan dan pengaksesan sensor/*device* yang tidak memiliki IP *address* melalui UPNP *network*, sehingga *control point* mampu mengakses dan melakukan permintaan data dari sensor non-IP atau non-UPNP dan juga sensor non-IP atau non-UPNP mampu mengirim data yang diminta ke *Sensor Bridge* agar dapat diakses oleh *control point*.

1.4 Manfaat

Dari pembuatan skripsi ini didapat manfaat sebagai berikut :

1. Dapat mengurangi ataupun menghilangkan keterbatasan yang dimiliki oleh protokol UPNP.
2. Dapat mengakses ataupun mengontrol *device* non-IP.
3. Memperbanyak pilihan *device* yang dapat diakses ataupun dikontrol.

1.5 Batasan Masalah

Agar pembahasan tidak melebar dari latar belakang dan terfokus, maka diberikan batasan masalah, yaitu:

1. Penelitian ini mengabaikan keamanan jaringan pada UPNP
2. UPNP tidak mengaplikasikan fitur *presentation*
3. Sistem hanya bisa diakses pada jaringan *local*
4. UPNP yang digunakan adalah versi 1.1
5. Elemen pemrosesan pada node sensor berupa Arduino dan pada *Sensor Bridge* adalah Raspberry Pi 1 model B
6. Lingkungan sistem ini diterapkan adalah rumah ataupun gedung tidak bertingkat dan tidak terlalu luas
7. Sistem operasi pada Raspberry Pi adalah Debian Wheezy

1.6 Sistematika Pembahasan

Sistematika penulisan dalam skripsi ini sebagai berikut :

BAB I : Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan skripsi, permasalahan, batasan masalah dan juga sistematika penyusunan skripsi.

BAB II : Landasan Kepustakaan

Bab ini berisi tentang dasar teori yang berkaitan dengan tema dan judul skripsi yang diambil, dan menjadi acuan dasar pengerjaan skripsi ini.

BAB III : Metodologi

Pada bab ini dijelaskan tentang metode untuk menyelsaikan permasalahan yang diangkat dan perancangan system yang akan dibuat.

BAB IV : Perancangan dan Implementasi Sistem

Bab ini membahas implementasi dari system yang dibuat dan disertai dengan potongan-potongan bagian program yang penting dalam sistem yang dirancang.

BAB V : Pengujian dan Analisis

Pada bab ini membahas tentang uji coba dari sistem yang dibuat dengan memperhatikan parameter parameter dan *output* yang dihasilkan oleh sistem tersebut, dan juga dilakukan analisa tentang hasil pengujian tersebut apakah sesuai dengan yang hipotesa atau tujuan yang ingin dicapai.

BAB VI : Penutup

Bab ini berisi kesimpulan dari hasil uji coba yang telah dilakukan dan juga saran saran untuk pengembangan sistem tersebut lebih lanjut lagi.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tinjauan pustaka dan dasar teori yang digunakan. Tinjauan pustaka membahas tentang penelitian penelitian yang ada dan diusulkan. Dasar teori berisi tentang teori yang diperlukan untuk membahas penelitian yang akan dilakukan.

2.1 Kajian Pustaka

Pada penelitian sebelumnya yang dilakukan oleh Sun Mi Jun dan Nam Hoon Park yang berjdul "*Controlling Non-IP Bluetooth Device in UPNP Home Network*" dimana digunakannya *bluetooth proxy server* (sebagai *bluetooth master* dan penghubung ke jaringan UPNP) dan *bluetooth slave* (sebagai penghubung alat non-UPNP ke *bluetooth proxy server*) untuk saling berkomunikasi serta menghubungkan jaringan UPNP dengan jaringan *bluetooth* yang terintegrasi dengan alat-alat non-UPNP. Fungsi utama dari *bluetooth proxy server* ini ada 2 yaitu sebagai tempat disimpannya data servis apa saja yang dimiliki oleh alat-alat non-UPNP yang terhubung dengannya melalui *bluetooth* (Jun & Park, n.d.), sehingga meminimalkan proses hubungan langsung dari *control point* menuju ke dalam jaringan *bluetooth slave* dan juga sebagai jembatan penghubung antara jaringan UPNP dengan alat-alat non-UPNP.

Selain itu pada penelitian yang dilakukan oleh Drazen Pasalic, Dusanko Bundalo, Branimir Civijic dan Zlatko Bundalo yang berjudul "*XBee based Data Transmission and Monitoring Wireless Sensor Network Integrated with the Intenet*" membahas tentang pengiriman data sensor menggunakan XBee ke sebuah koordinator *node* yang tehubung dengan sebuah *web server* sehingga data sensor tersebut dapat diakses secara *real time* darimanapun dan kapanpun (Drazen, et al., 2015). Dalam jurnal tersebut mengkaji mengenai pengiriman data sensor pada *end node* berupa Arduino yang terhubung dengan XBee dan kemudian data sensor dikirimkan ke XBee koordinator yang terhubung langsung dengan Raspberry Pi untuk diolah dan disimpan pada sebuah *database webserver* untuk diakses.

Pada penelitian yang dilakukan akan membahas mengenai *Sensor Bridge* sebagai penghubung antara perangkat UPNP berupa *control point* dengan sensor non UPNP, sehingga kita tetap dapat menggunakan atau sensor-sensor yang telah ada di pasaran selama ini. Terdapat beberapa perbedaan antara kajian pustaka yang digunakan dengan penelitian yang dilakukan diantaranya adalah, pada penggunaan media transmisi dan penghubung antara sensor dan *Sensor Bridge* menggunakan XBee, serta penerapan UPNP sebagai *protocol* untuk kontrol dan akses data sensor.

2.2 Dasar Teori

Pada dasar teori ini akan membahas dasar-dasar yang digunakan dalam pembuatan sistem. Dasar teori yang digunakan antara lain adalah *mikrokontroller* Arduino, XBee, protokol UPNP, sensor dan minikomputer Raspberry Pi.

2.2.1 UPNP

Universal Plug and Play atau biasa disingkat UPNP adalah sebuah protokol yang menjembatani antar peralatan cerdas, *wireless devices*, dan PC yang berasal dari berbagai macam dan jenis vendor yang berbeda beda, sehingga mampu bekerja sama dan saling berkomunikasi (Dononoho, et al., 2015). Selain itu penggunaan arsitektur UPNP ini juga mempermudah pengguna untuk terhubung dan berkomunikasi dengan apapun di dalam jaringan IP, baik itu di rumah, kantor, mobil, atau melalui internet (Forum UPNP, 2015).

Komponen utama dalam UPNP sendiri adalah sebuah *control point*, dan minimal satu *device* yang dikontrol ataupun diakses. Fungsi dari *control point* untuk mengenali dan mengontrol *device* yang terhubung dengannya dalam satu jaringan. Sedangkan *device* berfungsi sebagai sebuah perangkat ataupun alat yang akan diakses maupun dikontrol. Dalam pengoperasianya terdapat 5 fase mulai dari pengenalan *device* sampai pengontrolan ataupun pengaksesannya yang akan dijelaskan di bawah ini.

1. Discovery
   Jika sebuah device terhubung ke dalam jaringan yang mendukung UPNP maka *device* akan mengirim *discovery message* secara *multicast* yang berisi *description url*. Jika *control point* yang terhubung ke dalam jaringan maka akan dilakukan pengiriman *discovery message* yang mencari *device* yang *support* UPNP dan terhubung di dalam jaringan tersebut. Proses *discovery* yang dilakukan oleh *control point* dan *device* menggunakan protokol yang bernama SSDP (*Simple Service Discovery Protocol*). Protokol SSDP didalmnya berisi *start-line* sebagai penanda bahwa sedang melakukan pencarian *device* ataupun mengumumkan keberadaan sebuah *device*. Jika pencarian *device* maka pada bagian start-line akan berisi M-SEARCH dan jika sedang mengumumkan keberadaan sebuah *device* maka berisi NOTIFY.
2. Description
   Setelah *control point* mendapatkan *discovery message*, *control point* akan melakukan *request* ke alamat *description url* yang dibawa oleh *discovery message* untuk mengetahui tentang *device* dan layanan apa saja yang tersedia pada *device* tersebut, *control point* akan mengirimkan http *get request* kepada alamat URL yang kemudian akan dijawab oleh *device* berupa pengiriman file XML yang berisi informasi berupa nama *device*, nama perusahaan pembuat alat tersebut, dan juga URL yang digunakan untuk melakukan akses kepada layanan yang disediakan oleh *device* tersebut.
3. Control

Dengan alamat URL yang diberikan tadi *control point* mampu melakukan akses ataupun kontrol terhadap *device*, seperti melakukan *invoking value* untuk melakukan kontrol menghidupkan atau mematikan lampu dengan mengganti nilai kontrol lampu menjadi 0 ataupun 1. Cara *invoking* nilai ini dilakukan dengan mengirim *control message* kepada *device*.

4. Event
   Publish update dilakukan apabila terdapat perubahan pada servis ataupun *device*-nya. Sehingga kita akan tahu status dari *device* tersebut. Jika tiba-tiba *device* ataupun layanan tersebut hilang ataupun kembali terhubung dan bisa diakses kembali.
5. Presentation
   Jika *device* memiliki alamat URL untuk *redirect* ke sebuah alamat *website*, yang memungkinkan akses kontrol melalui web browser sehingga kita bisa melakukan kontrol, akses *device* lebih mudah dengan tampilan yang menarik (Hwantae, et al., 2012).

2.2.2 Raspberry Pi

Raspberry Pi merupakan sebuah *board* mikrokomputer yang dapat tersambung ke monitor atau TV dan mampu terhubung dengan *mouse* dan *keyboard* standar pada umumnya. Raspberry Pi sendiri memiliki spesifikasi prosesor ARM v6 dengan *clockspeed* 700Mhz didukung dengan RAM sebesar 512 MB, dan dilengkapi dengan Micro SD slot. Dan juga fitur-fitur tambahan seperti 4 buah *USB port*, 40 pin GPIO *port*, *ethernet port*, *full HDMI port,* 3.5mm *audio jack*, *camera interface* (CSI), *display interface* (DSI) (Pi, n.d.). Berikut pada Gambar 2.1 ditunjukkan bentuk Raspberry Pi model B serta penjelasan detail perbagiannya.

Gambar 2.1 Raspberry Pi

Sumber: (Rice, 2013)

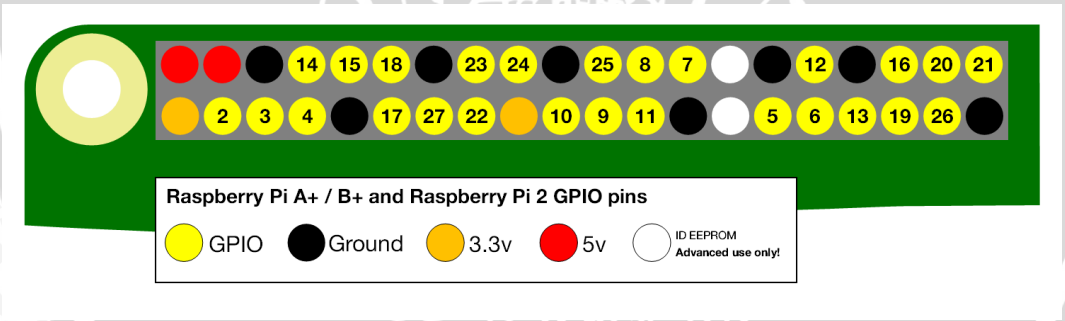
1. *Secure Digital* (SD) *Card Slot*

Slot ini digunakan sebagai tempat menancapkan SD card dimana OS Raspberry P Pi disimpan.

2. *Universal Serial Bus* (USB)
   Terdapat 2 buah *port* USB untuk keperluan menghubungkan Raspberry Pi dengan alat melalui *USB port*.
3. LAN *Port*
   Untuk menghubungkan Raspberry Pi dengan alat ataupun jaringan melalui konektor RJ45.
4. *High Devinition Multimedia Interface* (HDMI)
   Sebagai *port output* untuk menampilkan gambar dari Raspberry Pi.
5. RCA *Video*
   Fungsinya sama dengan *port* HDMI sebagai port output untuk menampilkan video dari Raspberry Pi namun dengan kualitas video lebih rendah dari HDMI.
6. *Power Port*
   Berfungsi sebagai sumber tegangan untuk Raspberry Pi.
7. Status LED
   Berfungsi sebagai tanda atau status proses yang sedang dijalankan oleh Raspberry Pi.

2.2.2.1 GPIO

GPIO (*general purpose input output*) adalah pin penghubung antara Raspberry Pi dengan peralatan lain berupa sensor, motor, dll. Pin tersebut dapat diprogram agar berfungsi sebagai input ataupun output. Raspberry Pi memiliki 40 buah pin GPIO, 26 diantarannya adalah pin *input/output* dan sisanya adalah pin VCC dan *ground* (Pi, n.d.) seperti yang ditunjukkan oleh Gambar 2.2 berikut.



Gambar 2.2 Pin GPIO pada Raspberry Pi

2.2.3 XBee

XBee adalah sebuah modul *radio frequency* (RF) yang mendukung standar IEEE 802.15.4, dan juga XBee mendukung kebutuhan penggunaan kebutuhan hardware RF yang *low cost* dan juga *low power wireless sensor network*, sehingga mampu menghemat konsumsi power tetapi tetap *reliable* dalam pengiriman datanya (Digi, 2015). Modul ini beroprasi pada frekuensi 2,4 GHz. Untuk spesifikasi lengkapnya seperti dibawah ini, serta disertakan juga gambar bentuk fisik dari XBee seperti yang ditunjukkan oleh Gambar 2.3:

- Indoor/Urban: up to 100'' (30 m)
- Outdoor line-of-sight: up to 300'' (90 m)
- Transmit Power: 1 mW (0 dBm)
- Receiver Sensitivity: -92 dBm

- TX Peak Current: 45 mA (@3.3 V)
- RX Current: 50 mA (@3.3 V)
- Power-down Current: < 10 µ

Gambar 2.3 XBee

Sumber: (Jerome, 2014)

XBee sendiri memiliki 3 jenis tipe fungsi yang dapat diubah atau diprogram sesuai dengan kebutuhan yaitu sebagai *cordinator*, *router* ataupun *node* dimana setiap fungsi tersebut memiliki kegunaan masing masing.

