



TUGAS AKHIR - KI091391

**IMPLEMENTASI PICTURE STREAMING PADA JARINGAN
MESH BERBASIS PROTOKOL BABEL MENGGUNAKAN
RASPBERRY PI UNTUK PEMANTAUAN JALAN RAYA**

**MUHTAROM WIDODO
NRP 5110100063**

**Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.**

**Dosen Pembimbing II
Baskoro Adi Pratomo, S.Kom, M.Kom.**

**JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014**



UNDERGRADUATE THESES - KI091391

IMPLEMENTATION OF PICTURE STREAMING USING RASPBERRY Pi BASED ON MESH NETWORK USING BABEL ROUTING PROTOCOL FOR HIGHWAY MONITORING

**MUHTAROM WIDODO
NRP 5110100063**

**Supervisor I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.**

**Supervisor II
Baskoro Adi Pratomo, S.Kom., M.Kom.**

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2014**

IMPLEMENTASI PICTURE STREAMING PADA JARINGAN MESH BERBASIS PROTOKOL BABEL MENGUNAKAN RASPBERRY PI UNTUK PEMANTAUAN JALAN RAYA

Nama Mahasiswa : MUHTAROM WIDODO
NRP : 5110100063
Jurusan : Teknik Informatika FTIF-ITS
Dosen Pembimbing 1 : Royyana Muslim Ijtihadie, S.Kom.,
M.Kom., PhD.
Dosen Pembimbing 2 : Baskoro Adi Pratomo, S.Kom., M.Kom.

Abstrak

Pada dekade terakhir ini, penggunaan *Closed-circuit Television* (CCTV) atau kamera pemantau di Indonesia semakin marak. Dengan fungsi CCTV sebagai alat untuk mengamati pergerakan manusia maka penggunaan CCTV atau kamera pemantau ini banyak digunakan sebagai alat untuk memantau keadaan sekitar untuk menghindari tindak kriminal ataupun sebagai alat untuk menelusuri pelaku tindak kriminal. Bahkan instansi pemerintahan seperti POLRI telah memanfaatkan teknologi ini untuk mengamati keadaan jalan raya untuk memantau lalu lintas maupun sebagai pencegahan ataupun penelusuran kejadian tindak kriminal.

Penerapan CCTV sendiri tidak tanpa kendala, seperti infrastruktur untuk membangun *base station* yang tergolong masih mahal, dan tergantung oleh jaringan listrik konvensional sebagai sumber energinya. Karena hal tersebut, penggunaan CCTV hanya berada pada titik tertentu yang dirasa strategis sehingga pemantauan menjadi tidak merata. Oleh sebab itu diperlukan sebuah perangkat alternatif CCTV dengan memanfaatkan komputer mini Raspberry Pi yang lebih terjangkau dan dengan memakan daya listrik yang kecil nantinya bisa dikembangkan dengan memanfaatkan teknologi panel surya.

Pengimplementasian *picture streaming* pada jaringan *wireless* mesh menggunakan perangkat Raspberry Pi merupakan jawaban atas permasalahan tersebut. Dengan mengimplementasikan perangkat ini pada jaringan *wireless* mesh akan menawarkan sebuah kemungkinan pemantau yang *plug-and-play*.

Protokol *routing* Babel digunakan sebagai sarana untuk mendistribusikan gambar karena kemampuannya yang sangat cocok diterapkan pada *embedded system*. Dengan menggunakan protokol Babel pada mini komputer Raspberry Pi penggunaan CPU hanya sekitar 1%.

Dari uji kinerja didapatkan tingkat penerimaan paket yang dikirim mencapai 93% dan kemampuan mengirim gambar sebanyak 2 *frame-per-second* sehingga sistem ini layak untuk diterapkan dalam kehidupan bermasyarakat.

Kata kunci: *Picture Streaming, Jaringan Wireless Mesh, Babel Protokol, Routing.*

IMPLEMENTATION OF PICTURE STREAMING USING RASPBERRY PI ON MESH NETWORK USING BABEL ROUTING PROTOCOL FOR HIGHWAY MONITORING

Student's Name : MUHTAROM WIDODO
Student's ID : 5110100063
Department : Teknik Informatika FTIF-ITS
First Advisor : Royyana Muslim Ijtihadie, S.Kom.,
M.Kom., PhD.
Second Advisor : Baskoro Adi Pratomo, S.Kom.,
M.Kom.

Abstract

In the last decade, the used of Closed-Circuit Television (CCTV) cameras in Indonesia has been increased. CCTV not only used for monitoring situation around to avoid crime event but also used to trace criminal actor. Even the government agencies such as the police have taken advantages of this technology to monitoring highway road.

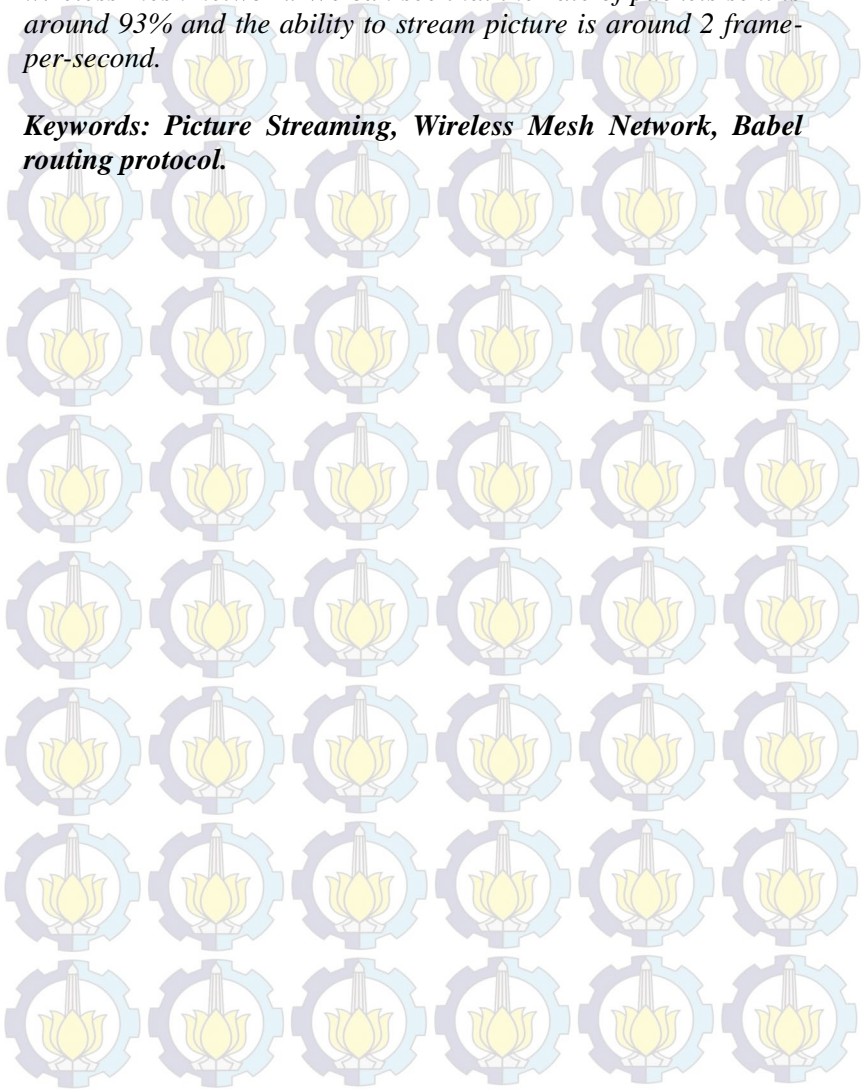
Applying CCTV is not so easy, because to build CCTV infrastructure like CCTV base station is not cheap. The need of conventional electric pole to supply its power make it difficult to install. Because of that reason, the use of CCTV only in restricted strategic area, so the monitoring become uneven.

This research try to find alternative system like CCTV but with more portable. Using Raspberry Pi as alternative tool it can be the solution. Because Raspberry Pi consume small power so it can developed by utilizing solar panel technology for their power supply. By implementing this device in wireless mesh network will offer a possibility of plug-and-play device for monitoring situation.

Because of its ability Babel routing protocol used for picture distribution in this system. With implementing Babel in embedded system in Raspberry Pi, its consume CPU usage around 1%. That reason make Babel suitable for this system.

From the performance testing of picture streaming in wireless mesh network. We can see that the rate of packets sent is around 93% and the ability to stream picture is around 2 frame-per-second.

Keywords: Picture Streaming, Wireless Mesh Network, Babel routing protocol.



LEMBAR PENGESAHAN

IMPLEMENTASI PICTURE STREAMING PADA JARINGAN MESH BERBASIS PROTOKOL BABEL MENGUNAKAN RASPBERRY PI UNTUK PEMANTAUAN JALAN RAYA

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh

MUHTAROM WIDODO
NRP : 5110 100 063

Disetujui oleh Dosen Pembimbing Tugas Akhir

1. Royyana Muslim I., S.Kom., M.Kom., PhD.

NIP: 197708242006041001

(Pembimbing 1)

2. Baskoro Adi P., S.Kom., M.Kom.

NIP: 510000003

(Pembimbing 2)

SURABAYA
JULI, 2014

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Alhamdulillahrabbi'l'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **“IMPLEMENTASI PICTURE STREAMING PADA JARINGAN MESH BERBASIS PROTOKOL MENGGUNAKAN RASPBERRY PI UNTUK PEMANTAUAN JALAN RAYA”**.

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Ayah, Ibu, dan Kakak saya yang telah memberikan dukungan moral dan material serta do'a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
2. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD. selaku pembimbing I yang telah membantu dan membimbing penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
3. Bapak Baskoro Adi Pratomo, S.Kom., M.Kom. selaku pembimbing II yang telah memberikan motivasi, nasehat, bimbingan dan bantuan yang banyak kepada penulis dalam mengerjakan Tugas Akhir ini.
4. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.

5. Jauza Irbah yang telah memberikan motivasi, bantuan, serta doa kepada penulis dalam mengerjakan Tugas Akhir.
6. Teman-teman laboratorium *Grid Computing* yang telah berbagi ilmu dan informasi dan motivasi dalam menyelesaikan Tugas Akhir ini.
7. Teman-teman HMTC yang telah membantu, berbagi ilmu dan motivasi kepada penulis.
8. Teman-teman TC angkatan 2010 yang selalu menjaga kebersamaan, serta adik-adik angkatan 2011 dan 2012 yang membuat penulis untuk selalu belajar.
9. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Juli 2014

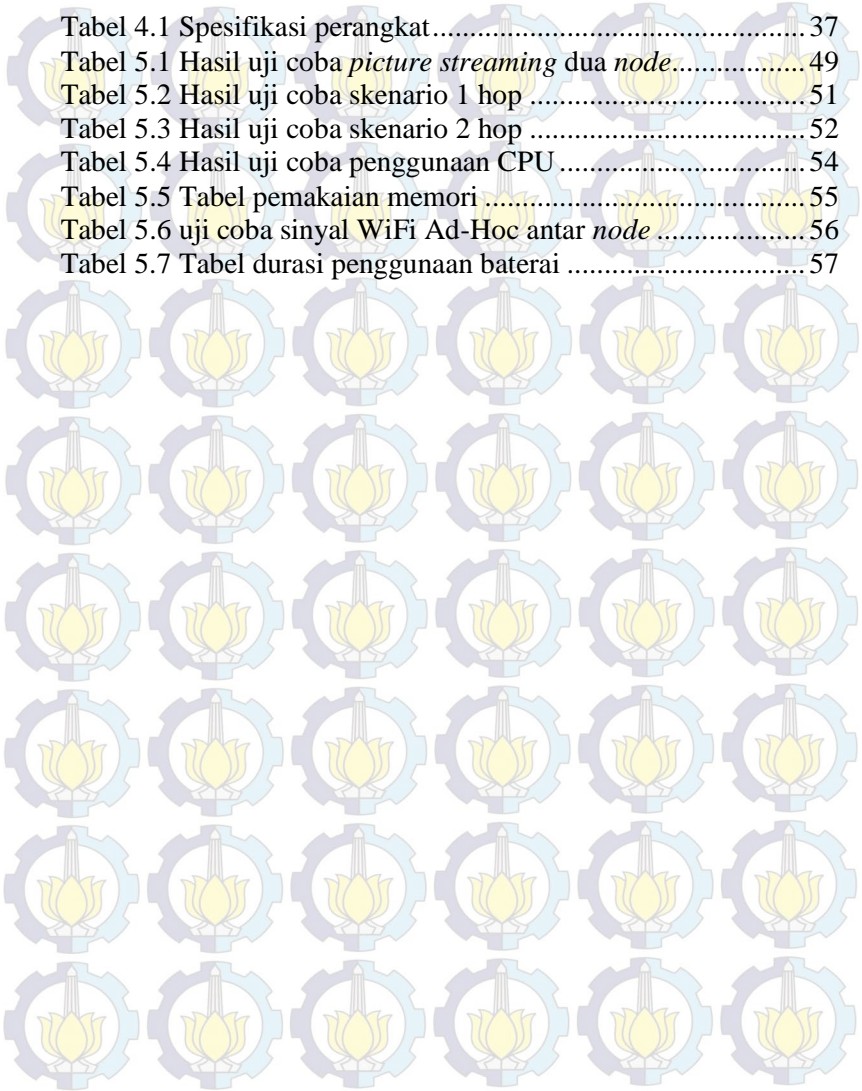
DAFTAR ISI

LEMBAR PENGESAHAN.....	v
Abstrak.....	vii
Abstract.....	ix
KATA PENGANTAR.....	xi
DAFTAR ISI.....	xiii
DAFTAR GAMBAR.....	xv
DAFTAR KODE SUMBER.....	xvii
DAFTAR TABEL.....	xix
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Batasan Masalah.....	2
1.4 Tujuan.....	3
1.5 Manfaat.....	3
1.6 Metodologi.....	3
1.7 Sistematika Penulisan Laporan Tugas Akhir.....	4
BAB II TINJAUAN PUSTAKA.....	7
2.1 Raspberry Pi.....	7
2.2 Raspbian OS.....	9
2.3 Jaringan <i>Wireless mesh</i>	10
2.4 Protokol <i>Routing</i> Babel.....	12
2.4.1 Transmisi Pesan dan Penerimaan Pesan.....	13
2.4.2 Pemilihan Rute.....	14
2.4.3 Pemeliharaan Rute.....	14
2.5 <i>Closed Circuit Television</i> (CCTV).....	15
2.6 <i>System Activity Report</i> (SAR).....	17
2.7 Python PiCamera.....	17
BAB III PERANCANGAN PERANGKAT LUNAK.....	21
3.1 Dekripsi Umum Sistem.....	21
3.2 Arsitektur Umum Sistem.....	22
3.3 Perancangan Perangkat Keras.....	23
3.4 Perancangan Diagram Alir Data level 0.....	25
3.5 Diagram Alir Aplikasi Sistem.....	25