Cordinator:

- Menentukan *chanel* gelombang mana yang akan digunakan untuk berkomunikasi
- Dapat memberikan ijin apabila ada *router* ataupun *end device* yang ingin bergabung ke dalam jaringan
- Dapat membantu proses *routing* data
- Tidak dapat mengaktifkan fungsi *sleep*
- Dapat menyimpankan data sementara di memori buffernya untuk *node end device* yang sedang dalam *mode sleep* dengan syarat *node* tersebut terhubung denganya secara langsung.

Router:

- Harus bergabung dengan jaringan yang dibentuk oleh *coordinator* melalui *router* ataupun *cordinator*
- Dapat memberikan ijin kepada *router* lain ataupun *end device* untuk bergabung dalam jaringannya
- Melakukan proses *routing* data
- Tidak dapat mengaktifkan fungsi *sleep*
- Dapat menyimpankan data sementara di memori buffernya untuk *node end device* yang sedang dalam mode *sleep* dengan syarat *node* tersebut terhubung denganya secara langsung

End Device:

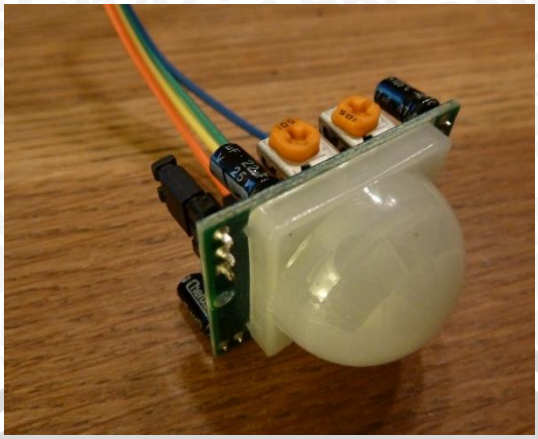
- Harus bergabung dengan jaringan yang dibentuk oleh *cordinator* melalui router ataupun *cordinator*
- Tidak mempunyai hak untuk memberikan ijin kepada *device* lain yang ingin bergabung ke dalam jaringan
- Tidak dapat meneruskan data
- Dapat mengaktifkan mode *sleep* dimana *end device* akan berada di kondisi *sleep* untuk beberapa waktu sehingga mampu menghemat penggunaan daya

Selain itu untuk pengalamatan yang digunakan untuk berkomunikasi dalam XBee dinamakan PAN (*Personal Area Network*) ID. PAN ID pada XBee berupa alamat sepanjang 16 bit akan tetapi karena keterbatasan jumlah alamat yang dapat dibentuk oleh 16 bit maka XBee juga menyediakan alamat PAN ID 64 bit yang bertujuan apabila terdapat kesamaan alamat 16 bit maka dapat dikenali menggunakan alamat 64 bit yang dimiliki. Untuk memperoleh alamat 16 bit dan 64 bit sendiri terdapat 2 cara yaitu dengan melakukan konfigurasi alamat secara manual sebelum *join* dengan *network* ataupun mendapatkan alamat *random* secara otomatis ketika bergabung dengan sebuah network (Digi, 2015).

2.2.4 Sensor Pasif Infrared

Sensor Pasif infrared (PIR) bekerja dengan cara membaca panas inframerah di area yang dijangkaunya. Sensor PIR membutuhkan waktu 10-60 detik untuk kalibrasi sehingga dalam jangka waktu tersebut jangan menggunakan sensor untuk melakukan pembacaan gerakan (Anon., n.d.).

Sensor ini membutuhkan tegangan sebesar 5V-20V dan memiliki kemampuan jangkau deteksi kurang dari 7 meter dan memiliki lebar sudut baca sebesar 120^{o}C dan juga dapat diatur sensitifitas dan *delay* pembacaan sensornya mulai dari 0 sampai dengan 5 menit. Berikut merupakan bentuk fisik sensor PIR yang ditunjukkan oleh Gambar 2.4:

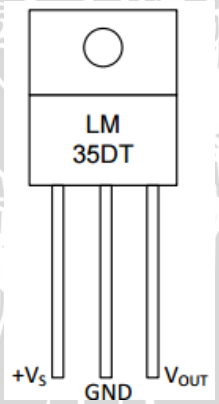


Gambar 2.4 Sensor Pasif Infrared

Sumber: (Roman, 2013)

2.2.5 Sensor LM35

Sensor LM35 berfungsi untuk mengukur suhu ruangan dimana tegangan keluarannya berbanding lurus dengan tempratur. Sensor LM35 besaran yang diubah adalah panas ke listrik. Sensor ini dapat mendeteksi suhu dengan range suhu -55 sampai 150^{O}C (Conductor, 2000). Sensor ini juga dapat dikalibrasi suhunya menjadi Kelvin sehingga memudahkan pengguna untuk pemakaiannya dan dapat disesuaikan dengan kebutuhanya, berikut merupakan gambar fisik sensor tersebut seperti ditunjukkan pada Gambar 2.5.

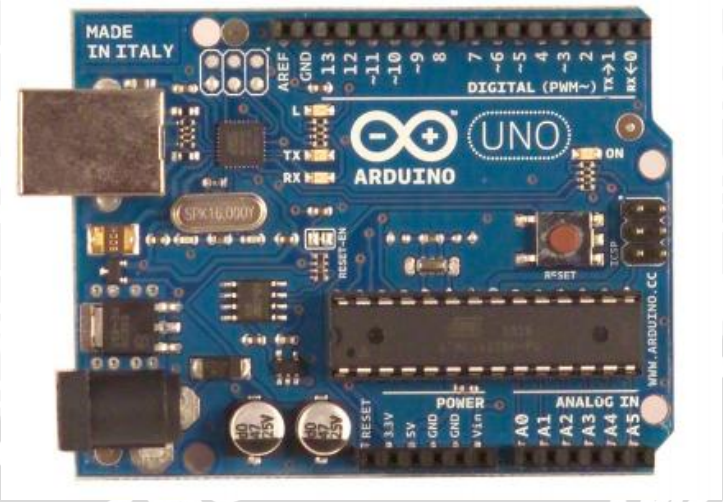


Gambar 2.5 Sensor LM35

(Instrument, 2015)

2.2.6 Arduino

Arduino adalah salah satu vendor *microcontroller* yang dibuat berdasarkan ATMega 328. Arduino sendiri memiliki 14 pin *digital* dimana 6 dari pin tersebut juga dapat berfungsi sebgai PWM *output*, 6 buah pin *analog*, 16 MHz *ceramic resonator*, USB *conection*, *power jack*, ICSP *header* dan *reset* button (Arduino, n.d.), berikut merupakan spesifikasinya secara rinci beserta gambar fisiknya seperti yang ditunjukkan oleh Gambar 2.6.



Gambar 2.6 Bagian Arduino

Sumber: (Arduino, n.d.)

1. 14 pin *input*/*output* digital

   Ke empat belas pin tersebut dapat deprogram menjadi *input* ataupun *output digital*, dan untuk 6 buah pin yaitu pin ke 3,5,6,9,10,11 juga dapat digunakan untuk *output analog* dimana nilai tegangan *output* dapat diatur, nilai 0-5v diwakili dengan nilai antara 0 sampai 255.

2. USB

   Berfungsi untuk memuat program dari komputer kedalam papan selain itu berfungsi sebagai media komunikasi serial antara papan dan komputer juga memberi daya listrik ke papan.

3. Q1 (kristal)

   Berfungsi sebagai *trigger* yang mengirimkan pemicu untuk menjalankan sebuah operasi.

4. Tombol Reset

   Berfungsi untuk melakukan *reset* program agar kembali jalan mulai dari awal program.

5. ICSP

   In Circuit Serial Programing berfungsi untuk pengguna memprogram Arduino secara langsung tanpa melalui *bootloader.*

6. IC1

   Komponen utama yang berupa *mikrocontroller* ATMega sebagai pengolah data di Arduino.

7. X1

   Sebagai sumber daya eksternal untuk Arduino yang biberikan pada tegangan 9 -12 v.

8. 6 pin *input*/*output* analog

   Dapat diprogram untuk menjadi *input* ataupun *output* pada alat elektronik yang membaca atau menghasilkan keluaran berupa sinyal *analog*.

2.2.7 Library GUPNP dan XBee

GUPNP adalah sebuah *framework* yang digunakan untuk membuat UPNP *devices* dan *control point* yang ditulis dalam bahasa C (Jonsson, 2015). Di dalam GUPNP sendiri terdapat beberapa *library* seperti gssdp, GUPNP, GUPNP-av, GUPNP dlna. Framework ini berguna untuk pembuatan UPNP objek, *device*, *control point*, aplikasi UPNP *audio video* dengan menggunakan bahasa C (Jonsson, 2015).

Selain itu digunakan *library* XBee pada Arduino dan Raspberry Pi. Disini terdapat 2 jenis *library* XBee yang digunakan yaitu XBee.h pada Arduino dan XBee pada Python, *library* tersebut menjembatani antara perangkat Arduino dengan XBee dan Raspberry Pi dengan XBee agar mampu berkomunikasi. Berikut pada tabel 2.1, 2.2 dan 2.3 terdapat beberapa fungsi yang digunakan.

Tabel 2.1 Fungsi yang digunakan pada library GUPNP

No.	Fungsi	Kegunaan	Library
1	gupnp_context_new	Membuat gUPNP context baru	G()UPNP
2	gupnp_root_device_new()	Membuat root device baru pada gUPNP	GUPNP
3	gupnp_device_info_get_Service()	Untuk mendapatkan info tentang servis dari sebuah device	GUPNP

Sumber: Jonsson (2015)

Tabel 2.2 Fungsi yang digunakan pada library XBee

No.	Fungsi	Kegunaan	Library
1	XBee.wait_read_frame()	Menunggu dan membaca frame data yang lewat pada serial line	XBee
2	XBee = ZigBee(init_serial(), escaped=True)	Membuat API object XBee	XBee

Sumber: Malmsten (2011)

Tabel 2.3 Fungsi yang digunakan pada library XBee.h

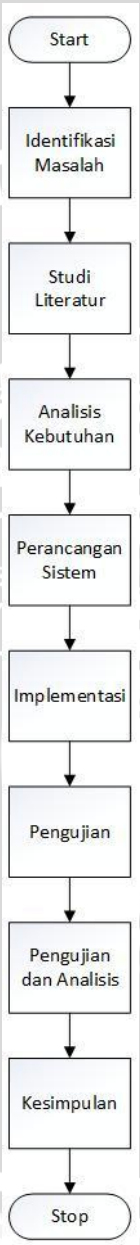
No.	Fungsi	Kegunaan	Library
1	uint8_t payload[] = { 0, 0 }	Membuat array payload untuk menyimpan data yang akan dikirim	XBee.h
2	XBeeAddress64 addr64 = XBeeAddress64(0x00000000, 0x00000000)	Pengesetan alamat tujuan pengiriman pada XBee	XBee.h
3	ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload))	Membuat Tx request yang akan dikirimkan	XBee.h
4	XBee.send(zbTx)	Kirim Tx Request	XBee.h

Sumber: Wrapp (2012)

BAB 3 METODOLOGI

3.1 Metodologi Penelitian

Pada bab ini akan dijelaskan tentang langkah-langkah yang akan ditempuh dalam penyusunan skripsi seperti yang ditunjukkan oleh Gambar 3.1 dimana meliputi studi kasus, studi literatur, analisis kebutuhan sistem, perancangan sistem, pengujian dan analisis, kesimpulan dan saran.



Gambar 3.1 Diagram alir metode penelitian

3.1.1 Studi dan Pengkajian Literatur

Studi literatur yaitu adalah kegiatan pengumpulan, pencarian bahan-bahan yang dapat digunakan sebagain referensi dan penunjang dalam penyusunan skripsi. Jenis-jenis literatur yang dapat digunakan sebagai penunjang dan penyusunan skripsi sendiri dapat berupa, jurnal, buku, dokumentasi resmi dari vendor atau suatu merek serta beberapa referensi lainya yang berhubungan dengan tema yang diangkat dimana disini adalah UPNP. Referensi yang digunakan dalam perancangan sistem ini adalah UPNP forum dan juga *website* GUPNP.

3.1.2 Analisis Kebutuhan

Dari studi literatur yang telah dilakukan, maka akan dilakukan analisis mengenai kebutuhan fungsional sistem yang akan dibuat atau dirancang.

3.1.2.1 Kebutuhan Fungsional

1. Sensor mampu mengirim *update* informasi nilai yang dibacanya pada *Sensor Bridge*.
2. *Sensor Bridge* dapat melakukan memberitahukan atau *advertising* masing-masing servis dari tiap sensor atau *node* kepada *client* atau *control point*.
3. *Sensor Bridge* mampu menerima permintaan untuk melakukan pengaksesan nilai data sensor pada saat itu.

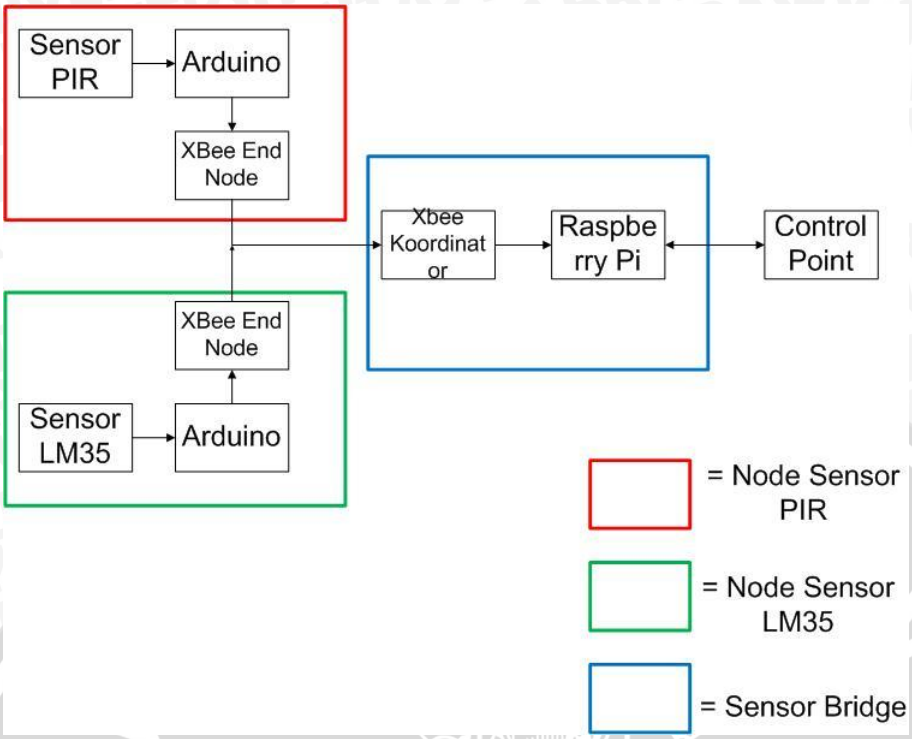
3.1.2.2 Kebutuhan Non Fungsional

Analisa kebutuhan non fungsional dilakukan untuk mengetahui kebutuhan sistem yang dirancang, analisa tersebut meliputi spesifikasi dari sisi *software* maupun *hardware*.