3.5.1	Diagram Alir protokol Babel	26
3.5.2	Diagram Alir Aplikasi Sisi <i>Server</i>	27
3.5.3	Diagram Alir Aplikasi Sisi Klien	31
3.6	Rancangan Antar Muka Aplikasi	32
BAB IV IMPLEMENTASI		35
4.1	Lingkungan Implementasi	35
4.1.1	Lingkungan Implementasi Perangkat Keras	35
4.1.2	Lingkungan Implementasi Perangkat Lunak	35
4.2	Implementasi Perangkat Keras	36
4.3	Implementasi Perangkat Lunak	39
4.3.1	Implementasi pada Raspberry Pi	39
4.3.2	Implementasi pada <i>viewer</i>	43
BAB V UJI COBA dan EVALUASI		47
5.1	Uji Coba Fungsionalitas	47
5.1.1	Lingkungan Uji Coba	48
5.1.2	Uji Coba <i>Picture streaming</i>	49
5.1.3	Uji Coba <i>Picture streaming</i> pada <i>Multi-Hop Distances</i>	50
5.2	Uji Coba Performa	53
5.2.1	Penggunaan CPU dan Memori pada Mini Komputer Raspberry Pi	53
5.2.2	Uji Coba Jarak Sinyal Antar <i>node</i>	55
5.2.3	Uji Coba Daya Tahan Baterai untuk Pemantauan pada Jalan Raya	56
BAB VI PENUTUP		59
6.1	Kesimpulan	59
6.2	Saran	59
DAFTAR PUSTAKA		61
LAMPIRAN		63
1.	Memasang Sistem Operasi Raspbian Wheezy	63
2.	Inisialisasi Modul Kamera	64
3.	Mengkonfigurasi Wifi	65
4.	Instalasi PiCamera	66
5.	Implementasi protokol Babel	68
BIODATA PENULIS		75

DAFTAR TABEL

Tabel 4.1 Spesifikasi perangkat.....	37
Tabel 5.1 Hasil uji coba <i>picture streaming</i> dua <i>node</i>	49
Tabel 5.2 Hasil uji coba skenario 1 hop	51
Tabel 5.3 Hasil uji coba skenario 2 hop	52
Tabel 5.4 Hasil uji coba penggunaan CPU	54
Tabel 5.5 Tabel pemakaian memori	55
Tabel 5.6 uji coba sinyal WiFi Ad-Hoc antar <i>node</i>	56
Tabel 5.7 Tabel durasi penggunaan baterai	57



DAFTAR GAMBAR

Gambar 2.1 Raspberry Pi	8
Gambar 2.2 Logo sistem operasi Raspbian	9
Gambar 2.3 Jaringan <i>wireless mesh</i>	10
Gambar 2.4 Protokol Babel	12
Gambar 2.5 format TLV pada protokol Babel.....	13
Gambar 2.6 Kamera CCTV	15
Gambar 2.7 Digital Video Recorder	16
Gambar 2.8 Pemakain SAR.....	17
Gambar 2.9 Syntax mengambil gambar	18
Gambar 3.1 Arsitektur umum sistem.....	22
Gambar 3.2 Perangkat keras untuk <i>node</i> pemantau	24
Gambar 3.3 Perancangan diagram alir data level 0	25
Gambar 3.4 Diagram alir protokol Babel	26
Gambar 3.5 Diagram alir pada sisi <i>server</i>	27
Gambar 3.6 Diagram alir fungsi memotong gambar	28
Gambar 3.7 Diagram alir <i>thread</i> untuk mengirim pesan memberitahu alamat <i>server</i>	29
Gambar 3.8 Diagram alir <i>thread</i> untuk setiap <i>viewer</i> yang terhubung	30
Gambar 3.9 Diagram alir <i>thread</i> untuk mengirim gambar.....	31
Gambar 3.10 Diagram alir aplikasi sisi klien.....	32
Gambar 3.11 Rancangan antar muka pengguna.....	33
Gambar 4.1 Prototipe <i>node</i> pemantau	38
Gambar 4.2 Implementasi letak tiap <i>node</i> pada jalan raya.....	38
Gambar 4.3 Hasil implementasi penerimaan gambar	45
Gambar 5.1 dua <i>node</i> saling terhubung dengan protokol Babel	49
Gambar 5.2 Skenario topologi 1 hop	51
Gambar 5.3 Skenario topologi 2 hop	51
Gambar 5.4 Perbandingan persentase penerimaan paket tiap topologi.....	52
Gambar 5.5 Hasil tangkapan sistem pada jalan raya.....	57
Gambar A.1 Aplikasi Win32 Disk Imager	63
Gambar A.2 Hasil tangkapan kamera Raspberry Pi.....	64

Gambar A.3 Konfigurasi dasar ethernet	65
Gambar A.4 Konfigurasi WiFi <i>dongle</i>	65
Gambar A.5 Konfigurasi WiFi	66
Gambar A.6 Contoh penggunaan PiCamera.....	67
Gambar A.7 Perintah <i>update repository</i>	68
Gambar A.8 Perintah instalasi Babeld.....	68
Gambar A.9 Konfigurasi jaringan Ad-Hoc.....	69
Gambar A.10 Konfigurasi IPv6 modprobe.....	69
Gambar A.11 Konfigurasi ad-hoc pada terminal	70
Gambar A.12 Babeld saat dijalankan	70
Gambar A.13 Konfigurasi AHCP.....	71
Gambar A.14 Babel log saat daemon berjalan.....	72
Gambar A.15 Log saat menerima pesan hello kedua.....	73
Gambar A.16 Log saat memperbarui rute.....	73



DAFTAR KODE SUMBER

Kode Sumber 4.1 Kode Sumber implementasi <i>thread</i> menangkap dan menyimpan gambar	41
Kode Sumber 4.2 Implementasi <i>thread</i> mengirim info <i>server</i>	41
Kode Sumber 4.3 implementasi dari <i>thread</i> untuk melayani <i>viewer</i>	43
Kode Sumber 4.4 Kode sumber implementasi pada <i>server</i>	44



BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada dekade terakhir ini, penggunaan *Closed-circuit Television* (CCTV) atau kamera pemantau di Indonesia semakin marak. Dengan fungsi CCTV sebagai alat untuk mengamati pergerakan manusia maka penggunaan CCTV atau kamera pemantau ini banyak digunakan sebagai alat untuk memantau keadaan sekitar untuk menghindari tindak kriminal ataupun sebagai alat untuk menelusuri pelaku tindak kriminal. Bahkan instansi pemerintahan seperti POLRI telah memanfaatkan teknologi ini untuk mengamati keadaan jalan raya untuk memantau lalu lintas maupun sebagai pencegahan ataupun penelusuran kejadian tindak kriminal.

Penerapan CCTV sendiri tidak tanpa kendala, seperti infrastruktur untuk membangun *base station* yang tergolong masih mahal, dan tergantung oleh jaringan listrik konvensional sebagai sumber energinya. Karena hal tersebut, penggunaan CCTV hanya berada pada titik tertentu yang dirasa strategis sehingga pemantauan menjadi tidak merata. Oleh sebab itu diperlukan sebuah perangkat alternatif pengganti CCTV dengan memanfaatkan komputer mini Raspberry Pi yang lebih terjangkau dan dengan memakan daya listrik yang kecil nantinya bisa dikembangkan dengan memanfaatkan teknologi panel surya.

Pengimplementasian *picture streaming* pada jaringan *wireless mesh* menggunakan perangkat Raspberry Pi merupakan jawaban atas permasalahan tersebut. Dengan mengimplementasikan perangkat ini pada *wireless mesh* akan menawarkan sebuah kemungkinan pemantau *plug-and-play*, yang berarti penyebaran perangkat ini akan semakin mudah karena tinggal dicolokkan pada stopkontak listrik perangkat langsung berfungsi. Selain itu, *wireless mesh* juga memiliki *coverage* yang

tidak dibatasi oleh ketersediaan koneksi secara langsung ke *base station*. Tidak seperti *base station* pada umumnya, *node user* tidak perlu terhubung secara langsung namun bisa melalui *user* lain yang masuk dalam *coverage* dari *gateway* [1]. Untuk distribusi gambar sendiri menggunakan protokol Babel karena protokol Babel ini dirancang untuk mampu dan efisien diterapkan pada jaringan *wireless mesh*. Babel yang pada awalnya didesain untuk jaringan *ad-hoc* nirkabel sangat *robust* dalam mendeteksi mobilitas perangkat sehingga hanya pada kejadian khusus saja Babel akan menyebabkan *transient routing loop* [2]. Dengan adanya perangkat ini, diharapkan ke depannya pemantauan jalan raya menjadi semakin optimal karena persebaran perangkat ini.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana membangun perangkat pemantau alternatif CCTV dengan menggunakan modul kamera Raspberry Pi pada jaringan *mesh*?
2. Bagaimana membangun jaringan *mesh* pada perangkat Raspberry Pi?
3. Bagaimana mendistribusikan citra yang ditangkap oleh perangkat Raspberry Pi menuju *server* pusat menggunakan protokol Babel?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Perangkat dibangun dengan menggunakan mini komputer Raspberry Pi dan modul kamera.
2. Algoritma *routing* yang digunakan adalah menggunakan protokol Babel.
3. Tugas Akhir ini mengabaikan aspek keamanan jaringan.

4. Raspberry Pi yang terpasang modul kamera untuk menjadi *node* pemantau berjumlah satu buah.

1.4 Tujuan

Tugas Akhir ini dibuat dengan tujuan untuk membangun suatu perangkat lunak *picture streaming* pada jaringan *mesh* dengan menggunakan algoritma protokol *routing* Babel pada Raspberry Pi.

1.5 Manfaat

Dengan dibuatnya Tugas Akhir ini, diharapkan mampu memantau keadaan jalan raya dengan pemantauan yang lebih menyebar sehingga mampu untuk memantau keadaan lalu lintas maupun sebagai alat pencegahan atau penelusuran terhadap tindak kriminal.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan untuk mengimplementasikan *picture streaming* pada jaringan *mesh* berbasis protokol Babel dengan memanfaatkan mini komputer Raspberry Pi.

2. Studi literatur

Pada tahap ini dilakukan pencarian informasi dan studi literatur yang diperlukan untuk pengumpulan data dan desain sistem yang akan dibuat. Dasar informasi yang diperlukan diantaranya mengenai Raspberry Pi dan penggunaan modul kameranya secara umum, protokol *routing* Babel sebagai algoritma dalam jaringan *mesh*. Informasi dan literatur

didapatkan dari buku dan materi-materi lain yang berhubungan.

3. Perancangan sistem

Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep aplikasi yang akan dibuat. Dengan bekal teori, metode dan informasi yang sudah terkumpul pada tahap sebelumnya diharapkan dapat membantu dalam proses perancangan sistem.

4. Implementasi perangkat lunak

Implementasi merupakan tahap membangun rancangan sistem yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan perancangan yang telah dibuat sebelumnya, sehingga menjadi sebuah rancang bangun sistem yang sesuai dengan apa yang telah direncanakan.

5. Pengujian dan evaluasi

Aplikasi akan diuji setelah selesai diimplementasikan menggunakan skenario yang sudah dipersiapkan. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perancangan. Dengan melakukan pengujian dan evaluasi dimaksudkan juga untuk mengevaluasi jalannya program, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan buku Tugas Akhir.

Pada tahapan ini disusun buku yang memuat dokumentasi pelaksanaan tugas akhir, yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

BAB II. TINJAUAN PUSTAKA

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

BAB III. PERANCANGAN PERANGKAT LUNAK

Bab ini berisi tentang perancangan sistem dan *flowchart* sistem yang akan dibuat. Perancangan yang dibahas meliputi perancangan sistem, pengambilan gambar, mengirimkan gambar hasil tangkapan kamera Raspberry Pi pada *server* melalui jaringan *mesh*.

BAB IV. IMPLEMENTASI

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *pseudocode* dari fitur-fitur yang digunakan pada rancang bangun sistem ini, dan beberapa *screenshot* aplikasi.

BAB V. UJI COBA DAN EVALUASI

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian fungsionalitas dan pengujian performa dalam beberapa skenario. Pengujian fungsionalitas merupakan pengujian jalannya aplikasi sesuai dengan perancangan aplikasi pada beberapa skenario yang telah ditentukan.

BAB VI. PENUTUP

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II

TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan implementasi *picture streaming* pada jaringan *mesh* berbasis protokol babel menggunakan Raspberry Pi. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat. Tinjauan pustaka ini juga digunakan sebagai dasar yang menunjang pengembangan perangkat lunak.

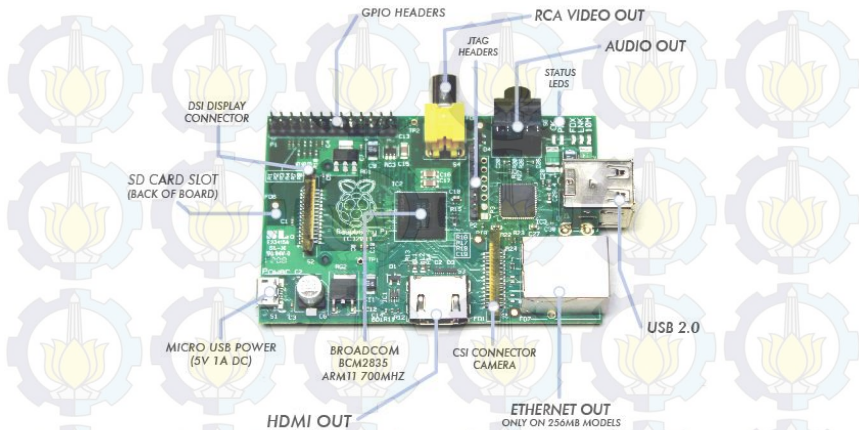
2.1 Raspberry Pi

Raspberry Pi (juga dikenal sebagai RasPi) adalah sebuah komputer mini yang mempunyai ukuran seperti kartu kredit yang dikembangkan oleh yayasan Raspberry Pi di Inggris dengan maksud untuk memicu pengajaran ilmu komputer dasar di sekolah-sekolah. Perangkat ini secara khusus diperuntukkan bagi anak-anak ataupun penggemar elektronika untuk belajar pemrograman dengan biaya yang cukup murah jika dibandingkan dengan membeli *Personal Computer* (PC) [3].

Raspberry Pi menggunakan *system on a chip* (SoC) dari Broadcom BCM2835, dan juga sudah termasuk prosesor ARM1176JZF-S 700 MHz, GPU VideoCore IV dan RAM sebesar 512 MB untuk Rev. B. Perangkat ini tidak menggunakan *hard disk*, namun menggunakan kartu SD untuk proses *booting* dan penyimpanan data.

Raspberry Pi ini mampu bekerja layaknya komputer pada umumnya dengan kemampuan untuk menjalankan sistem operasi Linux. Beberapa sistem operasi yang dapat digunakan pada Raspberry Pi:

- Raspbian OS
- Arch Linux ARM
- Raspbmc
- OpenELEC



Gambar 2.1 Raspberry Pi

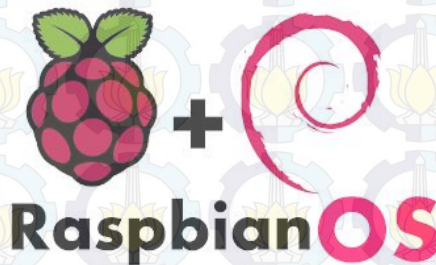
Pada Gambar 2.1 ditunjukkan beberapa bagian pada Raspberry Pi yang terdiri dari port RJ45 LAN, HDMI *output* untuk monitor, power *socket* sebagai sumber daya Raspberry Pi, slot kartu SD yang mendukung sampai ukuran 32 GB, *socket* GPIO, *socket* RCA video untuk tampilan ke TV, *socket* audio untuk luaran suara, lampu indicator LED yang terdiri dari lampu indikator *on/off* dan aktivitas jaringan LAN yang sedang terjadi dan yang terakhir adalah 2 buah *port* USB yang dapat digunakan untuk berbagai kebutuhan seperti *keyboard*, *mouse*, USB WiFi, ataupun *webcam*.

Raspberry Pi bisa digunakan untuk beragam proyek yang mudah diaplikasikan, antara lain adalah:

- Dapat dihubungkan dengan berbagai sensor seperti sensor cahaya, sensor suhu, sensor gerakan dan lainnya.
- Dapat dihubungkan dengan Arduino.
- *Network Attached Storage* (NAS), yaitu sebagai *file server* berbasis IP.
- Dapat digunakan sebagai *download manager* yang hemat listrik.

- *Server Torrent*, Raspberry Pi bisa dijadikan sebagai *server* untuk *download torrent*.
- *Home Automation*, dengan menggunakan Raspberry Pi bisa dimanfaatkan untuk mengontrol lampu, penyiraman taman, kipas angin, AC dan lain-lain melalui LCD ataupun perangkat ponsel pintar anda.

2.2 Raspbian OS



Gambar 2.2 Logo sistem operasi Raspbian

Raspbian adalah sistem operasi gratis berbasis Linux *distro* Debian yang dioptimasi untuk digunakan pada Raspberry Pi. Sistem operasi ini memiliki beberapa program standar dan beberapa kaskas bantu untuk dapat menjalankan perangkat keras dari mini komputer Raspberry Pi. Raspbian juga mampu bekerja seperti sistem operasi Debian pada umumnya. Gambar 2.2 merupakan gambar dari logo dari sistem operasi Raspbian.

Di dalam sistem operasi Raspbian berisi lebih dari 35000 paket dan *pre-compiled* perangkat lunak yang tersaji dalam bentuk format yang mudah untuk diinstalasi dalam mini komputer Raspberry Pi. Oleh karena itu Raspbian menjadi sistem operasi yang populer di kalangan pengguna mini komputer Raspberry Pi.

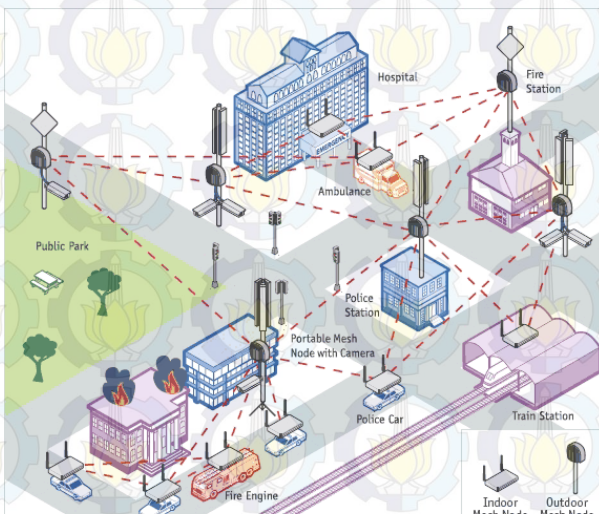
Pada Tugas Akhir ini Raspbian akan menjadi sistem operasi yang digunakan pada *node-node* untuk jaringan *wireless mesh*. Sistem Operasi yang digunakan adalah Raspbian Wheezy dengan tanggal rilis 26 Juli 2013.

2.3 Jaringan *Wireless mesh*

Jaringan *wireless mesh* merupakan salah satu jenis jaringan dimana setiap *node* di jaringan tidak hanya menerima atau mengirim data miliknya, tetapi juga berfungsi sebagai *router* untuk *node* yang lain. Dengan kata lain, setiap *node* bekerjasama untuk membangun dan mengirimkan data di jaringan [4].

Jaringan *wireless mesh* pada awalnya dikembangkan untuk aplikasi militer yang memang mempunyai arsitektur *mesh*. Sepuluh tahun belakangan, ukuran, biaya dan daya radio sudah semakin efisien, sehingga memungkinkan semakin banyak yang dapat bergabung sebagai *node mesh*.

Node mesh sendiri adalah transmitter radio yang berfungsi layaknya *wireless router*. *Node mesh* menggunakan WiFi standar yang kita tahu sebagai 802.11a, b, g dan n untuk berkomunikasi atau bertukar data dengan pengguna dan yang paling penting adalah berkomunikasi atau bertukar data dengan *node* yang lainnya. Pada Gambar 2.3 merupakan salah satu contoh jaringan *mesh*.



Gambar 2.3 Jaringan *wireless mesh*

Sebuah jaringan *wireless mesh* dapat dirancang menggunakan teknik *flooding* atau menggunakan teknik *routing*. Jika menggunakan teknik *routing* maka *message* akan dikirim melalui sebuah jalur atau rute dengan cara “loncat” dari satu *node* ke *node* yang lain sampai tujuan tercapai. Untuk menjamin keberadaan jalur atau rute maka sebuah mekanisme *routing* harus memungkinkan untuk terjadi sambungan terus menerus dan secara otomatis dapat mencari jalur rute yang tepat ketika salah satu rute atau jalur rusak (*self-healing*).

Kemampuan *self-healing* ini memungkinkan sebuah jaringan yang berbasis *routing* untuk bekerja jika salah satu *node* rusak atau mati. Akibatnya, jaringan ini umumnya sangat *reliable*, biasanya ada lebih dari satu sambungan antara sumber dan tujuan pada jaringan *wireless mesh*.

Node diprogram dengan sebuah perangkat lunak untuk memberitahu bagaimana mereka berinteraksi dengan jaringan yang lebih besar. Pesan pada *node* sumber ke *node* tujuan dikirim dengan “melompat” secara nirkabel dari satu *mesh node* ke *node* yang lainnya. *Node* secara otomatis memilih jalur tercepat dan teraman.

Keuntungan menggunakan jaringan *wireless mesh* antara lain:

- Menggunakan kabel yang lebih sedikit sehingga dapat mengurangi biaya, terutama jika diimplementasikan pada area yang luas.
- *Self-configuring* yang berarti secara otomatis menggabungkan *node* baru ke dalam struktur yang ada tanpa perlu penyesuaian oleh *administrator* jaringan.
- *Self-healing* yang berarti secara otomatis menemukan jalur tercepat dan paling *reliable* jika salah satu *node* diblokir atau kehilangan sinyal.
- Mudah untuk memasang dan melepas *node* dan mudah dikembangkan.
- Menggunakan standar WiFi yang sama yaitu 802.11a/b/g/n.

- Bisa diintegrasikan pada jaringan *indoor* maupun *outdoor*.

Untuk negara berkembang sendiri jaringan *wireless mesh* sangat berguna dimana infrastruktur jaringan kabel belum tersebar secara luas. Sebagai contoh *node* yang ditenagai oleh cahaya matahari bisa terhubung ke satu atau lebih jaringan selular sehingga bisa membuat tempat tersebut terjangkau internet.

Pada sisi pendidikan, setiap kampus bisa memulai untuk beralih menggunakan teknologi *wireless mesh*. Solusi ini mengurangi kebutuhan untuk memasang kabel bawah tanah. Dengan memasang *node* secara *indoor* maupun *outdoor* semua mahasiswa bisa terkoneksi dengan jaringan internet.

2.4 Protokol Routing Babel



The diagram shows three nodes represented by small circles. Two nodes are at the bottom, connected by a horizontal line. A third node is positioned above the left node, connected to it by a diagonal line. The word "Babel" is written in a large, bold, black font below the nodes.

Babel

Gambar 2.4 Protokol Babel

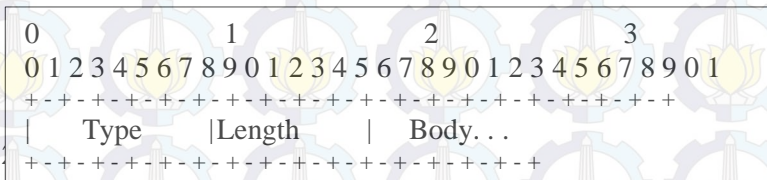
Pada gambar 2.4 ditunjukkan logo dari protokol Babel. Protokol *routing* Babel adalah sebuah protokol *routing* berbasis *distance-vector*. Babel didesain supaya mampu diimplementasikan dan efisien pada jaringan *wireless mesh* dan jaringan kabel. Babel didesain berdasarkan dari algoritma *routing* DSDV tetapi pada Babel menggunakan sesuatu pendekatan yang berbeda yaitu *Expected Transmission Count (ETX) link cost* daripada

menggunakan perhitungan *hop*. Babel memungkinkan untuk menghindari penyakit dari *routing* yaitu *routing loop* [4]

Fitur utama pada protokol Babel adalah:

- Efisien pada jaringan *wireless mesh* dan jaringan kabel.
- Mendukung untuk *overlay network*.
- Mendukung untuk IPv4 ataupun IPv6.
- Cocok untuk diimplementasikan pada *embedded system*.

Babel menempatkan informasi *routing* dengan format *type-length-value* (TLV) dan mengumpulkan beberapa TLV menjadi satu paket. Babel *node* secara opsional dapat meminta sebuah *acknowledgement* untuk setiap paket yang dikirim dengan cara menambahkan *acknowledgement request* TLV. Setiap Babel *node* secara periodik mengirimkan *broadcast Hello* TLV untuk semua tetangganya. Babel *node* secara periodik juga mengirimkan sebuah *I Heard You* (IHU) TLV untuk setiap tetangganya yang telah mengirim *Hello*. Dari informasi yang diturunkan oleh *Hello* dan IHU TLV yang diterima dari *neighbour node* sebuah Babel *node* mengalkulasikan *cost node* antara *neighbour* sampai ke *node* tersebut. Berikut format dari TLV protokol Babel pada Gambar 2.5.



Gambar 2.5 format TLV pada protokol Babel

Paket protokol Babel dikirim dalam sebuah *datagram* UDP. Setiap paket terdiri dari satu atau lebih TLV. Alamat sumber pada sebuah Babel paket selalu sebuah alamat *unicast*, *link-local* pada kasus IPv6. Babel paket dapat dikirim ke alamat *multicast* yang sudah diketahui (*link-local*) atau ke *link-local* alamat *unicast*.

Dalam keadaan normal, Babel mengirimkan kedua paket *multicast* dan *unicast* ke tetangganya. Dengan pengecualian pesan *Hello* dan *acknowledgment*, semua pesan Babel dapat dikirim ke alamat *unicast* ataupun *multicast*.

2.4.2 Pemilihan Rute

Dalam pemilihan rute, Babel dirancang untuk dapat memilih rute secara fleksibel. Dalam pemilihan rute tersebut terdapat 2 kriteria yang harus dipenuhi, yaitu:

1. Rute dengan metrik yang tidak terbatas tidak akan pernah dipilih.
2. Rute yang *unfeasible* tidak akan pernah dipilih.

Jadi bisa dikatakan bahwa, strategi sederhana untuk menentukan rute pada Babel adalah dengan memilih metrik yang paling kecil.

Sebagai catatan, Babel sebenarnya tidak menjamin stabilitas *routing* dan mengkonfigurasi kebijakan pemilihan rute yang bertentangan pada setiap router dapat menyebabkan *persistent route oscilation*.

Mendefinisikan kebijakan pemilihan rute yang baik untuk Babel sampai sekarang masih menjadi masalah penelitian yang terbuka. Pemilihan rute dapat mempertimbangkan beberapa kriteria, yaitu:

1. Rute dengan metrik kecil sebaiknya diutamakan dibanding rute dengan metrik besar.
2. Beralih router-id harus dihindari
3. Rute melalui *node* yang stabil harus lebih sering digunakan daripada melalui rute yang tidak stabil.

2.4.3 Pemeliharaan Rute

Babel *node* secara periodik memberitahukan rute yang dipilih kepada tetangganya dengan cara mengirim *update TLV*. Setiap rute pada Babel terdapat sebuah *sequence number s* dan metrik *m* untuk setiap *node n*. *Sequence number s* menyatakan

tingkat kebaruan rute tersebut diberitakan dan disebarakan ke jaringan dan *sequence number* hanya ditambah oleh n . Sebagai contoh jika sebuah *node* menerima dua pemberitaan rute untuk n dari dua tetangga yang berbeda, maka *node* tersebut akan memilih rute dengan s yang paling baru. Jika dibandingkan dengan DSDV, Babel mempercepat konvergensi ketika topologi berubah dengan secara reaktif meminta *sequence number* yang baru (dengan *sequence number request* TLV) daripada menunggu hingga *sequence number* yang baru dikirim sesuai jadwal pengiriman periodik.