1. 1 buah Raspberry Pi sebagai pemrosesan data, dan meneruskan *request* yang diminta oleh *client.*
2. 3 buah XBee yang berfungsi sebagai penghubung antara sensor dengan Raspberry Pi sebagai pusat pemrosesan.
3. 1 buah sensor passive infrared (PIR) dan sensor LM35 sebagai sensor sebagai penyedia servis yang akan diakses datanya.
4. 1 buah laptop sebagai *client* berfungsi untuk mengakses servis yang tersedia.
5. 2 buah Arduino sebagai *microcontroller* yang akan membaca dan mengolah data yang didapat dari sensor.

3.1.3 Perancangan Sistem

Perancangan sistem dilakukan setelah terpenuhinya analisa kebutuhan sistem yang telah dibuat. Berdasarkan kebutuhan fungsional yang telah dirancang sebelumnya, maka perancangan sistem yang akan dibuat dapat dilihat pada diagram blok pada Gambar 3.2.



Gambar 3.2 Diagram blok sistem

Pada Gambar 3.2 dapat dilihat bahwa terdapat blok berwarna merah dan hijau yang merupakan *node* sensor PIR dan LM35 di dalam blok *node* sensor berisi sensor, Arduino dan XBee. Terdapat juga blok berwarna biru yang merupakan *Sensor Bridge*, pada blok *Sensor Bridge* berisi XBee dan Raspberry Pi. Sistem ini akan bekerja dengan cara *node* sensor mengirimkan datanya pada *Sensor Bridge* yang kemudian *Sensor Bridge* akan memulai memberitahukan layanan yang ada pada *node* sensor. Dan apabila *control point* tertarik dengan layanan yang dimiliki oleh *node* sensor, *control point* cukup mengaksesnya melalui *Sensor Bridge* dan oleh *Sensor Bridge* akan mengirimkan data yang telah dibaca dan dikirimkan oleh *node* sensor sebelumnya.

3.1.4 Implementasi

Implementasi dilakukan dengan mengacu pada perancangan sistem yang dibuat sebelumnya, dan juga pada implementasi akan ditampilkan potongan - potongan program yang digunakan. pada sistem ini terdapat beberapa implementasi yang akan diterapkan yaitu adalah implementasi sensor PIR dan juga LM35, implementasi XBee sebagai modul komunikasinya, implementasi UPNP *device*, UPNP servis dan juga *Sensor Bridge*.

3.1.5 Pengujian dan Analisis

Pengujian akan dilakukan dalam beberapa tahapan yaitu pengujian fungsional, pengujian UPNP dan pengujian akurasi, dimana semua pengujian tersebut dilaksanakan pada ruangan laboratorium National Instrumen. Pengujian fungsional berfungsi untuk melihat apakah kebutuhan fungsional sistem berjalan sesuai dengan perancangan atau tidak. Pengujian fungsional dilakukan dengan

cara mengecek fungsi utama *Sensor Bridge* yaitu meneruskan data yang dibaca oleh sensor ke *control point*, sehingga akan dilakukan pembandingan data dari nilai yang dibaca oleh sensor dengan yang ditampilkan oleh *control point* saat mengakses *device* tesebut. Selain itu dilakukan juga pengujian akurasi berfungsi untuk melihat bagaimana data yang dikirimkan oleh *node* sensor ke *Sensor Bridge* apakah terjadi keterlambatan penerimaan data atau bahkan data yang dikirimkan tidak sampai. Data yang dikirim berupa urutan nomer paket yang dikirimkan dari 1 hingga 20 kemudian dibandingkan dengan data yang diterima pada *Sensor Bridge* sehingga diketahui apabila ada paket yang telat atau hilang.

3.1.6 Kesimpulan dan Saran

Kesimpulan akan diambil dari hasil pengujian dan analisis sistem yang telah dirancang. Isi dari kesimpulan tersebut akan berguna bagi orang lain yang ingin mengembangkan, menerapkan sistem ini atau juga dapat digunakan sebagai referensi untuk sistem dengan topik yang sama.

BAB 4 PERANCANGAN DAN IMPLEMENTASI SISTEM

Bab ini akan membahas tentang perancangan dari sistem yang akan dibuat berdasarkan hasil perancangan tersebut akan dilakukan implementasi sistem yang terdiri dari implementasi perangkat lunak dan perangkat keras.

4.1 Perancangan Sistem

Secara umum sistem yang akan dibangun dibagi menjadi 2 yaitu:

1. Perangkat lunak yang digunakan untuk menjembatani antara sensor yang tidak memiliki IP *address* dengan *control point* sehingga tetap dapat diakses, selain itu juga sebagai media penghubung agar *control point* tetap dapat mengakses sensor meskipun tidak terhubung secara langsung.
2. Perangkat keras yang digunakan untuk melakukan pembacaan nilai sensor serta menghubungkannya dengan *Sensor Bridge*.

Proses perancangan ini dibagi menjadi beberapa tahapan yaitu, analisa kebutuhan sistem, perancangan diagram blok sistem, perancangan perangkat lunak dan juga perancangan perangkat keras serta pengujian sistem yang telah dibuat.

4.1.1 Analisa Kebutuhan Sistem

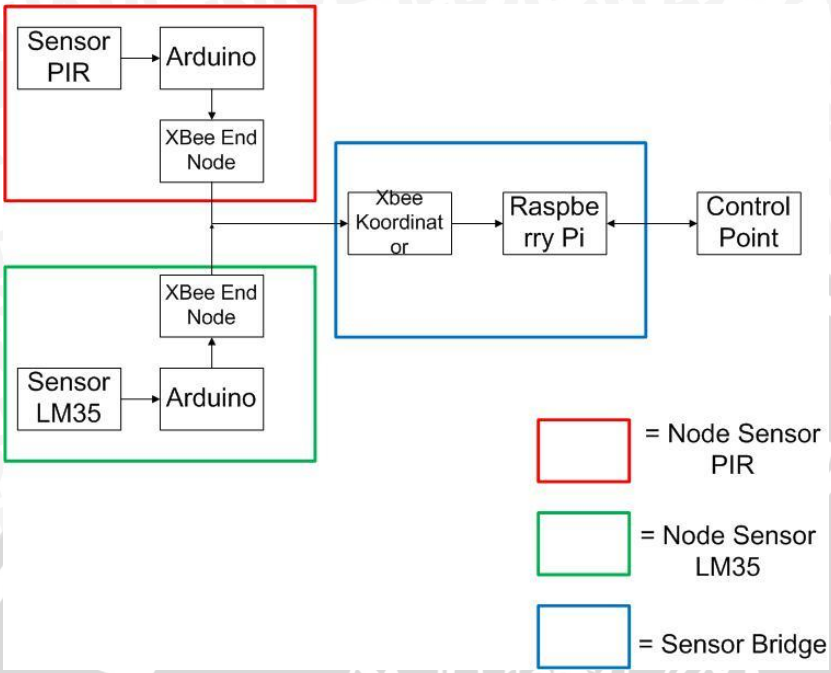
Analisa kebutuhan yang dibutuhkan seperti perangkat keras dan perangkat lunak yang akan digunakan untuk proses implementasi dari sistem yang ada, diantaranya yaitu:

- Arduino sebagai *microcontroller* yang akan membaca dan mengolah data yang didapat dari sensor.
- Raspberry Pi sebagai pemrosesan data, dan meneruskan *request* yang diminta oleh *client*.
- XBee berfungsi sebagai penghubung antara sensor dengan Raspberry Pi sebagai pusat pemrosesan.
- Sensor Passive Infrared sebagai penyedia data untuk layanan yang diberikan oleh sensor PIR.
- Sensor LM35 sebagai penyedia data untuk layanan yang diberikan oleh sensor LM35.
- DC Voltage sebagai sumber tegangan eksternal untuk sensor pada Arduino.
- PC/Komputer sebagai *control point* untuk mengakses nilai yang dikirimkan oleh sensor.

4.1.2 Diagram Blok Sistem

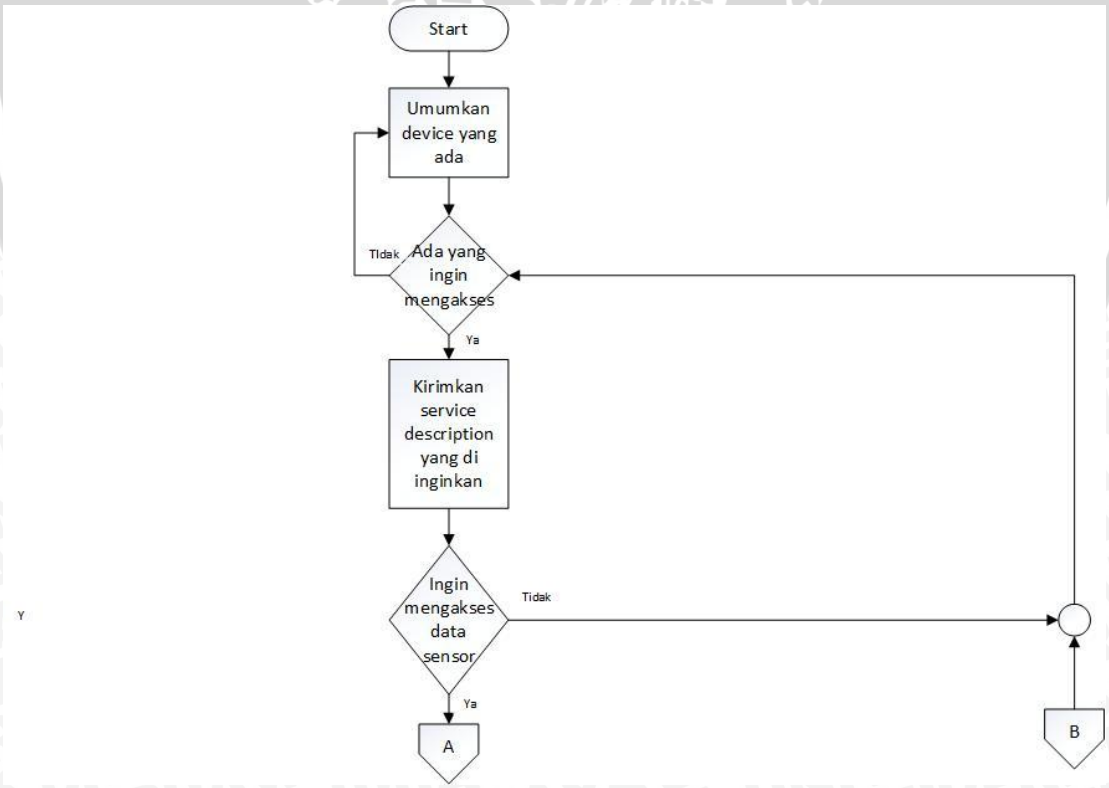
Dari analisa kebutuhan sistem yang telah dijelaskan sebelumnya maka dibuatlah diagram blok kerja sistem. Diagram blok sistem ditunjukkan oleh Gambar 4.1 dimana *input* berupa 2 buah sensor yaitu Sensor Pasif infrared (PIR) dan juga LM35 yang kemudian data tersebut diolah oleh Arduino, setelah data sensor selesai diolah kemudian dikirimkan pembacaaan data sensor tersebut

melalui XBee ke sisi XBee yang lain (koordinator). Data yang telah diterima dari XBee akan diolah ulang oleh *Sensor Bridge* agar dapat diakses oleh *control point*.

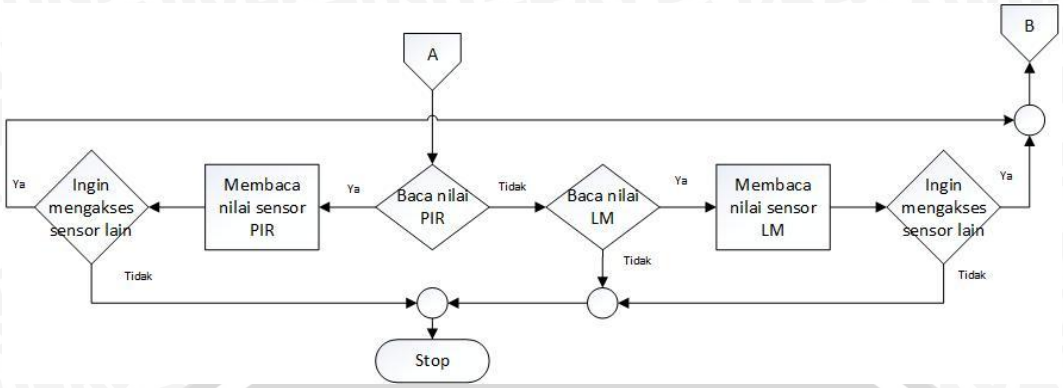


Gambar 4.1 Diagram blok sistem

4.1.3 Diagram Alir Keseluruhan



Gambar 4.2 Diagram alir sistem keseluruhan

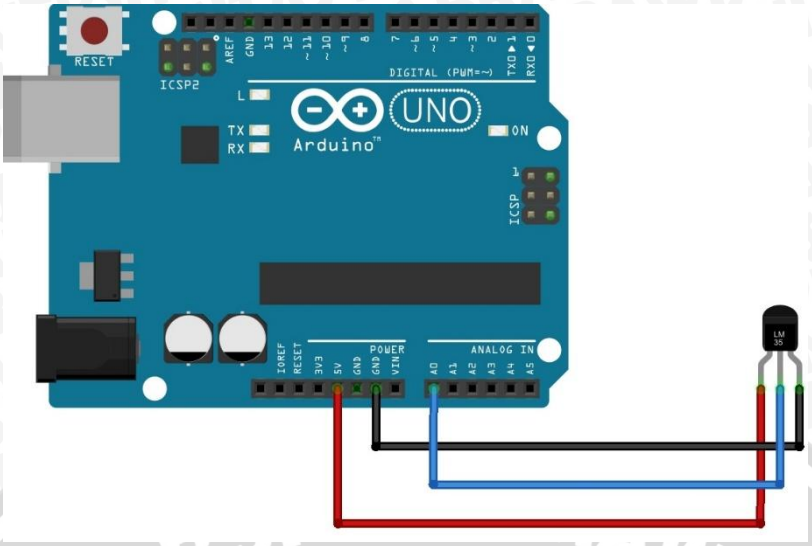


Gambar 4.3 Diagram alir sistem keseluruhan (lanjutan)

Pada Gambar 4.2 dan 4.3 ditunjukkan bahwa sistem ini dimulai dengan *Sensor Bridge* melakukan pengumuman atau *advertise* tentang layanan yang dimiliki oleh *device* non UPNP yang terhubung kepadanya dengan cara mengirimkan *device description* yang dimilikinya, agar diketahui oleh *control point*. Dan jika ada *control point* yang ingin mengakses *device* tersebut, maka *control point* akan meminta dokumentasi tentang informasi dan kemampuan apa yang dapat dilakukan (layanan yang disediakan) oleh *device* tersebut, maka *Sensor Bridge* akan mengirimkan dokumen *device description* yang berisi data-data tersebut pada *control point*. Setelah itu apabila *Sensor Bridge* menerima permintaan untuk akses layanan *device* tersebut, dari permintaan ijin akses akan terlihat sensor mana yang ingin diakses datanya apakah PIR atau LM35. Apabila yang ingin diakses oleh *control point* adalah sensor PIR, maka *Sensor Bridge* akan mengirimkan nilai terbaru yang dibaca dan dikirimkan oleh sensor PIR kepada *control point*. Dan apabila yang ingin diakses adalah sensor LM35 maka *control point* akan mengirmkan nilai suhu terbaru yang dikirimkan oleh sensor tersebut ke *control point*.