2.5 *Closed Circuit Television (CCTV)*

Kamera *Closed Circuit Television (CCTV)* atau Televisi sirkuit tertutup berarti adalah televisi yang menggunakan sinyal tertutup, tidak seperti televisi pada umumnya yang menggunakan sinyal siaran. CCTV adalah suatu alat yang dapat mengirimkan data berupa video melalui transmisi kabel coaxial, FO atau UTP bahkan tanpa kabel ke lokasi tertentu untuk dimonitor, direkam, atau untuk dianalisa. Tren saat ini penggunaan CCTV sudah mengarah ke IP network camera (IP CCTV). Pada Gambar 2.6 merupakan salah satu contoh dari kamera CCTV.

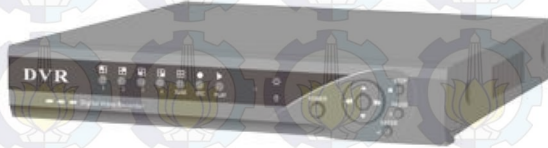


Gambar 2.6 Kamera CCTV

CCTV ini berfungsi untuk memonitor suatu ruangan melalui layar televisi atau monitor dengan menampilkan gambar dari kamera yang dipasang di setiap ruangan yang diinginkan oleh

penggunanya. Pada umumnya CCTV digunakan sebagai pelengkap keamanan dan banyak digunakan di dalam industri-industri seperti militer, bandara, toko bahkan perumahan pun telah banyak menggunakan teknologi ini.

CCTV sebagai satu kesatuan sistem memiliki dua perlengkapan utama yaitu kamera dan Digital Video Recorder (DVR). DVR adalah sistem yang digunakan CCTV untuk merekam semua gambar yang dikirim oleh kamera. Dalam sistem ini banyak fitur yang bisa digunakan salah satu yang utama adalah merekam semua kejadian dan hasil rekaman itu bisa digunakan dalam peradilan untuk membuktikan suatu kejadian dalam sebuah sistem kamera. Pada gambar 2.7 adalah contoh alat dari DVR.



Gambar 2.7 Digital Video Recorder

CCTV merupakan perangkat yang bermanfaat, salah satu manfaat CCTV adalah dapat digunakan selama 24 jam atau sesuai dengan kebutuhan pengguna, berikut ini adalah manfaat lain dari penggunaan CCTV:

- Dapat memantau dan merekam segala bentuk aktivitas yang terjadi di lokasi dari jarak jauh dari mana saja, tanpa batasan jarak melalui koneksi internet.
- Dengan menggunakan CCTV maka dapat memantau lokasi secara *realtime*.
- Rekam dan simpan hasil monitoring secara otomatis ke dalam hard disk dari rekaman selama 24 jam penuh atau hasil rekaman *motion detection*.
- Mendapatkan bukti otentik jika terjadi peristiwa yang tidak dikehendaki.

- Dapat berfungsi sebagai alarm yang akan membunyikan suara atau menelpon secara otomatis pada saat ada kejadian yang tidak dikehendaki.
- Dengan mengombinasikan fungsi *motion detect*, alarm dial, serta recording maka pengguna tidak harus terus menerus mengawasi kamera CCTV.
- Mengawasi dan memproteksi asset berharga.

2.6 System Activity Report (SAR)

System Activity Report (SAR) adalah aplikasi berbasis Solaris yang digunakan untuk memantau kinerja suatu proses dari perangkat CPU seperti contohnya untuk memantau kinerja aktivitas CPU, memantau penggunaan memory, memantau perangkat yang sedang bekerja, maupun untuk memantau kinerja dari suatu jaringan yang sedang terhubung dengan CPU tersebut. SAR ini didistribusikan melalui Linux dan melalui paket aplikasi *sysstat*.

		%user	%nice	%system	%iowait	%steal	%idle
7:02:45 PM	CPU						
7:02:47 PM	all	91.84	2.55	5.61	0.00	0.00	0.00
7:02:49 PM	all	29.74	49.74	7.69	0.00	0.00	12.82
7:02:51 PM	all	0.50	0.00	1.51	0.00	0.00	97.99
7:02:53 PM	all	5.05	0.00	15.15	0.00	0.00	79.80
7:02:55 PM	all	19.50	0.00	34.50	4.00	0.00	42.00
7:02:57 PM	all	8.59	0.00	10.10	0.00	0.00	81.31

Gambar 2.8 Pemakaian SAR

Pada Gambar 2.8 ditunjukkan hasil dari pemakaian aplikasi SAR. Terlihat total presentasi dari pemakaian CPU yang sedang terpakai juga pada gambar tersebut. Pada Tugas Akhir ini aplikasi SAR digunakan untuk melihat persentase pemakaian CPU saat aplikasi berjalan.

2.7 Python PiCamera

Python Picamera adalah paket yang menyediakan Python *interface* untuk modul kamera Raspberry Pi dengan berbasis

Python versi 2.7 atau versi di atasnya dan Python versi 3.2 atau versi di atasnya. Dengan menggunakan Python Picamera maka perintah modul kamera Raspberry Pi yang berbasis *command line* pada terminal dapat dieksekusi melalui Python *script*.

Python Picamera dapat digunakan untuk berbagai macam keperluan yang menggunakan keberadaan dari modul kamera Raspberry Pi. Salah satunya adalah untuk digunakan membuat karya seni berupa video timelapse. Dengan menggunakan Python PiCamera maka bisa didapatkan gambar dengan periode pengambilan gambar tertentu, misalnya adalah satu kali pengambilan gambar dalam rentang waktu satu jam. Selain itu Python PiCamera juga mampu untuk melakukan *streaming* video untuk ditampilkan pada video player seperti Video Lan Converter ataupun bisa untuk menangkap gambar secara terus-menerus yang nantinya bisa ditampilkan dengan membuat GUI sederhana pada Python.

Pada Gambar 2.9 adalah sebuah *syntax* sederhana untuk mengambil gambar dan menyimpannya dengan nama file tertentu.

```
import time
import picamera

with picamera.PiCamera() as camera:
    camera.resolution = (1024, 768)
    camera.start_preview()
    time.sleep(2)
    camera.capture('namaFile.jpg')
```

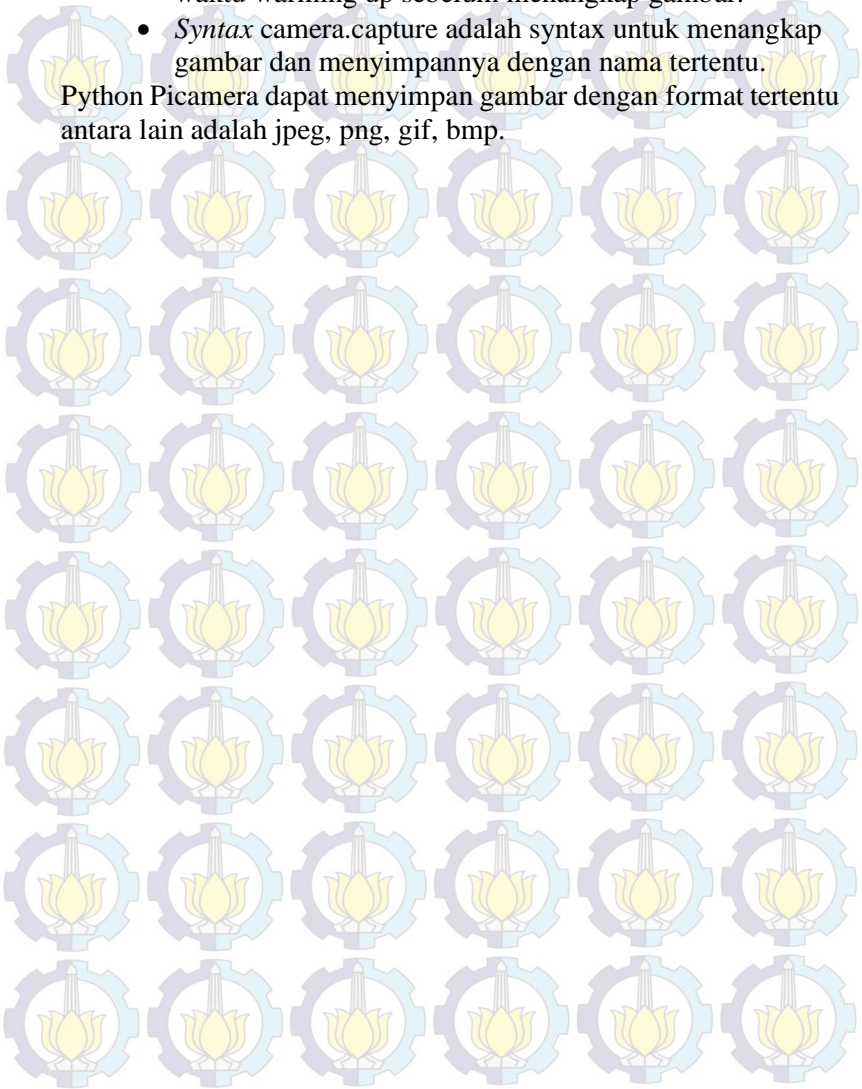
Gambar 2.9 Syntax mengambil gambar

Berikut adalah penjelasan syntax yang digunakan pada Gambar 2.9:

- *Syntax* camera.resolution adalah resolusi gambar yang akan diambil
- *Syntax* camera.start_preview adalah untuk melihat preview gambar sebelum gambar ditangkap.

- *Syntax* `time.sleep` adalah untuk memberikan kamera waktu `warming up` sebelum menangkap gambar.
- *Syntax* `camera.capture` adalah *syntax* untuk menangkap gambar dan menyimpannya dengan nama tertentu.

Python Picamera dapat menyimpan gambar dengan format tertentu antara lain adalah `jpeg`, `png`, `gif`, `bmp`.



BAB III

PERANCANGAN PERANGKAT LUNAK

Perancangan merupakan bagian penting dari pembuatan perangkat lunak yang berupa perencanaan-perencanaan secara teknis mengenai aplikasi yang akan dibuat. Bab ini secara khusus akan menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir ini. Penjelasan mencakup deskripsi umum aplikasi hingga perancangan proses, alur dan implementasinya.

3.1 Dekripsi Umum Sistem

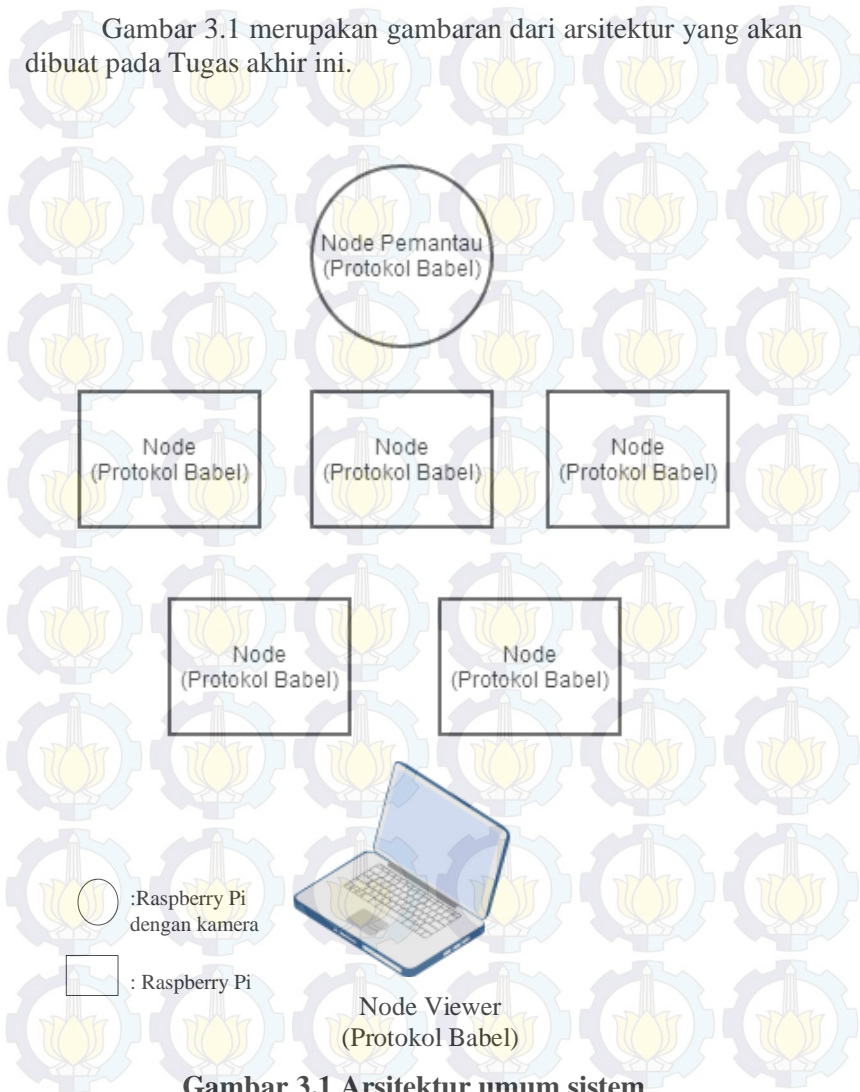
Pada Tugas Akhir ini akan dibangun suatu perangkat alternatif CCTV dengan memanfaatkan modul kamera yang ada pada Raspberry Pi. Perangkat ini akan tersambung dengan jaringan *mesh* di mana semua *node* yang ada pada jaringan ini menggunakan perangkat berupa mini komputer Raspberry Pi. Gambar yang ditangkap oleh perangkat pemantau akan dikirim menuju *viewer* melalui jaringan *mesh* dengan menggunakan metode protokol Babel.

Untuk menangkap gambar dari Raspberry Pi digunakan modul Python PiCamera yaitu Python *interface* untuk memberikan perintah pada kamera pada Raspberry Pi. Dengan menggunakan Python Picamera maka perintah-perintah modul kamera mini komputer Raspberry Pi yang berbasis *command line* pada terminal konsol dapat dieksekusi dengan menggunakan Python *script*.