4.1.4 Perancangan Sensor LM35

Sensor LM35 berfungsi untuk mengetahui temperatur ruangan disekitarnya. Cara kerjanya dengan mengubah besaran panas menjadi listrik. Sensor ini akan dihubungkan dengan Arduino seperti pada Gambar 4.4.

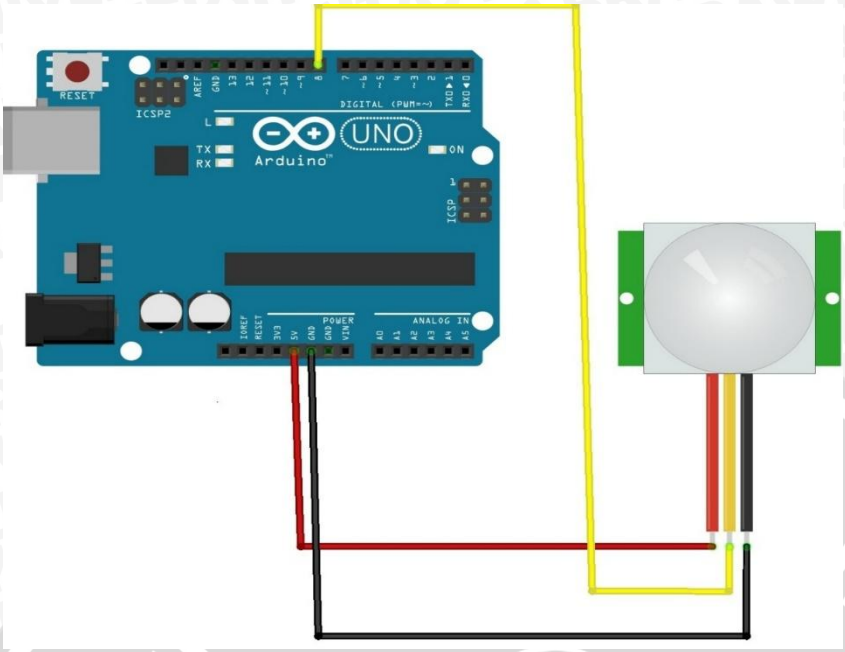


Gambar 4.4 Rangkaian sensor LM35 ke Arduino

Pada Gambar 4.4 terlihat LM35 memiliki 3 buah kaki yaitu VCC, data/*output*, dan *ground*, dimana masing-masing kaki tersebut terhubung pada Arduino. Kaki VCC pada LM35 terhubung dengan pin VCC 5v pada Arduino (dihubungkan dengan kabel merah), dan *ground* pada LM35 dihubungkan dengan *groun*d pada Arduino (dihubungkan dengan kabel hitam) dan pin *output* dihubungkan dengan pin *analog* A0 (ditunjukkan dengan kabel biru).

4.1.5 Perancangan Sensor Passive Infrared

Sensor *passive infrared* (PIR) bekerja dengan cara sensor ini cara menangkap energi panas yang dihasilkan dari pancaran sinar inframerah yang dimiliki setiap benda dengan suhu benda di atas nol mutlak.

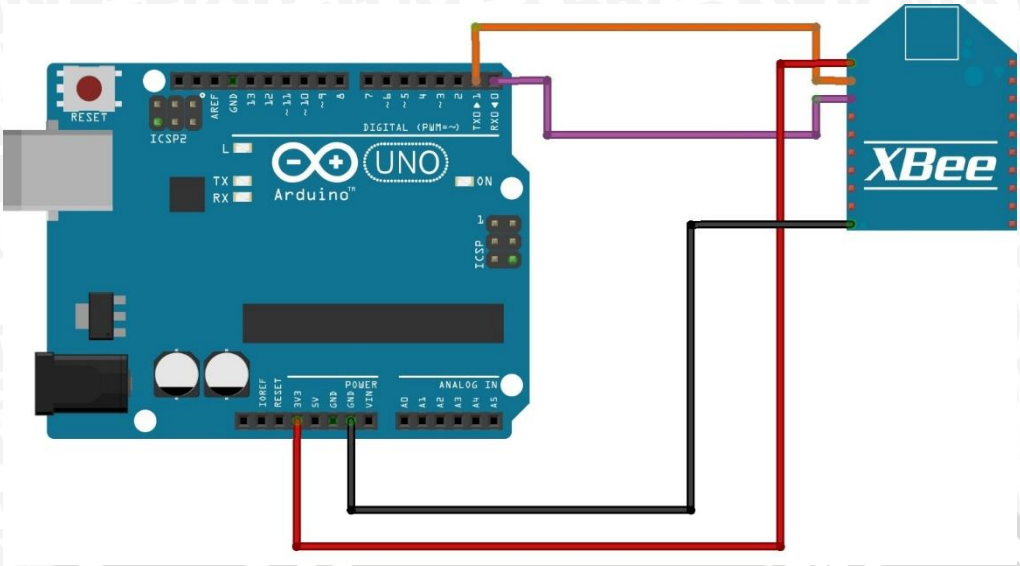


Gambar 4.5 Rangkaian sensor PIR ke Arduino

Pada Gambar 4.5 diperlihatkan bahwa sensor PIR dihubungkan ke Arduino UNO, dimana pin VCC pada sensor PIR dihubungkan (kabel warna merah) dengan pin 5v pada Arduino. Untuk pin *ground* pada PIR dihubungkan (kabel warna hitam) dengan *ground* pada Arduino sedangkan untuk pin *output* data pada sensor pir dihubungkan (kabel warna kuning) dengan *input* Arduino pada pin *digital* yaitu pin 8, hal ini dikarenakan *output* nilai dari sensor PIR berfungsi sebagai masukan data yang akan diolah oleh Arduino.

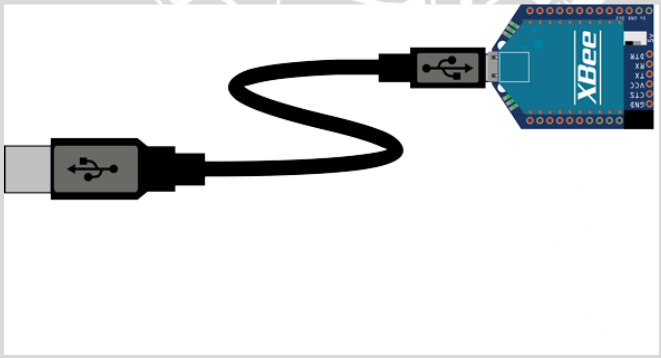
4.1.6 Perancangan XBee

XBee adalah sebuah modul *Radio Frequency* (RF) yang bekerja pada frekuensi 2,4 GHz dan 900 MHz dengan kecepatan transfer data 250 kbps, selain itu XBee bekerja dengan 2 mode yaitu API dan AT command. XBee dihubungkan dengan Arduino berfungsi untuk mengirimkan nilai sensor yang telah diolah oleh Arduino untuk dikirimkan menuju Raspberry Pi sehingga data dapat terupdate secara berkala sesuai dengan yang dikirimkan oleh Arduino. Pada perancangan XBee dibagi menjadi dua yaitu perancangan pada XBee *router* dan juga XBee koordinator.



Gambar 4.6 Rangkaian XBee ke Arduino

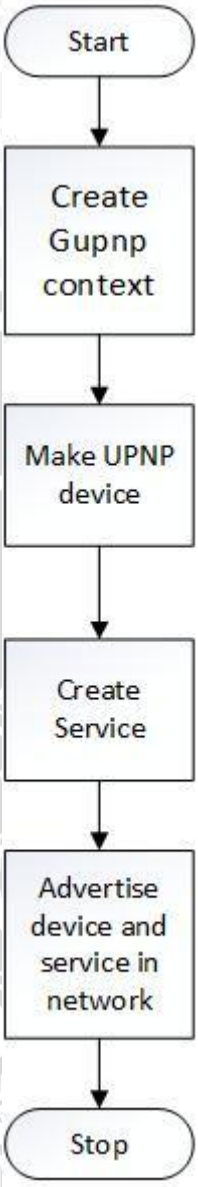
Pada XBee *router* yang seperti terlihat pada Gambar 4.6 bahwa VCC XBee terhubung dengan vdd 3,3v dari Arduino (kabel warna merah) dan untuk *ground* terhubung langsung dengan *ground* dari Arduino (kabel warna hitam). Untuk Tx Arduino terhubung kepada Tx XBee (kabel warna orange) begitu pula RX XBee terhubung langsung deng Rx XBee (kabel warna ungu). Untuk konfigurasi pada sisi koordinator dimana XBee terhubung dengan Raspberry Pi melalui USB mini to USB pc, XBee dihubungkan dengan XBee shield yaitu uartSBee sperti pada Gambar 4.7. Agar XBee dapat bekerja dan berkomunikasi dengan Raspberry Pi maupun Arduino diperlukan konfigurasi yang akan diterangkan pada bagian implementasi.



Gambar 4.7 UartSBee dengan XBee

4.1.7 Perancangan UPNP

UPNP merupakan sebuah protokol yang memungkinkan kita mencari, mengakses, mengontrol sebuah *device* pada sebuah jaringan yang sama tanpa perlu melakukan konfigurasi jaringan terlebih dahulu. Agar fungsi tersebut dapat berjalan dibutuhkan beberapa tahapan seperti yang ditunjukkan oleh Gambar 4.8:



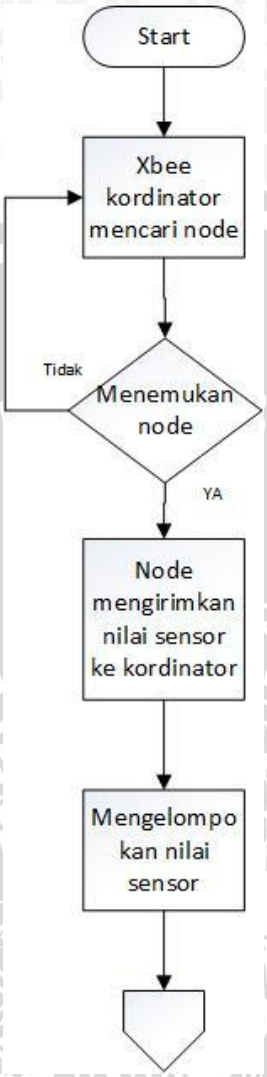
Gambar 4.8 Diagram alir proses UPNP

Agar dapat menggunakan GUPNP sebagai *framework* untuk menjalankan protokol UPNP, maka dilakukan inisialisasi yaitu dengan membuat GUPNP context terlebih dahulu. Setelah inisialisasi dilakukan, dibuatlah nama *device* sesuai dengan yang diinginkan. Lalu dilanjutkan dengan membuat layanan yang disediakan pada setiap *device*, dimana layanan tersebut juga disertai info vendor, kegunaan dan cara pengaksesannya dalam bentuk XML. Setelah semuanya selesai dan *device* tersebut terhubung ke dalam jaringan, maka dilakukan pengumuman bahwa terdapat *device* baru dan layanan baru yang terdapat pada jaringan tersebut.