Selain itu, perangkat pada Tugas Akhir ini akan disuplai oleh jaringan listrik konvensional yaitu menggunakan listrik dari power adaptor dengan output 5V 1,5A. Untuk membuat jaringan *wireless* sendiri digunakan modul USB WiFi yang kompatibel dengan perangkat mini komputer Raspberry Pi, yaitu salah satunya modul WiFi dengan chipset RTL8188CU. Untuk menangkap citra digunakan perangkat kamera yang dikeluarkan oleh element14 khusus untuk digunakan pada Raspberry Pi.

3.2 Arsitektur Umum Sistem

Gambar 3.1 merupakan gambaran dari arsitektur yang akan dibuat pada Tugas akhir ini.



Dalam sistem yang ditunjukkan Gambar 3.1 akan terdapat *node* berupa Raspberry Pi dan *node* pemantau berupa Raspberry Pi yang telah dipasang kamera. *Node* ini akan membentuk suatu jaringan *wireless mesh*. Untuk pemantauan digunakan seperangkat komputer yang berfungsi sebagai pusat pemantau. Pemantau pusat tersebut akan digunakan untuk menerima pesan gambar yang dikirim oleh *node* pemantau.

Berdasarkan pada Gambar 3.1, alur kerja perangkat dijabarkan sebagai berikut:

1. *Node* pemantau menangkap gambar dari Raspberry Pi yang telah dipasang modul kamera.
2. Setelah gambar didapatkan maka gambar akan disimpan pada perangkat *node* pemantau untuk kemudian gambar akan dikirim melalui protokol UDP ke *node viewer*.
3. Dengan menggunakan *script* bahasa pemrograman Python maka setiap gambar yang dikirim dikonversi menjadi paket berupa *string*.
4. Paket tersebut akan dikirim melalui jaringan *wireless mesh*. Tiap *node* dalam jaringan tersebut akan mencari rute sesuai dengan algoritma protokol Babel.
5. Pada *node viewer*, setelah paket diterima, maka pada *node viewer* akan mengubah isi paket yang berupa *string* menjadi gambar.
6. Gambar ditampilkan pada antarmuka aplikasi.

3.3 Perancangan Perangkat Keras

Perancangan perangkat keras pada penelitian ini menjelaskan tentang perangkat keras apa saja yang akan digunakan untuk mengimplementasikan Tugas Akhir ini. Tugas Akhir ini membutuhkan perangkat keras sebagai berikut:

1. Mini komputer Raspberry Pi model B.
2. 150Mbps *wireless* 802.11b/g/n Edimax nano USB *adapter*.
3. Modul kamera Element14 Raspberry Pi.

4. SDHC *card* Sandisk Ultra 8GB Class 10.
5. *Power adaptor* dengan *output* 5V 1,5A.
6. Kabel LAN.
7. *Keyboard* USB.
8. Monitor komputer.
9. Kabel HDMI *to* VGA.



Gambar 3.2 Perangkat keras untuk *node* pemantau.

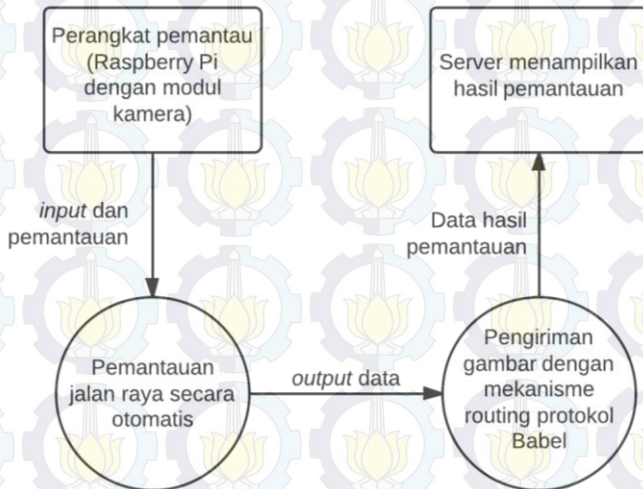
Pada Gambar 3.2 ditunjukkan perangkat keras yang dibutuhkan untuk membangun sebuah *node* pemantau. Jika untuk membangun *node* maka perangkat kamera tidak perlu dipasang pada *node* tersebut. Untuk membangun perangkat tersebut cukup mudah, tinggal memasang perangkat-perangkat tersebut pada slot yang sesuai pada Raspberry Pi.

Raspberry Pi sebagai perangkat utama untuk memasang perangkat lainnya. SDHC *card* digunakan untuk menyimpan data pada Raspberry Pi. Nano USB *adapter* nirkabel diperlukan untuk membuat jaringan *ad-hoc* untuk komunikasi data antar *node*. Pada perangkat ini penulis gunakan wifi *adapter* yang memiliki chipset RTL8188CU dikarenakan chipset tersebut kompatibel dengan Raspberry Pi. Perangkat ini akan disuplai dengan listrik dari power

adaptor dengan *output* 5V 1,5A sesuai dengan spesifikasi kebutuhan dari Raspberry Pi.

3.4 Perancangan Diagram Alir Data level 0

Pada diagram alir sistem *picture streaming* pada jaringan *mesh* dengan menggunakan protokol babel menggambarkan urutan fungsionalitas sistem secara keseluruhan. Diagram alir sistem dapat dilihat pada Gambar 3.3. Sistem diawali dengan input dari kamera yang ada pada Raspberry Pi. Kemudian input dari kamera tersebut diolah dalam bentuk gambar dengan format .jpg dengan resolusi gambar yang sudah ditentukan. Gambar tersebut akan dikirimkan kepada *server* sehingga *server* dapat menampilkan gambar secara *real-time*.



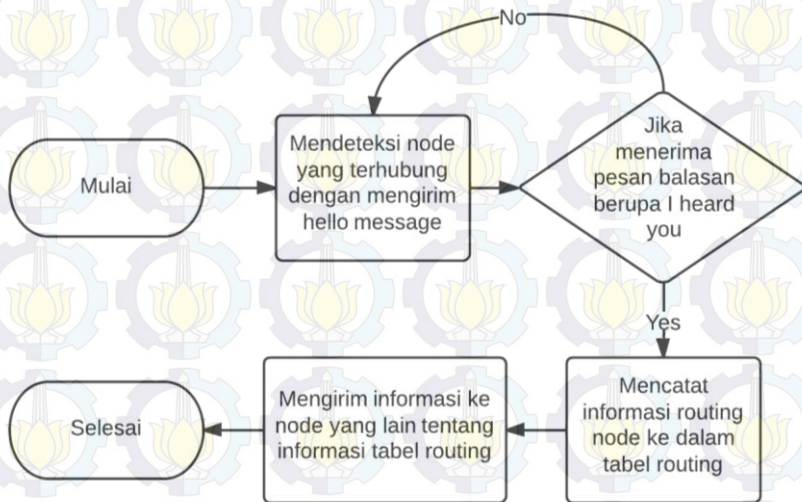
Gambar 3.3 Perancangan diagram alir data level 0

3.5 Diagram Alir Aplikasi Sistem

Alur setiap proses yang terdapat pada aplikasi digambarkan pada diagram alir untuk memudahkan pemahaman proses yang ada

pada sistem secara garis besar. Pada Tugas Akhir ini diagram alir aplikasi akan dibagi menjadi tiga bagian yaitu diagram alir protokol Babel, diagram alir pada sisi klien dan diagram alir pada server.

3.5.1 Diagram Alir protokol Babel

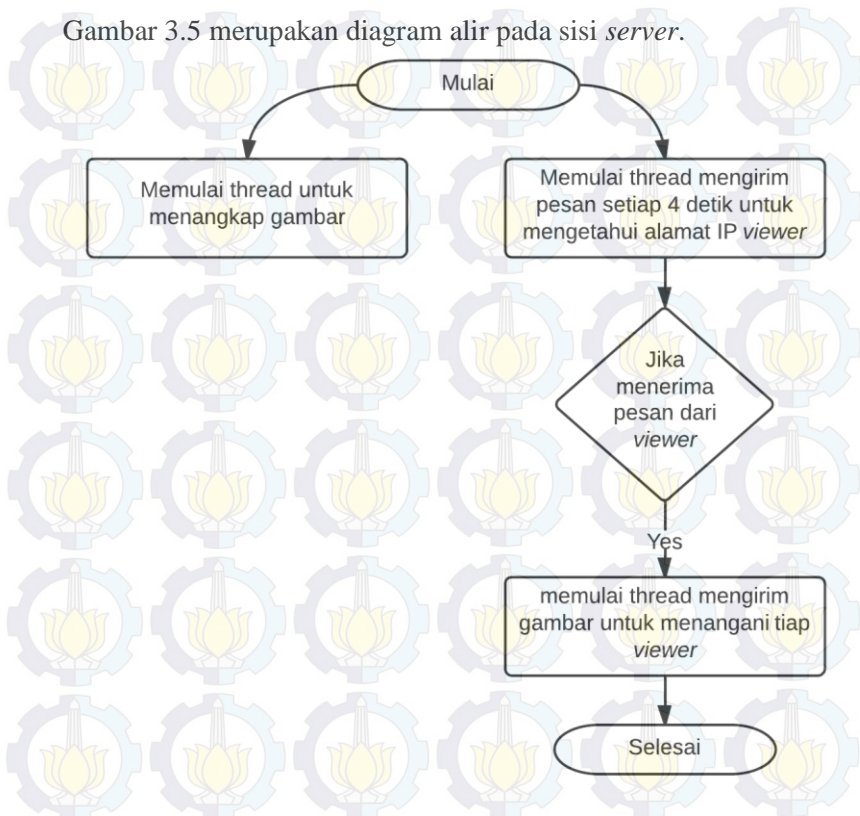


Gambar 3.4 Diagram alir protokol Babel

Untuk mekanisme routing pada sistem digunakan protokol Babel yang dibuat oleh seorang insinyur asal Prancis yang bernama Julius Chroboczek. Pada Gambar 3.4 adalah diagram alir protokol Babel. Pertama protokol Babel akan mendeteksi *node-node* tetangga dengan mengirimkan pesan *hello*. Tetangga yang mendengar pesan *hello* akan mengirimkan pesan *I Heard You* (IHU) kepada yang mengirim pesan *hello*, lalu *node* akan mencatat informasi routing tetangga dengan memasukkan ke routing table mereka. Informasi ini akan dikirimkan ke tetangga yang lain juga. Sehingga pada tiap *node* terbentuklah topologi jaringan.

3.5.2 Diagram Alir Aplikasi Sisi Server

Gambar 3.5 merupakan diagram alir pada sisi *server*.



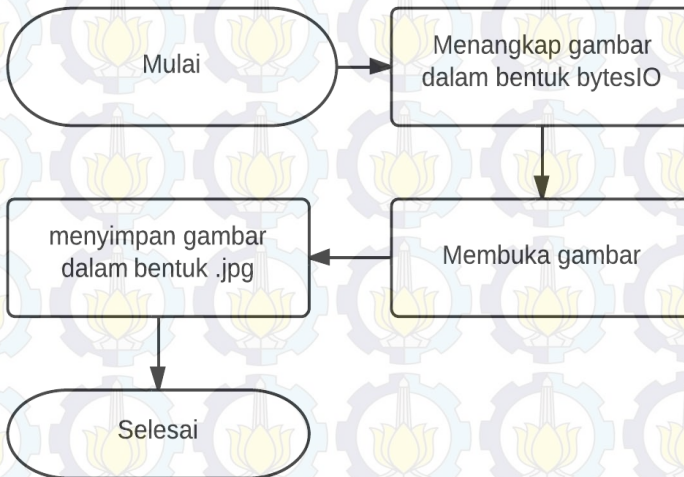
Gambar 3.5 Diagram alir pada sisi *server*

Pada sisi *server*, dalam hal ini adalah *node* pemantau. *Node* pemantau merupakan perangkat mini komputer Raspberry Pi dengan modul kamera. *Node* ini berfungsi untuk memantau keadaan jalan raya dengan cara menangkap gambar secara terus menerus. Setelah gambar ditangkap maka akan diolah untuk disimpan dalam bentuk jpg. Kemudian gambar yang sudah disimpan akan dikonversi ke dalam bentuk *string* agar dapat

dikirimkan melalui protokol UDP dalam jaringan *mesh*. Data yang akan dikirim pada klien akan berbentuk objek.

Pada sisi *server* terdapat tiga *thread*, yaitu *thread* untuk menangkap gambar, *thread* untuk mengirim pesan untuk memberitahu alamat IP *server*, *thread* untuk menerima pesan dari *viewer* dan *thread* untuk menangani setiap klien yang terhubung pada *server*. Keempat *thread* tersebut akan dijelaskan pada diagram alir berikut.

- **Thread untuk menangkap gambar**

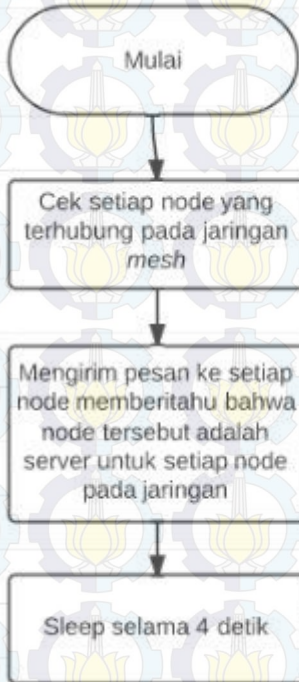


Gambar 3.6 Diagram alir fungsi memotong gambar

Pada Gambar 3.6 merupakan diagram alir dari *thread* menangkap dan mengolah gambar. *Thread* ini dibuat agar tidak ada interupsi ketika kamera menangkap gambar sehingga diharapkan agar gambar yang dikirimkan pada *viewer* akan semakin banyak.

- ***Thread* untuk mengirim pesan memberitahu alamat *server***

Pada Gambar 3.7 merupakan diagram alir dari *thread* untuk mengirim pesan memberitahu alamat *server*.