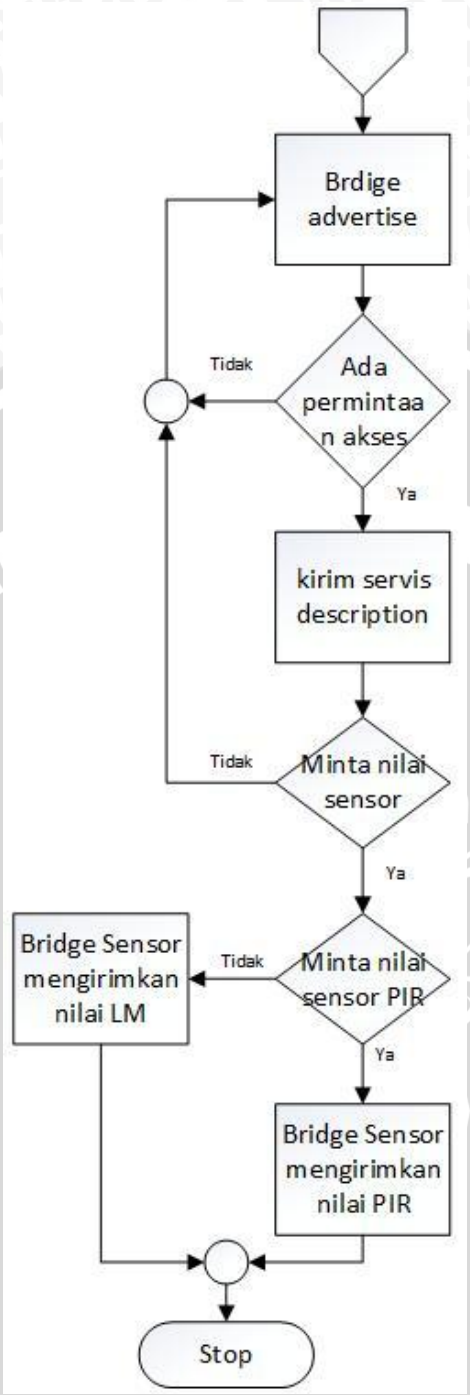
4.1.8 Perancangan Bridge Sensor

Sensor Bridge berfungsi sebagai penghubung antara peralatan yang tidak memiliki IP address ataupun tidak mendukung penerapan UPNP agar bisa tetap terhubung

kedalam jaringan UPNP sehinga dapat diakses oleh semua pihak. Agar dapat terlaksana fungsi tersebut, maka:



Gambar 4.9 Diagram alir jalan program bridge sensor



Gambar 4.10 Diagram alir jalan program bridge sensor (lanjutan)

Pada Gambar 4.9 dan 4.10 ditunjukkan proses menghubungkan sensor dengan UPNP melalui *Sensor Bridge* dimulai dengan berjalannya pencarian *node* oleh XBee koordinator pada *Sensor Bridge*, setelah *node* ditemukan masing-masing *node* tersebut mengirimkan data sensor miliknya kepada koordinator kemudian pada sisi koordinator data tersebut diteruskan ke *Sensor Bridge* untuk dipisah sesuai dengan *node* sensor pengirimnya. *Sensor Bridge* akan mulai melakukan *advertise device* yang dimilikinya, setelah itu *Sensor Bridge* akan menunggu *request*, jika ada *request* untuk ingin mengakses atau untuk mengetahui info layanan yang dimiliki

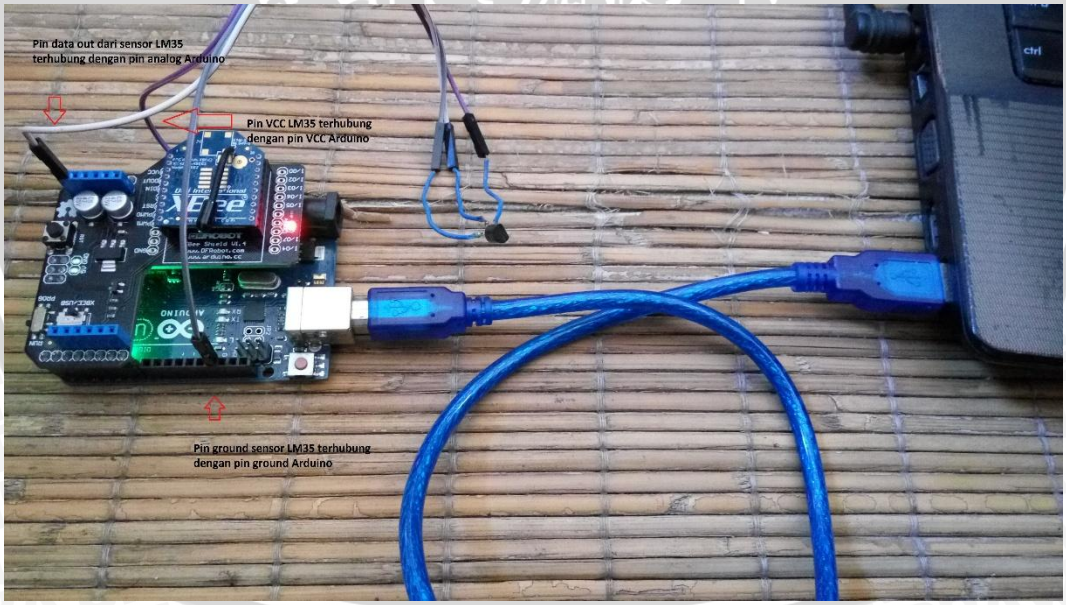
oleh *device* tersebut. Setelah itu *control point* ingin mencoba mengakses layanan yang dimiliki berdasarkan informasi yang didapat sebelumnya, *Sensor Bridge* akan membaca data nilai sensor yang dikirimkan dan dipilah tadi untuk diteruskan kepada *control point* yang meminta data tersebut.

4.2 Implementasi

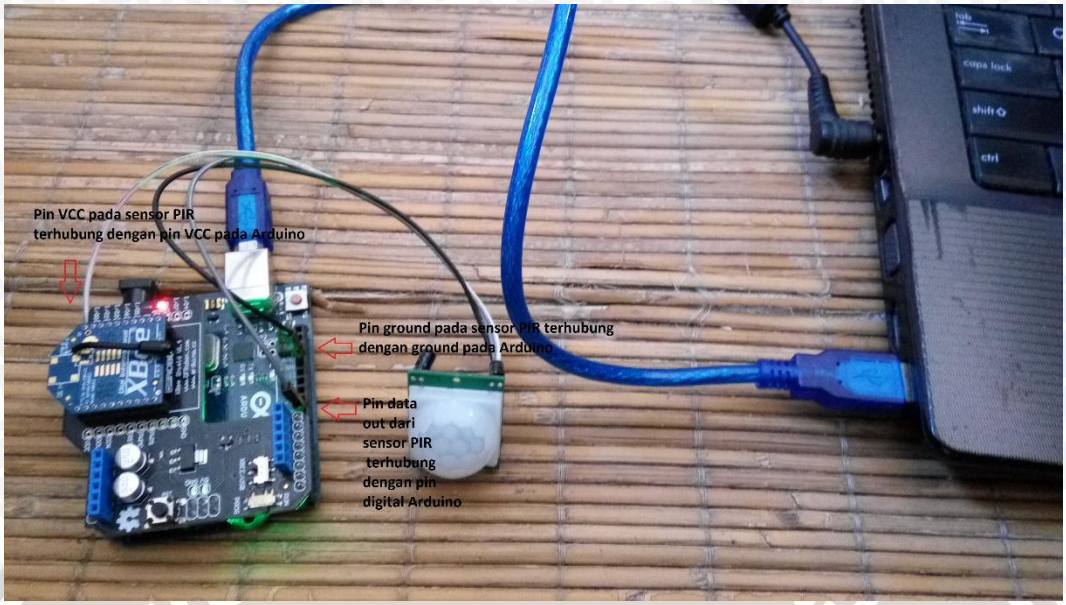
Subbab implementasi sistem akan membahas tahapan-tahapan dalam penerapan sistem berdasarkan perancangan yang telah dijelaskan pada subbab sebelumnya.

4.2.1 Implementasi Perangkat Keras

Implementasi perangkat keras akan disesuaikan dengan subbab sebelumnya, dimana digunakan sensor PIR, sensor lm35, XBee, Arduino UNO dan Raspberry Pi. Sensor PIR dan sensor LM35 masing-masing akan dihubungkan ke sebuah Arduino, sensor PIR akan terhubung dengan pin *digital* Arduino karena hasil keluaran sensor tersebut berupa data *digital* dan sensor LM35 dihubungkan pada pin *analog* hal itu disesuaikan dengan nilai keluaranya berupa data *analog*. Masing masing Arduino tersebut akan dihubungkan dengan sebuah XBee dimana Arduino dan XBee tersebut berkomunikasi secara serial menggunakan pin TX dan RX sehingga data yang ingin dikirim oleh Arduino dapat diterima dan dikirimkan melalui sinyal radio oleh XBee dan terbentuklah sebuah *node* sensor seperti pada Gambar 4.11 dan 4.12 di bawah ini:



Gambar 4.11 Implementasi node sensor LM35



Gambar 4.12 Implementasi node sensor PIR

Sedangkan pada sisi koordinator atau bagian pengumpul data yang dikirim oleh masing-masing node Arduino dan sensor tadi terdapat sebuah XBee yang terhubung dengan minikomputer Raspberry Pi, dimana Raspberry Pi dan XBee tersebut dapat berkomunikasi secara serial via USB port atau yang disebut *Sensor Bridge* seperti yang ditunjukkan oleh Gambar 4.13.

Gambar 4.13 Implementasi *Sensor Bridge*

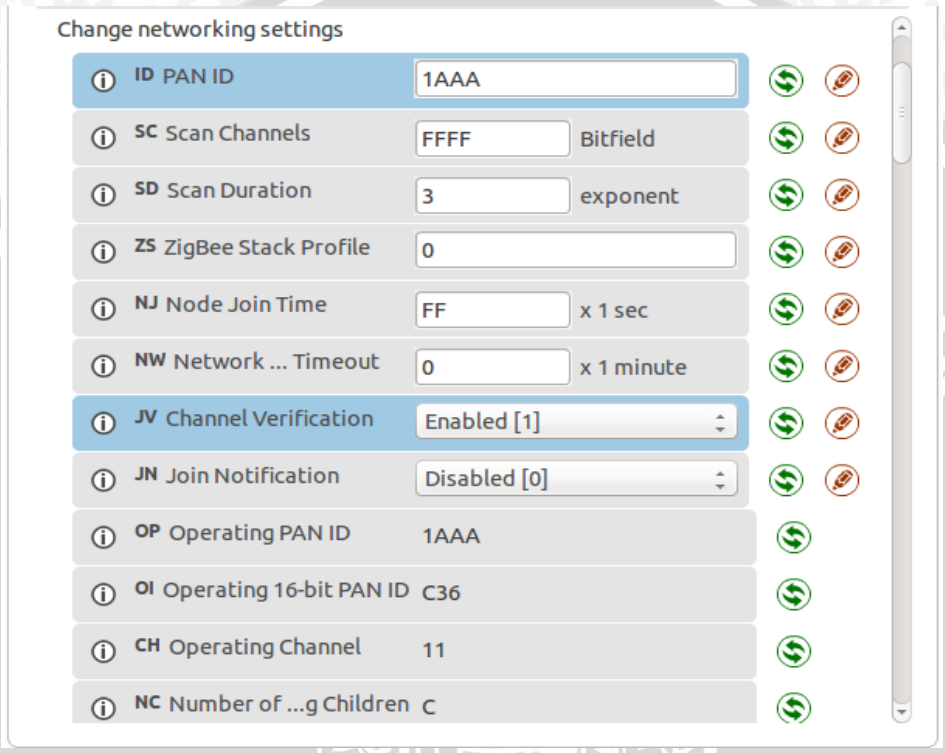
4.2.2 Implementasi Perangkat Lunak

Implementasi perangkat lunak yang diterapkan meliputi implementasi sensor, implementasi pengiriman data sensor menggunakan XBee, implementasi penerimaan data sensor pada sisi koordinator menggunakan XBee, implementasi UPNP. Program program tersebut diimplementasikan pada IDE Arduino dan

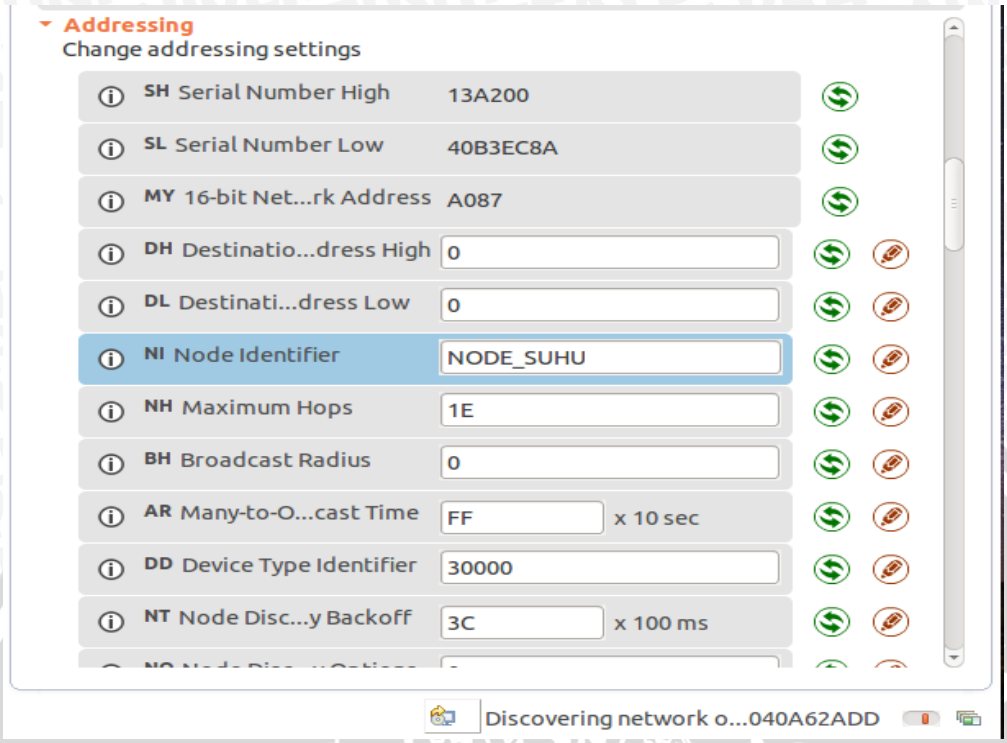
menggunakan bahasa C serta Python. Selain itu juga terdapat X-CTU yang digunakan untuk konfigurasi alamat, pengiriman mode, tes koneksi, dll pada XBee.

4.2.3 Implementasi Konfigurasi X-CTU

X-CTU merupakan software bawaan XBee yang digunakan untuk melakukan konfigurasi, pengetesan koneksi, dan pengiriman data antara komputer dengan node XBee. Untuk menggunakan mode API pada XBee maka perlu dilakukan beberapa konfigurasi seperti ditunjukkan pada Gambar 4.14 dan 4.15 merupakan konfigurasi yang diperlukan pada sisi *node sensor* atau *router*.



Gambar 4.14 Konfigurasi router pada X-CTU



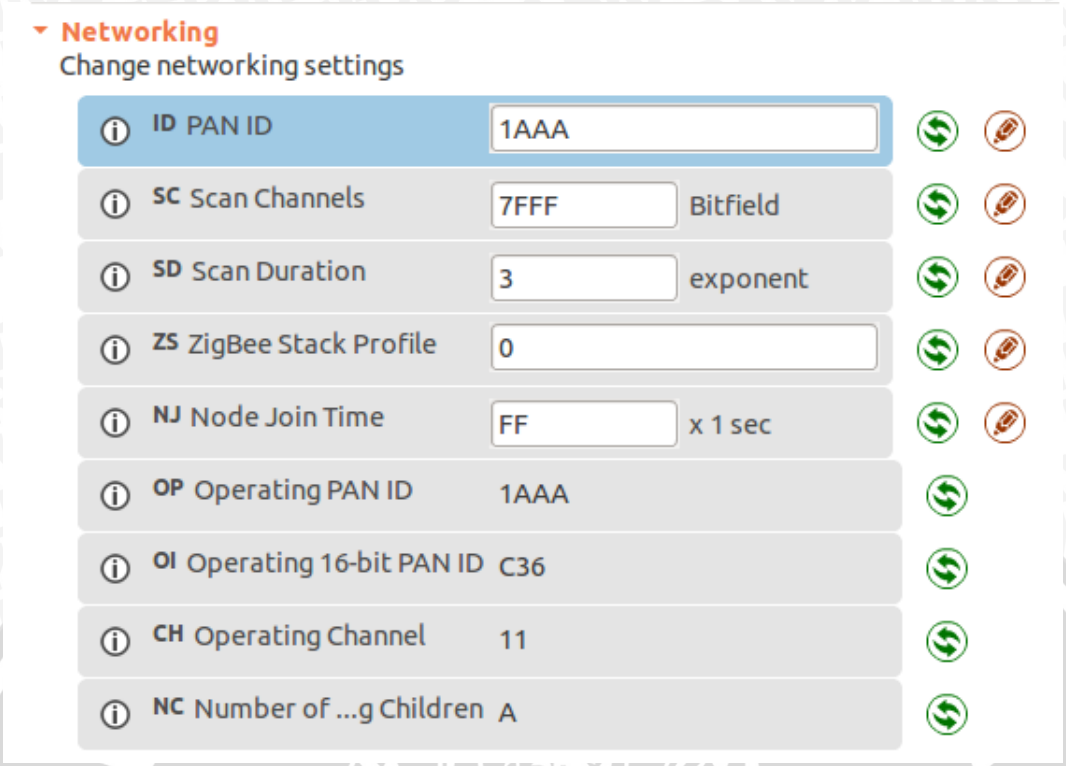
Gambar 4.15 Konfigurasi Router pada X-CTU

Agar XBee dapat saling berkomunikasi menggunakan mode API, maka perlu dilakukan konfigurasi pada aplikasi yang bernama X-CTU, pada Gambar 4.14 dan 4.15 merupakan konfigurasi yang digunakan pada XBee *router* dimana ada beberapa hal yang harus diperhatikan yaitu:

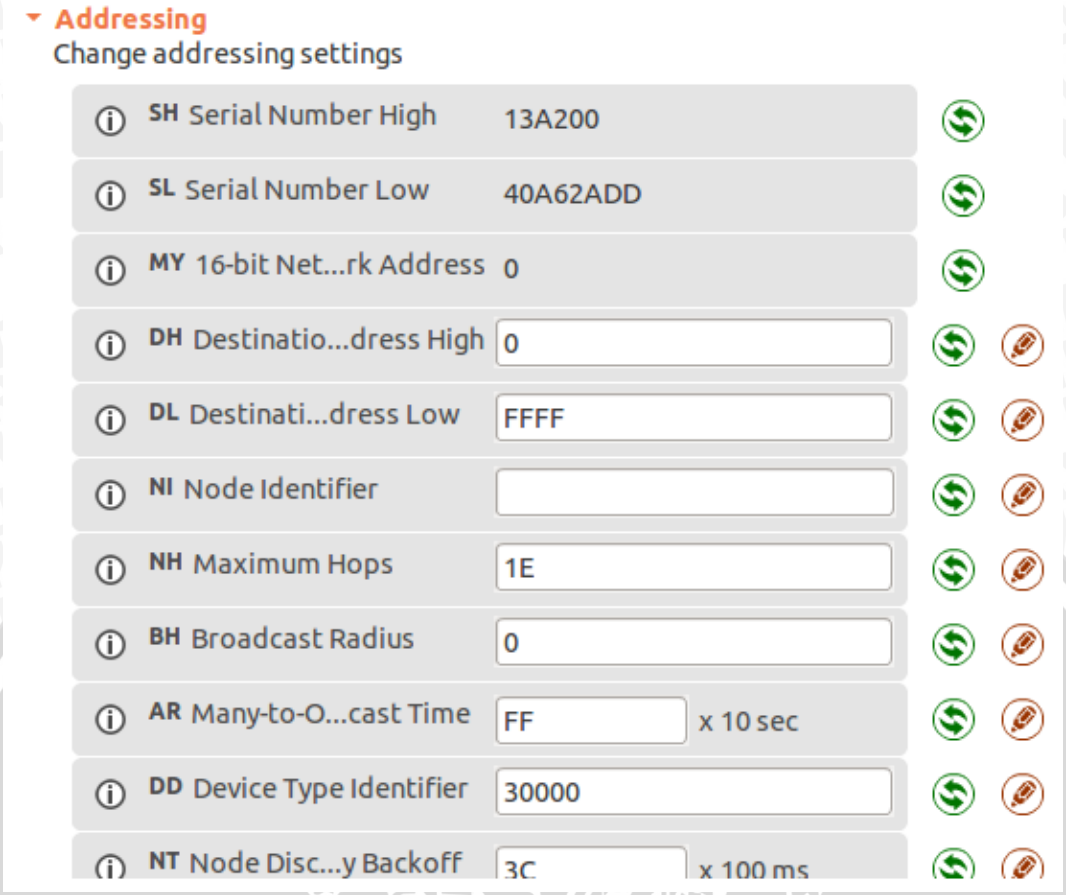
1. PAN ID
   Personal area network merupakan sebuah id unik yang digunakan dalam sebuah *network* untuk saling berkomunikasi, PAN ID di dalam satu *network* haruslah sama agar antar *node* dengan *node* ataupun *node* dengan koordinator dapat berkomunikasi.
2. DH/DL
   Destination address high dan *destination address low* merupakan sebuah kolom yang bertujuan untuk mengeset alamat *node* target yang akan dituju untuk melakukan komunikasi. Akan tetapi apabila komunikasi tersebut hanya ditujukan untuk koordinator saja maka alamat DH dan DL diset menjadi 0.
3. NI
   Node Identifier adalah sebuah nama pengenal pada *node* sehingga memudahkan kita membedakan antar *node* satu dengan yang lainya.

Untuk konfigurasi pada sisi koordinator yang berada pada Raspberry Pi, berikut merupakan langkah langkah konfigurasinya.

Sedangkan konfigurasi untuk X-CTU dapat memperhatikan Gambar 4.16 dan 4.17 berikut.



Gambar 4.16 Konfigurasi koordinator pada X-CTU



Gambar 4.17 Konfigurasi koordinator pada X-CTU

Agar koordinator mampu berkomunikasi dengan *node-node* lainya maka ada beberapa hal yang harus diperhatikan, yaitu:

1. PAN ID
   PAN ID pada koordinator haruslah sama dengan yang dimiliki oleh *node-node* yang lainya agar dapat berkomuniaksi dan terhubung di dalam jaringan yang sama.
2. DH/DL
   Untuk alamat DH dan DL pada koordinator diset menjadi DH=0 dan DL=FFFF, hal ini berarti koordinator akan mengirimkan pesan atau berkomunikasi dengan *node-node* yang lain secara *broadcast*.

4.2.4 Implementasi Sensor PIR

Sensor Passive Infrared (PIR) digunakan untuk mendeteksi keberadaan manusia pada sebuah ruangan. Untuk mendapatkan data tersebut, maka dilakukan pembacaan dan pengolahan data pada Arduino menggunakan Arduino IDE.

```
1.  . . . . . .
2.  int pirPin = 8;
3.
4.  int bacapir() {
5.    if(digitalRead(pirPin)==HIGH){
6.     digitalWrite(ledPin, HIGH);
7.     Serial.println (HIGH);
8.     data == HIGH;
9.     return 1;
10.   }
11.   else   {
12.    digitalWrite(ledPin, LOW);
13.    Serial.println(LOW);
14.    data == LOW;
15.    return 0;
16.   }
17. }
18.
19. void setup() {
20.   pinMode(pirPin, INPUT);
21. . . . . .
```

Gambar 4.18 Program pembacaan data sensor PIR pada Arduino

Dari Gambar 4.18 terdapat 2 buah fungsi yang pertama adalah fungsi bacapir() yang terletak pada baris ke 4 dan setup() pada baris ke 19. Fungsi Bacapir() berguna untuk melakukan pembacaan nilai sensor PIR dan juga menampilkan nilai yang dibacanya jika nilai yang dibaca adalah low atau tidak ada gerakan maka sensor akan menampilkanya dengan nilai 0 dan apabila ada sebuah gerakan maka akan dibaca high dan akan ditampilkan sebagai nilai 1. Untuk fungsi selanjutnya adalah fungsi setup(), fungsi tersbut merupakan sebuah inisialisasi program yang mendeklarasikan pin dan diberi nama pirPin. PirPIN merupakan representasi nama dari pin digital nomer 8 pada Arduino UNO yang digunakan sebagai masukan.

4.2.5 Implementasi Sensor LM35

Sensor LM35 digunakan untuk mendeteksi suhu ruangan disekitar sensor tersebut. Untuk melakukan pengelolahan data mentah dari sensor LM35 menjadi

nilai suhu dalam ^{o}C dilakukan pengolahan data pada Arduino menggunakan Arduino IDE seperti pada Gambar 4.16.

```
1.  . . . . . .
2.  int lmPin = 0;
3.  int data;
4.
5.  int bacalm() {
6.    data = analogRead(lmPin);
7.    data = data * (5.0*100.0/1024.0);
8.    suhuC = data;
9.    Serial.println (data);
10.   return data;
11. }
12.
13. void setup() {
14.   pinMode(lmPin, INPUT);
15. . . . . .
```

Gambar 4.19 Program pembacaan data sensor LM35 pada Arduino

Dari Gambar 4.19 terdapat 2 buah fungsi yang pertama adalah fungsi bacalm() yang terletak pada baris ke 5 dan setup() pada baris ke 13. Fungsi bacalm() untuk melakukan pembacaan nilai sensor LM35 dan juga dilakukan pengkonversian nilai sensor menjadi ^{o}C dengan menggunakan rumus.

$$Suhu = Suhu * (5*100/1024) \quad (4.1)$$

Dimana variabel suhu merupakan hasil *output* yang dikeluarkan, 5 merupakan nilai voltase pada Arduino UNO, 100 merupakan batas nilai suhu maksimal yang dibaca dan 1024 merupakan jumlah resolusi bit pin analog pada Arduino UNO sehingga kita dapat mengatur *range* antara 0v sampai 5v. Untuk fungsi selanjutnya adalah setup(), fungsi merupakan sebuah inisialisasi program yang mendeklarasikan pin yang bernama lmPin yang merupakan representasi nama dari pin analog nomer 0 digunakan sebagai input.

4.2.6 Implementasi XBee Pada Arduino

XBee digunakan untuk mengirimkan data dari satu titik ke titik yang lain melalui gelombang radio. Untuk dapat mengirimkan data XBee perlu dibekali dengan alamat penerima serta format paket data yang dikirim, hal itu akan diprogram pada Arduino menggunakan Arduino IDE.

```
1.  #include <XBee.h>
2.
3.  XBee XBee = XBee();
4.
5.  uint8_t payload[] = {0};
6.
7.  XBeeAddress64 addr64 = XBeeAddress64(0x00000000, 0x000000
8.  00);
9.  ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
10. ZBTxStatusResponse TxStatus = ZBTxStatusResponse();
11.
12. int dumpData;
13. void setup() {
14. . . . . .
15.   Serial.begin(9600);
16.   XBee.setSerial(Serial);
17. }
18.
19. void loop() {
20.   dumpData = bacalm();
21.
22.   payload[0] = dumpData;
23.
24.   XBee.send(zbTx);
25. . . . . .
    }
```

Gambar 4.20 Program pengiriman paket data pada Arduino

Pada Gambar 4.20 terdapat 2 buah fungsi yaitu setup() seperti pada baris 13 yang berisikan inisialisasi serial juga terdapat fungsi loop() dimana fungsi ini melakukan pengulangan pengiriman dan inisialisasi data yang akan dikirim. Serta terdapat juga beberapa baris program seperti pada baris 5 yang berfungsi untuk membuat *array* dan juga sebagai *payload* yang akan menampung data apa saja yang akan dikirim. Selain itu juga pada baris 7 terdapat program untuk mengeset alamat

tujuan pengiriman. Dan 8 berfungsi sebagai pengesetan urutan apa saja yang akan dikrim ke koordinator, setelah itu pada baris 9 berfungsi untuk menunggu ACK pengiriman yang berhasil.

4.2.7 Implementasi XBee pada Raspberry Pi

Setelah subbab sebelumnya menjelaskan tentang bagaimana melakukan implementasi XBee pada Arduino yang berfungsi sebagai *node*, maka kali ini akan dijelaskan implementasi XBee sebagai koordinator pada Raspberry Pi menggunakan program Python.

```
1.  def hex(bindata):
2.      return ''.join('%2x' % ord(byte) for byte in bindata)
3.
4.  def init_serial():
5.      port = '/dev/ttyUSB0'
6.      baud_rate = 9600
7.      ser = serial.Serial(port, baud_rate)
8.      return ser
9.
10. XBee = XBee(init_serial(), escaped=True)
11.
12. def explode_data():
13.             response = XBee.wait_read_frame()
14.             long_addr = hex(response['source_addr_long'][4:])
15.             rf_data = hex(response['rf_data'])
16.             shsa = hex(response['source_addr'])
17.             return rf_data, long_addr
```

Gambar 4.21 Program penerima paket data pan Raspberry Pi

Pada Gambar 4.21 terdapat 3 buah fungsi yaitu hex(), init_serial(), dan juga explode_data(). Pada baris 1 pertama dapat dilihat fungsi hex() berfungsi mengubah nilai byte menjadi hexadesimal. Pada baris ke 4 terdapat fungsi inisialisasi komunikasi serial berupa penyamaan nilai *baud rate*, dll. Pada fungsi explode_data() yang terletak pada baris 12 berguna untuk memisahkan paket data yang dikirim tadi agar dapat diambil datanya dan alamat asal pengirimnya.