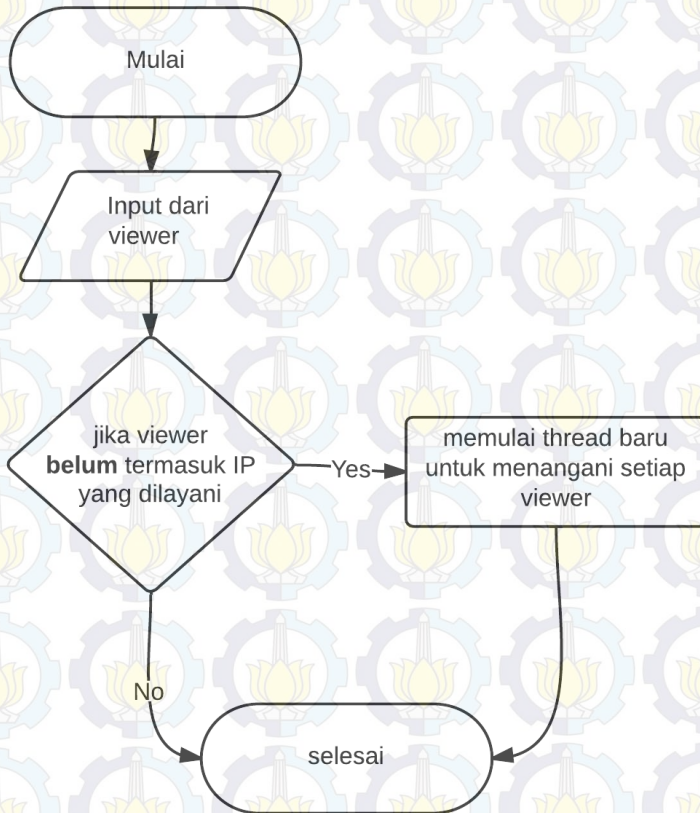


Gambar 3.7 Diagram alir *thread* untuk mengirim pesan memberitahu alamat *server*

Pada Gambar 3.7 adalah diagram alir *thread* untuk mengirim pesan memberitahu bahwa *node* tersebut adalah *server* ke *node* lainnya yang terhubung dalam jaringan *wireless mesh*. *Thread* tersebut berjalan ketika program dimulai. *Thread* ini secara periodik akan mengirim pesan bahwa *node* tersebut adalah *server* setiap 4 detik sekali. *Thread* ini dibuat terpisah dengan penerimaan

pesan agar tidak mengganggu proses pengiriman informasi ke setiap *node* dengan menunggu balasan dari *viewer* terlebih dahulu.

- **Thread untuk menangani viewer yang terhubung**

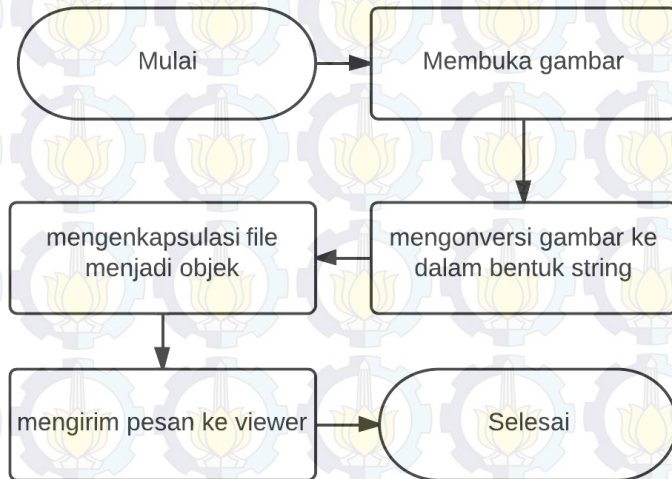


Gambar 3.8 Diagram alir *thread* untuk setiap *viewer* yang terhubung

Pada Gambar 3.8 adalah diagram alir *thread* untuk menerima pesan balasan dari *viewer* untuk info yang diberikan oleh *server* pada *thread* mengirim informasi alamat. Setelah mendapat balasan maka akan diperiksa alamat IP dari pengirim pesan

tersebut, jika alamat IP *viewer* belum dilayani maka akan dibuat *thread* baru (*thread* mengirim gambar) untuk menangani *viewer* tersebut. Diagram alir dari *thread* untuk mengirim gambar dapat dilihat pada Gambar 3.9 berikut.

- ***Thread* untuk mengirim gambar pada *viewer***



Gambar 3.9 Diagram alir *thread* untuk mengirim gambar

Pada Gambar 3.9 adalah diagram alir *thread* untuk mengirim gambar ke *viewer* yang terhubung. Untuk setiap *viewer* yang terhubung nantinya akan dilayani oleh satu *thread* untuk mengirim gambar secara *realtime*.

3.5.3 Diagram Alir Aplikasi Sisi Klien

Pada Gambar 3.10 merupakan diagram alir aplikasi pada sisi klien. Klien atau *viewer* akan menunggu pesan dari *server* yang berisi info bahwa *node* tersebut adalah *server* sehingga didapatkan alamat IP dari *node server*. Jika menerima pesan berupa informasi

tentang *server* maka akan dikirimkan balasan berupa informasi bahwa *node* ini adalah *viewer*. Kemudian jika menerima pesan berupa data gambar berupa string maka akan diproses dengan mengonversi string tersebut menjadi gambar yang kemudian akan ditampilkan pada antarmuka *viewer*.

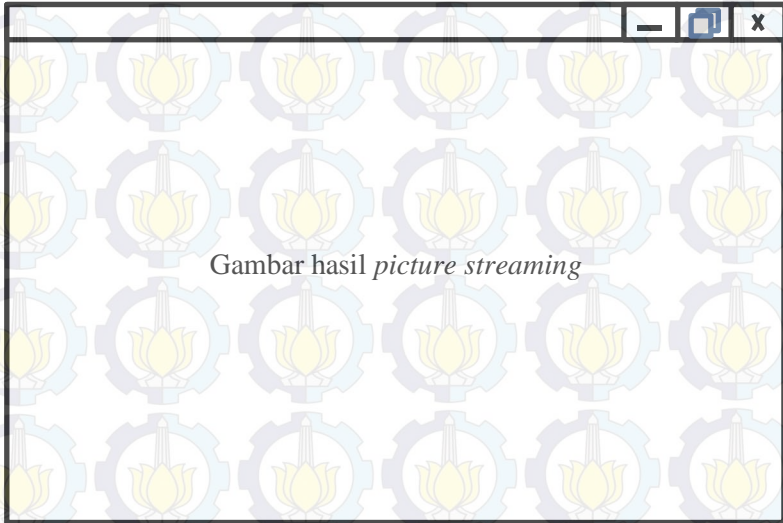


Gambar 3.10 Diagram alir aplikasi sisi klien

3.6 Rancangan Antar Muka Aplikasi

Untuk memudahkan pengguna melihat kejadian yang tertangkap melalui mekanisme *picture streaming* maka akan dibuat

sebuah antar muka sederhana untuk menampilkan gambar yang telah dikirim oleh *server*. Antar muka ini nantinya akan menggunakan modul Tkinter Python. Pada Gambar 3.11 merupakan rancangan antar muka untuk menampilkan gambar yang dikirim oleh klien dalam jaringan *wireless mesh*.



Gambar 3.11 Rancangan antar muka pengguna

BAB IV IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Dalam merancang perangkat lunak ini digunakan beberapa perangkat pendukung pengembangan sistem sebagai berikut.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem adalah komputer/*notebook* dan mini komputer Raspberry Pi. Spesifikasi dari perangkat-perangkat tersebut adalah sebagai berikut:

- Notebook HP Pavillion DV4T-1000, Intel(R) Core(TM)2 Duo CPU T9400 @2.4GHz dan 4GB memori DDR2.
- Raspberry Pi model B versi UK.

4.1.2 Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengembangan sistem adalah sebagai berikut:

- Linux Mint 14 Nadia 64 bit sebagai sistem operasi pada *notebook*.
- Raspbian Wheezy dengan tanggal rilis 26 Juli 2013 sebagai sistem operasi pada mini komputer Raspberry Pi.
- Babeld sebagai *ad-hoc routing* daemon yang digunakan untuk mengimplementasikan Babel *protocol* pada *node*.

- Bahasa pemrograman Python untuk mengimplementasikan aplikasi *picture streaming*.
- PuTTY sebagai aplikasi untuk *remote console* terhadap Raspberry Pi.
- LucidChart untuk merancang diagram alir.

4.2 Implementasi Perangkat Keras

Implementasi perangkat keras ini diawali dengan sebuah prototype untuk menguji apakah perangkat keras dapat berfungsi sebagai sistem pemantau jalan. Pada sistem ini, perangkat keras yang digunakan adalah sebagai berikut:

1. 5 buah mini komputer Raspberry Pi model B.
2. 5 buah Edimax EW-7811un 150Mbps *wireless* 802.11b/g/n nano USB *adapter*.
3. 1 buah modul kamera Element14 Raspberry Pi.
4. 5 buah SDHC *card* Sandisk Ultra 8GB Class 1
5. 5 buah *power* adaptor dengan *output* 5V 1,5A.
6. 1 buah Kabel LAN.
7. 1 buah *keyboard* USB.
8. 1 buah kabel HDMI *to* VGA.
9. 1 buah monitor.
10. 1 buah Notebook HP Pavillion DV4T-1000, Intel(R) Core(TM)2 Duo CPU T9400 @2.4GHz dan 4GB memori DDR2.

Untuk implementasi tiap-tiap perangkat keras dapat dilihat dalam subbab sebagai berikut ini.

- **Implementasi Node Pemantau dan Node Perantara**

Untuk implementasi perangkat keras berupa *node* pemantau, penulis menggunakan mini komputer Raspberry Pi Rev. B versi UK yang nantinya juga berfungsi sebagai *router*. Untuk menyuplai daya pada mini komputer Raspberry Pi digunakan sebuah power adaptor dengan output 5V 1,5A. Agar dapat memantau keadaan

jalan raya, Raspberry Pi pada *node* pemantau akan dipasang modul kamera yang dikeluarkan oleh element14 khusus untuk digunakan pada Raspberry Pi. Modul kamera ini mempunyai kemampuan untuk menangkap gambar hingga 5 megapiksel. Untuk membangun jaringan ad-hoc maka tiap perangkat akan dipasang Edimax EW-7811un 150Mbps *wireless* 802.11b/g/n nano USB *adapter*.

Untuk memasang modul kamera pada Raspberry Pi, maka yang harus diperhatikan adalah bahwa kamera ini merupakan perangkat yang sensitif, dimana jika perangkat terkena aliran listrik statis maka perangkat akan rusak. Modul kamera sendiri dipasang pada konektor CSI pada Raspberry Pi, yang terletak pada bagian belakang dari *socket* Ethernet LAN.

Sedangkan untuk implementasi perangkat keras berupa *node* perantara, maka bagian yang dihilangkan adalah perangkat modul kamera saja. Pada Tabel 4.1 adalah spesifikasi perangkat *node* pemantau dan *node* perantara.

Tabel 4.1 Spesifikasi perangkat

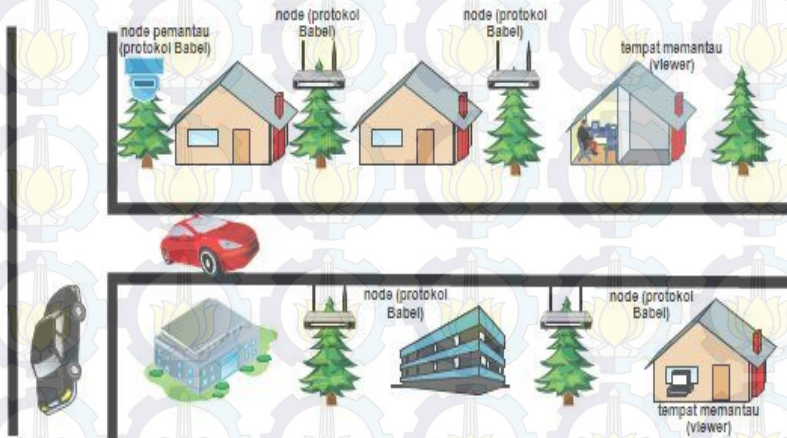
	<i>Node</i> Pemantau	<i>Node</i> Perantara
Perangkat	Raspberry Pi Rev. B versi UK	Raspberry Pi Rev. B versi UK
WiFi adapter	Edimax EW-7811un	Edimax EW-7811un
Modul Kamera	element14 kamera	tidak ada
Power Supply	Power adaptor 5V 1,5A	Power adaptor 5V 1,5A

Pada Gambar 4.1 merupakan prototipe dari alat untuk memantau keadaan jalan raya.



Gambar 4.1 Prototipe *node* pemantau

Pada Gambar 4.2 merupakan implementasi letak tiap *node* yang akan direncanakan. *Node-node* ini akan diletakkan dekat dengan sumber listrik. *Node* pemantau diletakkan pada tempat-tempat yang strategis untuk melakukan pemantauan.



Gambar 4.2 Implementasi letak tiap *node* pada jalan raya

4.3 Implementasi Perangkat Lunak

Pada subbab ini akan lebih banyak dibahas bagaimana implementasi perangkat lunak pada *notebook* HP dan mini komputer Raspberry Pi agar semua perangkat dapat bekerja dengan baik dan sesuai dengan rancangan sistem yang dibuat agar sistem ini dapat berjalan dengan sempurna.

4.3.1 Implementasi pada Raspberry Pi

Pada subbab ini akan dijelaskan mengenai perangkat lunak yang dipasang pada Raspberry Pi agar perangkat dapat bekerja sebagai mana mestinya sesuai sistem yang dibuat. Untuk mengimplementasikan perangkat lunak pada Raspberry Pi terdapat beberapa hal penting yang harus dilakukan, antara lain memasang sistem operasi Raspbian Wheezy, menginisialisasi perangkat kamera pada mini komputer Raspberry Pi, mengkonfigurasi WiFi, memasang modul Python-PiCamera dan memasang Babel daemon dan mengkonfigurasinya. Hal-hal tersebut dapat diketahui lebih lanjut pada lampiran.