4.2.8 Implementasi UPNP

Program UPNP berdasarkan perancangan yang telah dibuat ditunjukkan oleh Gambar 4.22 dibawah ini:

```
1.  void init_upnp(){
2.    GOptionContext *optionContext;
3.    GError *error = NULL;
4.    GUPNPContext *utama;
5.    GUPNPRootDevice *sensor;
6.  printf("Device created");
7.  /#if !GLIB_CHECK_VERSION(2,35,0)
8.    g_type_init ();
9.
10. utama = gupnp_context_new (NULL, NULL, 0,NULL);
11.
12. sensor = gupnp_root_device_new (utama, "XmlServer1.xml", ".");
13.
14. gupnp_root_device_set_available (sensor, TRUE);
15.
16. GUPNPServiceInfo *servicepir;
17. servicepir = GUPNP_device_info_get_service
18.   (GUPNP_DEVICE_INFO (sensor), "urn:schemas-UPNP-
19. org:service:SensorPir:1");
20.
21. GUPNPServiceInfo *servicelm;
22. servicelm = GUPNP_device_info_get_service
23.   (GUPNP_DEVICE_INFO (sensor), "urn:schemas-UPNP-
    org:service:SensorLm:1");
```

Gambar 4.22 Program UPNP Device

Pada Gambar 4.22 terdapat 1 buah fungsi yaitu init_upnp(). Fungsi ini berguna untuk pembuatan UPNP modul, pembuatan UPNP *root device*, melakukan pengumuman tentang *root device* dan layanan yang dipunyai, serta sebagai perujukan alamat deskripsi layanan yang ada.

4.2.9 Implementasi Device Description

XML *script* sebagai pemberi informasi mengenai *root device* saat dilakukan pengumuman/*advertise* kepada *control point* berdasarkan perancangan UPNP sebelumnya adalah sebagai berikut.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <root xmlns="urn:schemas-upnp-org:device-1-0">
3   <specVersion>
4    <major>1</major>
5    <minor>1</minor>
6   </specVersion>
7
8   <device>
9     <deviceType>urn:schemas-upnp-
10 org:device:XmlServer:1</deviceType>
11        <friendlyName>Home Sensor</friendlyName>
12        <manufacturer>OpenHand</manufacturer>
13        <modelName>Home Sensor</modelName>
14        <UDN>uuid:cc93d8e6-6b8b-4f60-87ca-228c36b5b0e8</UDN>
15
16        <serviceList>
17         <service>
18              <serviceType>urn:schemas-upnp-
19 org:service:SensorPir:1</serviceType>
20              <serviceId>urn:upnp-
21 org:serviceId:SensorPir:1</serviceId>
22              <SCPDURL>/SensorPir1.xml</SCPDURL>
23              <controlURL>/SensorPir/Control</controlURL>
24              <eventSubURL>/SensorPir/Event</eventSubURL>
25         </service>
26         <service>
27              <serviceType>urn:schemas-upnp-
28 org:service:SensorLm:1</serviceType>
29              <serviceId>urn:upnp-
   org:serviceId:SensorLm:1</serviceId>
```

```
30                  <SCPDURL>/SensorLm1.xml</SCPDURL>
31                  <controlURL>/SensorLm/Control</controlURL>
32                  <eventSubURL>/SensorLm/Event</eventSubURL>
33              </service>
34          </serviceList>
35      </device>
    </root>
```

Gambar 4.23 Implementasi device description

Pada Gambar 4.23 terdapat beberapa bagian yaitu specVersion, *device*, *service list*. Pada bagian specVersion berisi versi UPNP yang digunakan. Pada bagian *device* berisi tentang nama *device* yang digunakan ketika diumumkan, pembuatnya, dan nama modelnya. Berikutnya merupakan bagian servis *list* berisi layanan apa saja yang disediakan oleh *device* dan apabila ingin mengakses layanan tersebut, maka disediakan *link* untuk mengakses informasi layanan tersebut pada bagian ini.

4.2.10 Implementasi Servis

XML file untuk servis berisi tentang info-info bagaimana servis itu diakses, tipe data *variable* yang nantinya akan befungsi sebagai *input* ataupun *output* dalam UPNP itu sendiri, berikut merupakan contoh implementasi servis XML.

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <scpd xmlns="urn:schemas-upnp-org:service-1-0">
3     <specVersion>
4       <major>1</major>
5           <minor>0</minor>
6     </specVersion>
7     <actionList>
8       <action>
9         <name>GetStatus</name>
10        <argumentList>
11          <argument>
12            <name>PIR</name>
13            <relatedStateVariable>StatusPIR</relatedStateVariable>
14            <direction>out</direction>
15          </argument>
```

```
16      </argumentList>
17    </action>
18   </actionList>
19   <serviceStateTable>
20    <stateVariable sendEvents="yes">
21     <name>StatusPIR</name>
22     <dataType>int</dataType>
23     <defaultValue>0</defaultValue>
24          <allowedValueRange>
25               <minimum>0</minimum>
26               <maximum>1</maximum>
27          </allowedValueRange>
28    </stateVariable>
29   </serviceStateTable>
30  </scpd>
```

Gambar 4.24 Implementesi servis

Pada Gambar 4.24 terdapat beberapa bagian penting yaitu *spec version*, *action list* dan *state variable*. Pada bagian *spec version* berisi tentang versi upnp yang digunakan. Kemudian pada bagian *action list* merupakan servis apa saja yang dapat dilakukan dan masing masing *action* memilik namanya sendiri dan jenisnya apakah itu *output* atau *input* dan *related state variable*, dimana *related state variable* merupakan *variable* yang terhubung dari file XML dengan UPNP nantinya. Dan yang terakhir adalah *state variable* dimana pada bagian ini mendefinisikan tipe data, *range value* dari *related state variable* tadi.

4.2.11 Implementasi Bridge Sensor

Program untuk menghubungkan sensor dengan UPNP berdasarkan perancangan yang telah dibuat:

```
1.  import binascii
2.  import serial
3.  import struct
4.  import datetime
5.  from XBee import XBee
6.
7.
8.  def hex(bindata):
```

```
9.      return ''.join('%2x' % ord(byte) for byte in bindata)
10.
11.  def init_serial():
12.      port = '/dev/ttyUSB0'
13.      baud_rate = 9600
14.      ser = serial.Serial(port, baud_rate)
15.      return ser
16.
17.  XBee = XBee(init_serial(), escaped=True)
18.
19.  def explode_data():
20.      response = XBee.wait_read_frame()
21.      long_addr = hex(response['source_addr_long'][4:])
22.      rf_data = hex(response['rf_data'])
23.      shsa = hex(response['source_addr'])
24.      return rf_data, long_addr
25.
26.  def sensor_pir():
27.      data, alamat = explode_data()
28.      if alamat  == '40b7a017':
29.              data_sensor = data
30.              print datetime.datetime.now()
31.              return int(data_sensor, 16)
32.
33.  def sensor_lm():
34.      data, alamat = explode_data()
35.      if alamat == '40b3ec8a':
36.              data_suhu = data
37.              print datetime.datetime.now()
38.              return int(data_suhu, 16)
39.
40.  def main():
41.      lm = sensor_lm()
```

```
42.     if lm is None:
43.         lm = 0
44.         f = open('dataTest.Txt', 'w')
45.         f.write('%d' % lm)
46.         f.close()
47.         print lm
48.     else:
49.         f = open ('dataTest.Txt','w')
50.         f.write('%d' % lm)
51.         f.close()
52.         print lm
53.
54.     pir = sensor_pir()
55.     print pir
56.     if pir is None:
57.         pir = 0
58.         p = open ('data.Txt','w')
59.         p.write('%d' % pir)
60.         p.close
61.     else:
62.         p = open ('data.Txt','w')
63.         p.write('%d' % pir)
64.         p.close
65.
66.
67. if __name__ == '__main__':
68.         main()
69.
```

Gambar 4.25 Program Bridge Sensor

Pada Gambar 4.25 terdapat 6 buah fungsi yaitu fungsi hex(), init_serial(), explode_data(), sensor_pir(), sensor_lm() dan main. Pada baris 1 pertama dapat dilihat fungsi hex() berfungsi mengubah nilai *byte* menjadi hexadesimal. Pada baris ke 4 terdapat fungsi inisialisasi komunikasi serial berupa penyamaan nilai *baud rate*, dll. Pada fungsi explode_data() yang terletak pada baris 12 berguna untuk

memisahkan paket data yang dikirim tadi agar dapat diambil datanya dan alamat asal pengirimnya. Pada baris 25 dan 33 terdapat fungsi sensor_pir() dan juga sensor_lm() yang berfungsi untuk mengambil data sesuai dengan alamat asal pengirimnya. Dan yang terakhir adalah fungsi main, fungsi ini berguna untuk menulis data yang dipisah sesuai dengan alamat pengirimnya tadi kedalam file txt setiap ada penerimaan *update* data terbaru.

BAB 5 PENGUJIAN DAN ANALISIS

5.1 Pengujian

Pada bab ini dilakukan pengujian sistem yang telah dirancang. Pengujian yang akan dilakukan adalah pengujian fungsionalitas dimana sistem akan diuji fungsinya dan keakuratan pengiriman data serta pengujian metode yang digunakan. Dimana hasil pengujian tersebut akan digunakan sebagai rujukan dalam pengambilan kesimpulan dan hasil dari perancangan sistem ini.

5.1.1 Pengujian Sistem Keseluruhan

Tujuan dari pengujian sitem adalah untuk mengetahui apakah nilai yang dibaca oleh sensor sesuai sesuai dengan nilai yang diterima atau diakses oleh *end user*. Pengujian dilakukan dengan cara membaca nilai keluaran sensor pada Arduino dan kemudian disamakan dengan nilai yang dibaca pada *control point*. Disini terdapat dua buah sensor yang digunakan yaitu sensor LM35 dan sensor pasif infrared (PIR). Dimana masing-masing sensor tersebut merupakan *node* tersendiri. Untuk skenario pengujian sensor PIR dilakukan dengan cara sensor PIR diletakan pada dinding ruangan laboratorium, kemudian akan dilakukan pengujian dengan melakukan pergerakan berupa jalan di depan area sensor tersebut.

Untuk skenario pengujian sensor LM35 dilakukan juga pada ruangan laboratorium National Instrument dengan cara meletakan sensor di atas meja agar terpapar udara ruangan tersebut sehingga di dapatkan suhu sekitar sensor tersebut. Agar terlihat perbedaan mencolok dan dipastikan sensor itu benar-benar membaca panas suhu sekitar, maka dilakukan tes dengan mendekatkan sensor tersebut dengan korek api dan didaptkan perubahan suhu seperti pada tabel 5.2.

Hasil pengujian sensor dapat dilihat pada tabel 5.1 dan 5.2 dibawah:

1 = Ada gerakan

0 = tidak ada gerakan

Tabel 5.1 Hasil pengujian data sensor PIR

No.	Nilai Sensor PIR pada Arduino	Nilai Sensor pada Control Point
1	0	0
2	0	0
3	1	1
4	1	1
5	1	1
6	1	1

7	1	1
8	0	0
9	1	1
10	0	0
11	0	0
12	0	0
13	1	1
14	0	0
15	1	1
16	1	1
17	1	1
18	0	0
19	0	0
20	0	0

Tabel 5.2 Hasil pengujian data sensor LM35

No.	Nilai Sensor LM35 pada Arduino dalam ^{o}C	Nilai pada Control Point dalam ^{o}C
1	34	34
2	34	34
3	67	67
4	78	78
5	80	80
6	79	79
7	80	80
8	71	71
9	64	64
10	59	59
11	55	55
12	51	51
13	48	48
14	46	46

15	44	44
16	38	38
17	37	37
18	37	37
19	36	36
20	36	36

Dari hasil pengujian yang ditunjukkan pada tabel 5.1 dan 5.2 didapatkan hasil dimana nilai sensor PIR dan LM35 yang dibaca pada Arduino sama dengan nilai yang dibaca pada Raspberry Pi dengan kesamaan data 100%. Dari table 5.1 dan 5.2 dapat disimpulkan nilai yang dibaca dan diakses *end user* adalah sama.

5.1.2 Pengujian Paket Sequence

Paket *Sequence* digunakan untuk menandai nomer urutan paket data sesuai dengan urutan pengiriman sehingga di tujuan kita mengetahui data mana yang dikirim pertama dan data mana yang dikirim terakhir. Tujuan dari pengujian ini adalah dengan menggunakan paket *sequence* kita mengetahui apakah terjadi *delay* dalam pengiriman sebuah paket data dan apakah terjadi *timeout* pengiriman paket data sehingga menyebabkan paket *lost*. Cara pengujian yang dilakukan adalah dengan mengganti paket data dengan nomer nomer berurutan sesuai dengan urutan paket yang akan dikirim jadi paket pertama akan diberi nomer 1 dan paket kedua akan diberi nomer dua dan seterusnya, lalu pada penerima kita samakan hasil pengiriman dengan yang diterima apakah nomer-nomer tersebut masih berurutan atau ada yang hilang.

Berikut merupakan hasil dari pengujian paket *sequence* ditunjukkan oleh table 5.3

Tabel 5.3 Hasil pengujian paket *sequence*

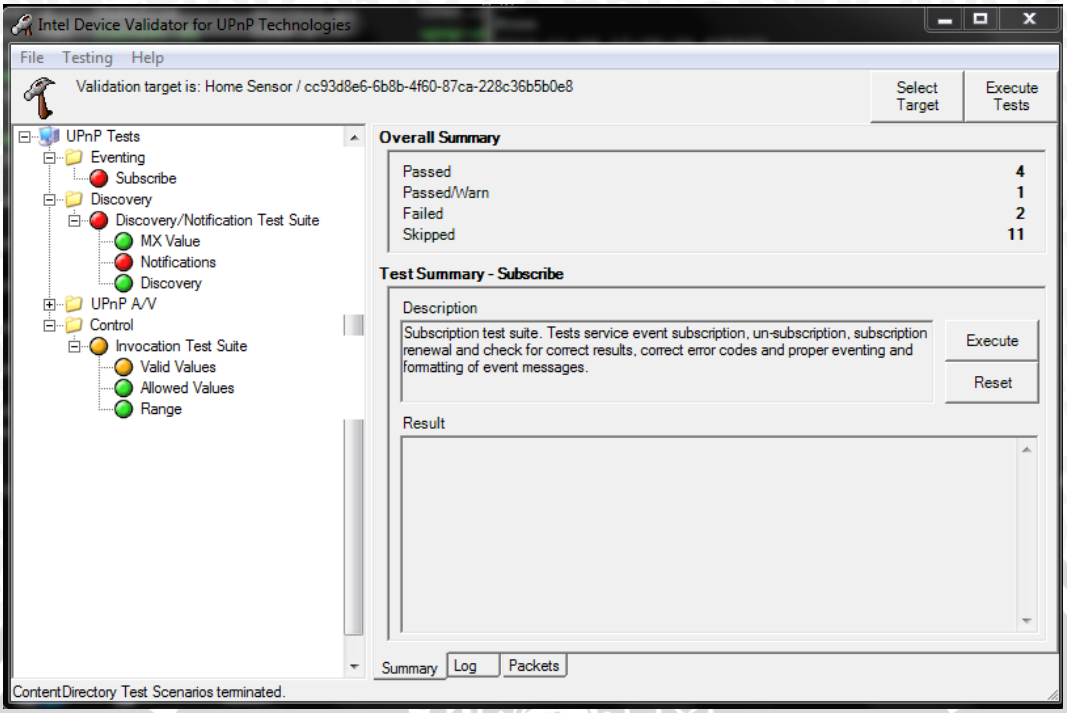
No.	Urutan Packet Squence yang Dikirim	Urutan Paket Squence yang Diterima
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8

9	9	9
10	10	10
11	11	11
12	12	12
13	13	13
14	14	14
15	15	15
16	16	16
17	17	17
18	18	18
19	19	19
20	20	20

Berdasarkan hasil pengujian pada table 5.3 didapatkan hasil urutan paket yang dikirim diterima atau tiba sesuai dengan urutan pengirimannya dan tidak ada paket yang *lost* atau tidak sampai ke tujuan. Dari hasil tersebut dapat disimpulkan bahwa data diterima sesuai dengan urutan pengirimannya dan tidak ada *timeout* ataupun paket *lost* selama proses pengirimannya.

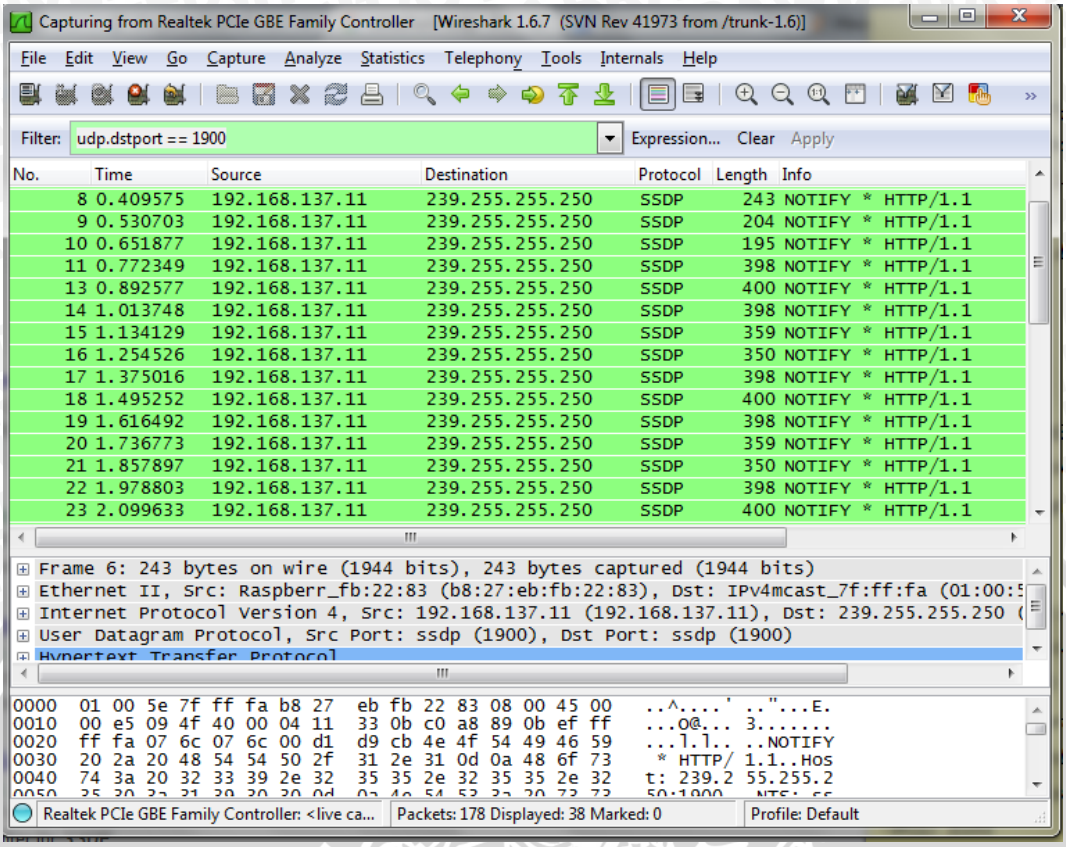
5.1.3 Pengujian Metode UPNP

Pengujian ini bertujuan untuk membuktikan apakah metode UPNP memang diterapkan dan berjalan sesuai dengan fungsinya di dalam sistem yang telah dibuat. Pengujian ini dilakukan dengan menggunakan aplikasi yaitu UPNP validator, dimana akan diuji fungsi-fungsi utama dari UPNP yaitu discovery *eventing* dan *controlling* dengan cara memilih UPNP *devic*e atau *bridge* yang akan kita uji setelah itu jalankan program dengan *execute test*.



Gambar 5.1 Hasil Pengujian UPNP

Pada Gambar 5.1 bagian *overall summary* terlihat 5 *passed* dan 1 *failed*. Dimana pada bagian *eventing* dan *control* serta sub bagiannya telah berjalan dan pada bagian *discovery* terlihat pada sub bagian *notification* terjadi kegagalan karena tidak ditemukannya atau tidak dapat berjalannya subfungsi *notification*. Akan tetapi setelah dilakukan pengecekan melalui aplikasi wireshark dengan melakukan pengamatan *traffic* pada *port ethernet* yang digunakan untuk mengakses UPNP didapatkan hasil seperti pada Gambar 5.2 dimana dengan melakukan filtering berdasarkan *port* tujuan yaitu port 1900 terlihat bahwa sebenarnya proses *notifications* telah berjalan seperti terlihat pada kolom info. Sehingga dapat disimpulkan penerapan UPNP pada sistem ini telah diimplementasikan dan berjalan sesuai dengan fungsinya.



Gambar 5.2 Filtering multicast port pada wireshark

5.2 Analisis Pengujian Sistem

Subbab ini menjelaskan tentang analisa data sesuai dengan hasil pengujian yang telah dilakukan sehingga dapat ditarik kesimpulan.

Pada pengujian sistem yang telah dilakukan didapatkan hasil yang baik, dimana sistem dapat berjalan sesuai dengan fungsinya yaitu mampu diterapkanya metode UPNP agar dapat menjangkau sensor yang tidak memiliki IP *address* dengan menggunakan *Sensor Bridge*. Sehingga data sensor tersebut tetap dapat diakses oleh para *end user* sesuai dengan nilai data terakhir yang dibaca oleh sensor tersebut.

Sehingga dapat disimpulkan bahwa sistem dapat diimplementasikan secara nyata dan dapat disempurnakan dengan menambahkan pengaksesan melalui WiFi ataupun pengaksesan *device* non-UPNP dapat diubah menjadi lebih kompleks seperti pengontrolan lampu ataupun motor sehingga memungkinkan lebih banyak pilihan servis yang diberikan.

BAB 6 PENUTUP

Berdasarkan perancangan, implementasi dan pengujian yang telah dilakukan sebelumnya maka dapat diambil kesimpulan sebagai berikut:

6.1 Kesimpulan

Kesimpulan yang dapat diambil setelah melakukan perancangan dan pengujian sistem yang telah dibuat adalah:

1. Device/sensor yang tidak memiliki IP *address* agar mampu daikses melalui jaringan yang menerapkan UPNP menggunakan *control point*. Perlu ditambahkann Sensor Bridge agar dapat diakses secara mudah dalam jaringan yang menerapkan UPNP. Sensor Bridge bekerja dengan cara menerima data dari sensor dan melakukan pengumuman/*advertising* servis sesuai dengan sensor/*device* yang terhubung dengannya, sehingga *control point* akan menerima *advertising* servis dari *device* tersebut dan tinggal mengakses servis yang diingikannya ke *Sensor Bridge* yang melakukan *advertising* tadi. Hal tersebut dibuktikan dengan pengujian sistem keseluruhan bahwa sistem mampu diakses oleh *control point* dan memberikan jawaban berupa nilai sensor yang diminta.
2. XBee sebagai media komunikasi antara *Sensor Bridge* dengan *node* sensor, memungkinkan pengaksesan data sensor tersebut secara tepat dan akurat. Hal tersebut dibuktikan dengan hasil pengujian *packet sequence* dan pengujian sistem yang mendapatkan hasil sesuai dengan apa yang dibaca dan dikirim oleh sensor.

6.2 Saran

Saran yang dapat digunakan untuk pengembangan sistem ini kedepanya adalah :

1. Untuk pengembangan sistem lebih lanjut dapat dilakukan pengaksesan *Sensor Bridge* melalui WiFi.
2. *Device* non IP yang digunakan atau yang akan diakses dikembangkan menjadi device yang dapat dikontrol seperti saklar, motor, dll.
3. Untuk sistem UPNP yang sekarang masih belum memiliki sistem keamanan, sehingga siapa saja bisa mengaksesnya. Untuk kedepanya dapat melakukan penelitian tentang bagaimana merancang sistem keamanan UPNP.
4. SDigunakannya modul RF yang lain untuk media pengiriman datanya karena untuk di Indonesia sendiri harga XBee masih terbilang masih cukup mahal.

DAFTAR PUSTAKA

Anon.,*datasheet-pdf.*[Online]
Available at: http://www.datasheet-pdf.com/PDF/HC-SR501-Datasheet-ETC-775435
[Accessed 1 Oktober 2015].

Arduino,*Arduino.*[Online]
Available at: https://www.arduino.cc/en/Main/ArduinoBoardUno
[Accessed 30 Oktober 2015].

Arduino, n.d. *Disabling Auto Reset on Serial Connection.* [Online]
Available at: http://playground.arduino.cc/Main/DisablingAutoResetOnSerialConnection
[Accessed 25 Oktober 2015].

Conductor, N. S., 2000. *Alldatasheet.* [Online]
Available at: http://pdf1.alldatasheet.com/datasheet-pdf/view/8866/NSC/LM35.html
[Accessed 1 Oktober 2015].

Digi, 2015. *Zigbee RF Modules User Guide.* s.l.:s.n.

Dononoho, A., Roe, B., Gildred, J. & Messer, A., 2015. *UPNP Device Architecture 2.0.* s.l.:UPNP Forum.

Drazen, P., Bundalo, Z., Bundalo, D. & Cvijic, B., 2015. *Zigbee Based Data Transmission and Monitoring Wireless Smart Sensor Network Integrated with the Internet.* Montenegro, MECO.

Forum UPNP, 2015. *upnp.org.* [Online]
Available at: http://upnp.org/resources/whitepapers/UPnP_SmartHome_Whitepaper_2015.pdf
[Accessed 13 Oktober 2015].

Forum, UPNP, 2008. *upnp.org.* [Online]
Available at: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf
[Accessed 13 Agustus 2015].

Hwantae, K., Suk Kyu , L., Hyunsoon, K. & Hwangnam, K., 2012. Implementing Home Energy Management System with UPNP and Mobile Application.

Instrument, T., 2015. *Texas Instrument.* [Online]
Available at: http://www.ti.com/lit/ds/symlink/lm35.pdf
[Accessed 28 Oktober 2015].

Jerome, A., 2014. *XBee and Arduino.* [Online]
Available at: http://jeromeabel.net/ressources/xbee-arduino
[Accessed 25 Oktober 2015].

Jonsson, A., 2015. *GUPNP.* [Online] Available at: https://wiki.gnome.org/Projects/GUPnP [Accessed 9 April 2015].

Jun, S.-M. & Park, N.-H., n.d. Controlling Non IP Bluetooth Device in UPNP Home Network.

Malmsten, P., Rapp, G., B. & Brackert, C., 2011. *Github.* [Online] Available at: https://github.com/nioinnovation/python-xbee [Accessed 17 Juli 2015].

Pi, R., n.d. *GPIO-PLUS-AND-RASPI2.* [Online] Available at: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/ [Accessed 25 Januari 2016].

Pi, R., n.d. *Raspberry Pi.* [Online] Available at: https://www.raspberrypi.org/help/what-is-a-raspberry-pi/ [Accessed 24 Oktober 2015].

Pi, R., n.d. *Raspberry Pi 1 Model B+.* [Online] Available at: https://www.raspberrypi.org/products/model-b-plus/ [Accessed 24 Oktober 2015].

R, B., J.R. , d. l. P. & J.M., M., 2012. Integration of Service Robots in the Smart Home by UPNP: A Surveilance Robot Case Study.

Rice, K., 2013. *RASPBERRY PI POWER SUPPLY TEST.* [Online] Available at: https://keithstechblog.wordpress.com/2013/01/20/raspberry-pi-power-supply-test/ [Accessed 24 Januri 2016].

Roman, 2013. *HC-SR501 PASSIVE INFRARED SENSOR WITH ARDUINO.* [Online] Available at: http://blog.roman-mueller.ch/index.php/2013/01/26/hc-sr501-passive-infrared-sensor-with-arduino/ [Accessed 25 Oktober 2015].

Shen-Tzong, C., CHi-Hsuan, W. & Gwo-Jiun, H., 2012. Expert System with Application. *OSGi-based smart home architecture for heterogeneous network.*

Tom, n.d. *LM35 Temprature Sensor.* [Online] Available at: http://www.instructables.com/id/LM35-Temperature-Sensor/ [Accessed 24 Oktober 2015].

Wrapp, . A., 2012. *Github.* [Online] Available at: https://github.com/andrewrapp/xbee-arduino [Accessed 23 Juni 2015].