1. Implementasi *picture streaming*

Proses mengirim gambar yang dilakukan pada *node* pemantau merupakan proses utama untuk implementasi Tugas Akhir ini disamping proses untuk melakukan *routing* oleh protokol Babel. Proses *picture streaming* diawali oleh modul kamera pada mini komputer Raspberry Pi yang bertugas untuk menangkap gambar yang kemudian disimpan dan kemudian akan dikirim melalui UDP *socket*. Pada subbab berikut akan dijelaskan lebih lanjut mengenai proses-proses yang terlibat.

a. Implementasi pada *server*

Pada sisi *server* terdapat tiga *thread*, yaitu *thread* untuk menangkap dan memotong gambar, *thread* untuk mengirim pesan

untuk memberitahu posisi *server* dan *thread* untuk menangani setiap klien yang terhubung pada *server*. Ketiga *thread* tersebut akan berjalan sebagai fungsi-fungsi penyusun aplikasi pada sisi *server*. Implementasi ketiga *thread* tersebut akan dijelaskan pada kode sumber berikut.

- **Implementasi *thread* menangkap dan memotong gambar.**

Thread ini dibuat agar tidak ada proses yang menginterupsi proses menangkap gambar. Dengan dibuatnya *thread* ini diharapkan gambar yang dikirimkan pada *node* klien menjadi semakin banyak.

1	<code>import picamera</code>
2	<code>import time</code>
3	<code>import io</code>
4	<code>import PIL.Image</code>
5	<code>import os</code>
6	<code>camera = picamera.PiCamera()</code>
7	<code>camera.resolution = (640, 480)</code>
8	<code>camera.vflip = True</code>
9	<code>camera.hflip = True</code>
10	<code>camera.start_preview()</code>
11	<code>time.sleep(2)</code>
12	<code>imgCount = 1</code>
13	<code>stream = io.BytesIO()</code>
14	<code>for foo in</code> <code>camera.capture_continuous(stream,</code> <code>'jpeg', use video port=True)</code>
15	<code>path = "file "+str(imgCount)+".jpg"</code>
16	<code>stream.seek(0)</code>
17	<code>image = PIL.Image.open(stream)</code>
18	<code>image.save(path)</code>
19	<code>stream.seek(0)</code>
20	<code>stream.truncate()</code>

21	<code>imgCount +=1</code>
22	<code>if imgCount > 10:</code>
23	<code>imgCount = 1</code>

Kode Sumber 4.1 Kode Sumber implementasi *thread* menangkap dan menyimpan gambar

- **Implementasi *thread* mengirim pesan memberitahu alamat *server*.**

Thread ini digunakan untuk mengirim pesan memberitahu bahwa *node* tersebut adalah *server* ke *node* lainnya yang terhubung dalam jaringan *wireless mesh*. *Thread* tersebut berjalan ketika program dimulai. *Thread* tersebut secara periodik akan mengirim pesan bahwa *node* tersebut adalah *server* setiap 4 detik sekali. Pada Kode Sumber 4.2 merupakan implementasi *thread* ini.

1	<code>from netils import Route</code>
2	<code>def sendInfo (runEvent):</code>
3	<code>message = "I am server"</code>
4	<code>message= pickle.dumps(message)</code>
5	<code>while runEvent.is set()</code>
6	<code>rute = Route.read route table()</code>
7	<code>for ipAddr in rute:</code>
8	<code>socket.sendto(message, str(ipAddr),</code> <code>8888)</code>
9	<code>print "send to", ipAddr</code>
10	<code>time.sleep(4)</code>

Kode Sumber 4.2 Implementasi *thread* mengirim info *server*

- **Implementasi *thread* untuk menerima balasan dan melayani *viewer***

Thread ini akan mengakses file gambar yang telah dipotong-potong menjadi beberapa bagian untuk kemudian dikirim pada *viewer*. *Thread* ini dimulai ketika fungsi utama / fungsi main mendapat pesan balasan dari *viewer*, kemudian akan diperiksa

apakah IP *viewer* tersebut sudah ada dalam list yang dilayani, jika belum maka akan dibuat *thread* baru untuk melayani IP *viewer* tersebut.

1	import socket
2	import base64
3	import threading
4	def sender (runEvent, ipAddr):
5	Message = []
6	imgName = 0
7	imgCount = 1
8	while runEvent.is set():
9	path =
	str(imgName)+" "+str(imgCount)+".jpg"
10	imageFile = open(path, "rb")
11	strImage =
	base64.b64encode(imageFile.read())
12	message = imgCount, strImage
13	mPickle = pickle.dumps(message)
14	sock.sendto(mPickle, (ipAddr, 8888))
15	imgCount +=1
16	if imgCount > 3:
17	imgName +=1
18	imgCount = 1
19	if imgName > 3:
20	imgName = 0
21	imgCount = 1
22	listClient = []
23	poolThread = []
24	def receiveMsg(runEvent)
25	while runEvent.is set():
26	pesan, addr = sock.recvfrom(65504)
27	print "received message from", addr
28	if pesan == I am v':
29	if ip not in listClient:

30	<code>listClient.append(ip)</code>
31	<code>thread = threading.Thread(target=sender, args=(runEvent1, ip))</code>
32	<code>poolThread.append(thread)</code>

Kode Sumber 4.3 implementasi dari *thread* untuk melayani *viewer*

4.3.2 Implementasi pada *viewer*

Untuk perangkat lunak yang dipasang pada *server* yang digunakan untuk memantau adalah antara lain:

1. Linux Mint Nadia sebagai sistem operasi pada *server*.
2. Babel daemon sebagai daemon *routing* protokol Babel.
3. Python-dev sebagai bahasa pemrograman untuk menampilkan hasil dari *picture streaming*.
4. Pillow dan libjpeg-dev sebagai modul pengolah gambar.

- **Implementasi penerimaan gambar**

Proses ini adalah proses menerima paket dari *node* pemantau dan menampilkan paket tersebut pada antar muka pengguna. Proses diawali dari input berupa pesan dari *node server* yang memberitahukan alamat IP-nya. Kemudian klien akan membalas pesan tersebut dengan balasan berupa pemberitahuan jika klien ini adalah *node viewer* bukan *node* perantara. Pesan-pesan tersebut dikirim melalui protokol UDP. Maka setelah itu akan mendapat pesan berupa gambar yang berbentuk string. Kemudian paket dibaca oleh *klien* dan ditampilkan pada antarmuka sederhana dengan memanfaatkan modul Tkinter. Pada Kode Sumber 4.4 merupakan implementasi menerima dan menampilkan gambar.

1	<code>import socket</code>
2	<code>import StringIO</code>
3	<code>import pickle</code>
4	<code>import Tkinter as tk</code>

5	<code>import PIL</code>
6	<code>from PIL import ImageTk</code>
7	<code>sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)</code>
8	<code>sock.bind(('0', 8888))</code>
9	<code>root = tk.Tk()</code>
10	<code>root.title("Aplikasi <i>Picture streaming</i>")</code>
11	<code>while True:</code>
12	<code>message, address = sock.recvfrom(65504)</code>
13	<code>message = pickle.loads(message)</code>
14	<code>if message == "I am server":</code>
15	<code>sock.sendto("I am viewer", address)</code>
16	<code>else:</code>
17	<code>seq, imgStr = message</code>
18	<code>imgStr = imgStr.strip()</code>
19	<code>path = str(seq) + ".jpg"</code>
20	<code>fh = open(path, "wb")</code>
21	<code>fh.write(imgStr.decode('base64'))</code>
22	<code>fh.close()</code>
23	<code>img = ImageTk.PhotoImage(PIL.Image.open(path))</code>
24	<code>panel = tk.Label(root, image = img, borderwidth=0)</code>
25	<code>panel.grid(row=0, column=1)</code>
26	<code>root.update()</code>

Kode Sumber 4.4 Kode sumber implementasi pada *server*

Pada Gambar 4.3 adalah hasil keluaran dari implementasi penerimaan gambar yang ditampilkan dengan menggunakan Tkinter Python.



Gambar 4.3 Hasil implementasi penerimaan gambar

BAB V

UJI COBA dan EVALUASI

Pada bab ini akan dibahas mengenai uji coba dari segi fungsionalitas dan performa dari sistem yang telah dirancang. Uji coba fungsionalitas dan performa akan dibagi ke dalam beberapa skenario uji coba.

5.1 Uji Coba Fungsionalitas

Uji coba fungsionalitas merupakan sebuah pengujian terhadap jalannya fungsi-fungsi utama yang ada pada sistem. Pada sistem ini yang terdiri dari *node-node* berupa mini komputer Raspberry Pi dan sebuah laptop, uji coba terbagi menjadi dua bagian utama, yaitu *picture streaming* dan uji coba fungsionalitas protokol *routing* Babel. *Picture streaming* diimplementasikan dengan menggunakan bahasa pemrograman Python. Program dijalankan pada mini komputer Raspberry Pi dan laptop dengan menggunakan terminal dari Raspbian Wheezy pada mini Komputer Raspberry Pi dan Linux Mint Nadia pada laptop. Pada uji coba ini akan terdapat empat skenario untuk menguji fungsionalitas sistem, meliputi.

1. *Picture streaming*

Implementasi *picture streaming* merupakan salah satu bagian inti dari sistem karena proses ini sebagai proses utama. Dalam implementasi *picture streaming* memiliki beberapa fungsionalitas antara lain:

- **Mengirim gambar**

Dalam mengirim gambar terdapat empat proses yang terlibat, yaitu menangkap gambar, mengirim informasi alamat *server*, menerima pesan balasan dari *viewer* dan melayani *viewer* dengan mengirimkan gambar ke *viewer*.

- **Menerima gambar**

Dalam proses ini terdapat dua proses yang terlibat antara lain menerima dan mengirim informasi dari dan ke *server* dan menampilkan gambar yang dikirim oleh *server*. Gambar pada proses ini akan ditampilkan pada antarmuka sederhana.

2. Protokol *Routing* Babel

Protokol *routing* Babel sebagai protokol yang digunakan dalam jaringan *wireless mesh*. Pengujian ini dilakukan untuk menguji apakah protokol Babel bekerja sebagai mestinya pada *node-node* mini komputer Raspberry Pi. Pengujian aplikasi *picture streaming* digunakan untuk menguji komunikasi pengiriman pesan antar *node*. Dari hasil pengujian tersebut diharapkan Tugas Akhir ini bisa diterapkan sebagai alternatif CCTV untuk memantau jalan raya.

5.1.1 Lingkungan Uji Coba

Pada subbab ini, dijelaskan mengenai gambaran lingkungan yang digunakan sebagai uji coba sistem. Uji coba dilakukan pada Laboratorium Teknik Informatika ITS. Lingkungan uji coba yang diuji memiliki spesifikasi sebagai berikut:

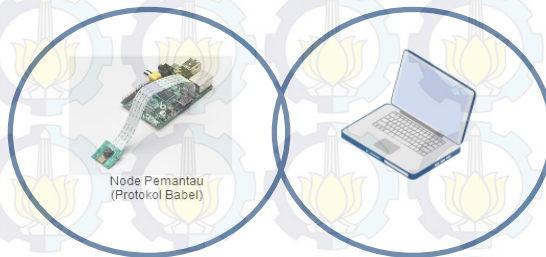
1. Ruang laboratorium Grid Computing Teknik Informatika ITS.
2. Ruang laboratorium pemrograman Teknik Informatika ITS.
3. Jalan raya ITS.
4. Perangkat *node* pemantau (mini komputer Raspberry Pi, WiFi *dongle*, modul kamera).
5. Perangkat *node* (mini komputer Raspberry Pi dan WiFi *dongle*).
6. *Power* adaptor dengan *output* 5V 1,5A untuk mini komputer Raspberry Pi.

7. Notebook HP Pavillion DV4T-1000, Intel(R) Core(TM)2 Duo CPU T9400 @2.4GHz dan 4GB memori DDR2 dengan sistem operasi Linux Mint Nadia.

5.1.2 Uji Coba *Picture streaming*

Pada uji coba *picture streaming* dilakukan pengiriman oleh satu *node* ke *node* yang lain. Tujuan dari hasil uji coba adalah untuk mengetahui apakah fungsi-fungsi pada implementasi *picture streaming* berjalan sesuai dengan yang dirancang. Selain itu uji coba ini juga mencatat persentase paket yang hilang dalam pengiriman selama satu menit.

Uji coba dilakukan dengan skenario terdapat dua *node*, *node server* sebagai pengirim gambar dan *node client* sebagai penerima gambar. Dua *node* diletakkan pada ruang tertutup. Dua *node* tersebut memiliki jarak 4 meter. *Node* saling terhubung dengan menggunakan jaringan *wireless mesh* menggunakan protokol *routing* Babel.



Gambar 5.1 dua *node* saling terhubung dengan protokol Babel

Dari hasil uji coba tersebut maka akan didapatkan hasil pengujian seperti pada Tabel 5.1.

Tabel 5.1 Hasil uji coba *picture streaming* dua *node*

No.	Paket terkirim	Paket diterima	Persentase keberhasilan
1.	168	157	93.45%
2.	168	156	92.86%

3.	165	154	93.33%
4.	170	161	94.71%
5.	171	158	92.40%

Paket terkirim pada Tabel 5.1 merupakan jumlah gambar yang dikirimkan ke *node* penerima. Jumlah gambar yang berhasil di tangkap oleh kamera pada *node* pemantau (*viewer*) berjumlah 157, 156, 154, 161 dan 158 dengan persentase rata-rata tingkat keberhasilan penerimaan paket adalah sebesar 93,35%.

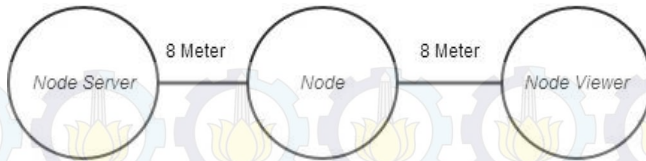
Pada uji coba ini berarti implementasi *picture streaming* yang diimplementasikan pada *node* pemantau maupun *node* penerima sudah sesuai dengan yang dirancang karena *node* pemantau sudah mampu mengirim dan *node viewer* mampu menerima gambar lalu menampilkannya pada antarmuka aplikasi. Selain itu uji coba ini juga akan menjadi tolok ukur untuk pengujian selanjutnya dengan melakukan Picture Streaming pada *multi-hop distances*.

5.1.3 Uji Coba *Picture streaming* pada *Multi-Hop Distances*

Uji coba *picture streaming* pada *Multi Hop Distances* dilakukan dengan tujuan apakah *node viewer* masih bisa mengetahui alamat IP *node server* jika *node* bertambah jaraknya menjadi *multi hop*. Pengiriman oleh satu *node* ke *node* yang lain. Selain itu uji coba ini juga untuk mengetahui apakah *routing* protokol Babel berjalan sesuai dengan mestinya hingga tiap *node* mampu untuk bertukar *routing table*. Selain itu uji coba ini juga akan mencatat persentase paket yang hilang dalam pengiriman selama satu menit untuk dibandingkan dengan hasil uji coba sebelumnya, yaitu uji coba *picture streaming* pada 2 *node*. Pada uji coba ini akan dilakukan dengan 2 skenario topologi yang berbeda yaitu skenario 1 *hop* dan 2 *hop*.

1. Skenario 1 hop

Pada skenario ini, dibuatlah topologi dengan jarak antara *node server* (pemantau) dengan *node viewer* memiliki jarak 1 *hop*. Topologi pada skenario ini adalah seperti pada Gambar 5.2



Gambar 5.2 Skenario topologi 1 hop

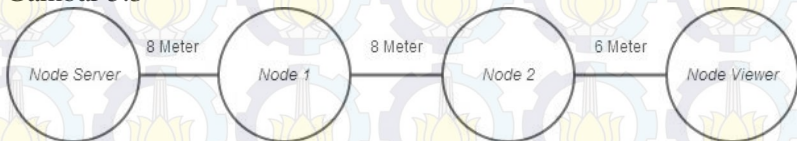
Jarak 8 meter didapatkan dari jarak maksimal sinyal WiFi Ad-Hoc bisa ditangkap oleh masing-masing Raspberry Pi. Jarak ini didapatkan pada hasil uji coba jarak sinyal antar *node* pada subbab 5.2.3. Hasil uji coba pada skenario ini adalah seperti pada Tabel 5.2.

Tabel 5.2 Hasil uji coba skenario 1 hop

No.	Paket terkirim	Paket diterima	Persentase keberhasilan
1.	167	155	92.81%
2.	165	154	93.33%
3.	168	157	93.45%
4.	168	156	92.86%
5.	170	158	92.94%

2. Skenario 2 hop

Pada skenario ini dibuatlah topologi dengan jarak antara *Node Server* dengan *Node Viewer* adalah 2 hop. Jarak antara *Node Server* dengan *Node Viewer* sendiri adalah sekitar 22 meter dengan rincian jarak *Node Server* dengan *Node 1* berjarak 8 meter, sedangkan jarak *Node 1* dan *Node 2* berjarak 8 meter dan jarak *Node 2* dengan *Node Viewer* adalah 6 meter. Skenario topologi dapat dilihat pada Gambar 5.3



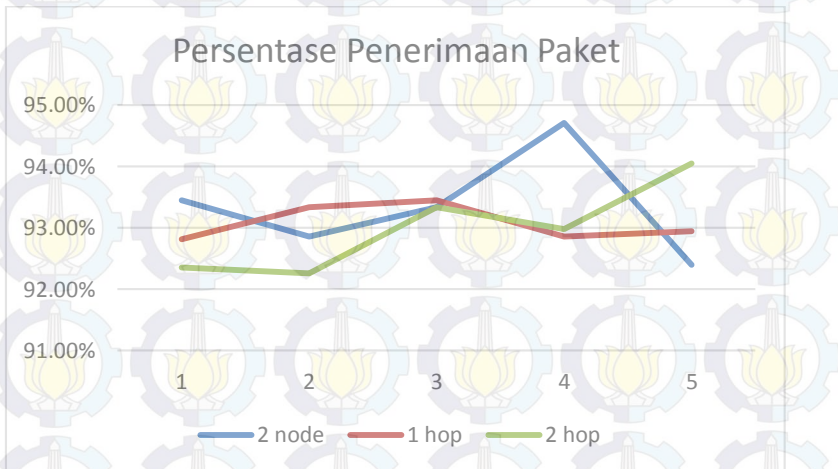
Gambar 5.3 Skenario topologi 2 hop

Pada Tabel 5.3 ditunjukkan hasil uji coba picture streaming pada skenario 2 hop.

Tabel 5.3 Hasil uji coba skenario 2 hop

No.	Paket terkirim	Paket diterima	Persentase keberhasilan
1.	170	157	92.35%
2.	168	155	92.26%
3.	165	154	93.33%
4.	171	159	92.98%
5.	168	158	94.05%

Maka dapat dibandingkan bahwa hasil uji coba picture streaming pada multi-hop memiliki sedikit penurunan performa jika dibandingkan dengan hasil uji coba sebelumnya. Hal ini tidak menjadi masalah karena penurunannya sangat kecil sekali. Pada Gambr 5.4 merupakan hasil perbandingan persentase pada pengujian picture streaming ini.



Gambar 5.4 Perbandingan persentase penerimaan paket tiap topologi

5.2 Uji Coba Performa

Uji coba performa dilakukan untuk melihat perilaku yang dihasilkan oleh sistem pada Tugas Akhir ini. Uji coba ini bertujuan untuk mengetahui seberapa besar pemakaian CPU dan memori pada saat menjalankan fitur-fitur yang ada pada Tugas Akhir ini.

5.2.1 Penggunaan CPU dan Memori pada Mini Komputer Raspberry Pi

Uji coba ini dilakukan untuk mengukur seberapa besar pemakaian dari kapasitas CPU dan memori yang digunakan oleh prosesor mini komputer Raspberry Pi saat melakukan *picture streaming*. Pada uji coba ini digunakan aplikasi yang berjalan pada terminal Raspbian Wheezy yang bernama SAR (*System Activity Report*) yaitu suatu aplikasi yang digunakan untuk memantau kinerja suatu proses dari perangkat CPU. Sedangkan untuk mengukur tingkat pemakaian memori pada mini komputer Raspberry Pi digunakan perintah “free”. Uji coba ini dilakukan pada saat aplikasi *picture streaming* dan protokol Babel berjalan dan pada saat tidak menggunakannya. Hal ini dikarenakan agar dapat mengetahui perbedaan saat pemakaian dan tidak.

1. Uji coba penggunaan CPU

Pada uji coba penggunaan CPU ini akan dibandingkan hasil dari penggunaan CPU oleh node server pada Raspberry Pi. Uji coba ini dilakukan untuk melihat perbedaan persentase penggunaan CPU saat program dijalankan dengan melayani 1 pengguna atau lebih.

Pada Tabel 5.4 ditunjukkan data penggunaan CPU mini komputer Raspberry Pi pada saat melayani *viewer* yang aktif. Pengukuran dilakukan selama 5 kali dengan pencatatan satu kali setiap detik. Pencatatan dilakukan ketika terdapat 1 *viewer* dan 2 *viewer* yang sedang dilayani oleh *server*.

Pada Tabel 5.4 terlihat bahwa pemakaian CPU saat melakukan *picture streaming* cukup tinggi baik saat sedang melayani satu *viewer* ataupun saat melayani multi *viewer*. Hal ini dikarenakan perangkat Raspberry Pi selalu meminta respon untuk memperbarui gambar yang ditangkap secara terus-menerus. Hal ini juga membuat mini komputer Raspberry Pi ini menjadi cepat panas.

Tabel 5.4 Hasil uji coba penggunaan CPU

No	Presentase penggunaan CPU (%)		
	Nonaktif	<i>Single Viewer</i>	<i>Multi Viewer</i>
1	0	76	81
2	1	82	70
3	0	35	75
4	1	84	67
5	0	68	82

Pada Tabel 5.4 terlihat bahwa perbedaan penggunaan CPU saat melayani *single viewer* dan *multi viewer* sedikit berbeda persentase rata-ratanya. Saat melayani satu pengguna rata-rata penggunaan CPU adalah 69% sedangkan saat melayani dua pengguna rata-ratanya adalah 75%. Dengan kemampuan dari CPU mini komputer Raspberry Pi yang hanya sebesar 700 Mhz maka *picture streaming* dengan melayani lebih dari dua pengguna akan menguras kinerja dari CPU.

2. Uji Coba penggunaan memori

Setelah menguji penggunaan CPU pada saat melakukan *picture streaming* yang hasilnya cukup besar, maka kemudian penulis akan menguji coba seberapa banyak penggunaan memori pada saat melayani *single viewer* ataupun *multi viewer*. Pada uji coba ini penulis menggunakan aplikasi yang berjalan pada *terminal Linux* yang dapat dijalankan dengan perintah “free”. Skenario uji

cobanya adalah pengukuran penggunaan memori dilakukan pada saat *picture streaming* aktif melayani *single viewer* ataupun *multi viewer* dan pada saat *picture streaming* non-aktif. Hal ini dilakukan agar terlihat perbedaan penggunaan memori pada saat menggunakan fitur *picture streaming* ini dan pada saat tidak menggunakannya.

Pada Tabel 5.5 ditunjukkan bahwa pemakaian memori pada saat *picture streaming* non-aktif adalah 60MB, pemakaian memori yang terpakai tersebut adalah untuk menjalankan sistem operasi yang sedang digunakan pada mini komputer Raspberry Pi. Sedangkan pemakaian memori saat sedang aktif melayani satu pengguna adalah 74 MB dan saat melayani lebih dari satu adalah 77 MB.

Maka dalam hal *picture streaming* bisa disimpulkan bahwa pemakaian memori tidak banyak yang terpakai saat melakukan *picture streaming* namun penggunaan CPU lah yang paling banyak terpakai pada saat fitur *picture streaming* ini dijalankan.

Tabel 5.5 Tabel pemakaian memori

No	Pemakaian memori (MB)		
	<i>Picture streaming non-aktif</i>	<i>Picture streaming single viewer</i>	<i>Picture streaming multi viewer</i>
1	60	74	77

5.2.2 Uji Coba Jarak Sinyal Antar *node*

Pada implementasi *picture streaming* pada jaringan mesh ini maka tidak lepas dari kemampuan untuk menangkap sinyal jaringan Ad-Hoc oleh tiap *node*. Pengujian ini dilakukan untuk mengetahui jarak maksimal yang mampu dijangkau oleh antar *node* pada jaringan mesh menggunakan protokol Babel. Untuk menangkap sinyal sendiri pada masing-masing *node* digunakan 150 Mbps *wireless 802.11b/g/n nano USB adapter* Edimax EW-

7811un. Pada uji coba ini akan dilakukan pengukuran seberapa jauh sinyal mampu menjangkau *node* yang lainnya.

Tabel 5.6 uji coba sinyal WiFi Ad-Hoc antar *node*

No	Jarak Antar <i>Node</i> (meter)	Status (terhubung / tidak terhubung)
1	2 m	Terhubung
2	4 m	Terhubung
3	6 m	Terhubung
4	8 m	Terhubung
5	10 m	Tidak terhubung

Pada Tabel 5.6 ditunjukkan hasil pengujian kekuatan sinyal wireless WiFi USB adapter yang digunakan pada mini komputer Raspberry Pi. Pada pengujian ini untuk pengecekan antar *node* terhubung atau tidak dilakukan dengan menjalankan daemon dari protokol Babel, jika antar *node* tidak terhubung maka *node* tidak akan saling bertukar *routing table* dan tidak dapat melakukan ping satu sama lain.

5.2.3 Uji Coba Daya Tahan Baterai untuk Pemantauan pada Jalan Raya

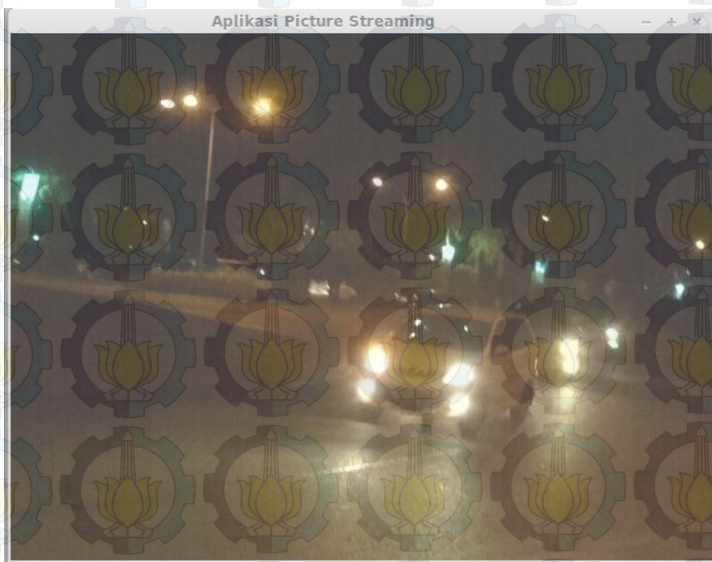
Pada uji coba ini ditujukan untuk mengukur tingkat ketahanan baterai dengan menggunakan powerbank dengan ukuran 5600mah dan memiliki keluaran tenaga listrik sekitar 1 Ampere. Uji coba ini juga untuk mengetahui apakah sistem masih bekerja dengan baik jika daya diganti menggunakan baterai.

Pada Tabel 5.7 berikut merupakan hasil uji coba untuk mengetahui ketahanan baterai pada *node pemantau* yang digunakan untuk menangkap gambar dan *node perantara* yang digunakan untuk menyalurkan gambar ke *node tujuan (viewer)*.

Tabel 5.7 Tabel durasi penggunaan baterai

No.	Durasi daya tahan <i>powerbank</i> yang digunakan oleh sistem <i>picture streaming</i>	
	Node Pemantau	Node Perantara
1	360 menit	480 menit

Pada Gambar 5.5 merupakan hasil tangkapan gambar di jalan raya pada saat sistem menggunakan baterai.

**Gambar 5.5 Hasil tangkapan sistem pada jalan raya**

BAB VI PENUTUP

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap implementasi *picture streaming* pada jaringan *mesh* berbasis protokol Babel menggunakan Raspberry Pi untuk pemantauan jalan raya dapat diambil kesimpulan sebagai berikut:

1. Implementasi *picture streaming* pada jaringan *mesh* menggunakan protokol Babel menghasilkan persentase rata-rata paket gambar yang dikirim mencapai 93%.
2. Dengan jarak sinyal WiFi mencapai 8 meter, maka dalam penerapan di jalan raya membutuhkan lebih banyak node jika ditempatkan pada tiang listrik.
3. Distribusi citra (gambar) kepada node viewer menghasilkan jumlah *frame-per-second* terbaik adalah 2,66 FPS.
4. Persentase penerimaan gambar menjadi sedikit menurun ketika jumlah *hop* bertambah.

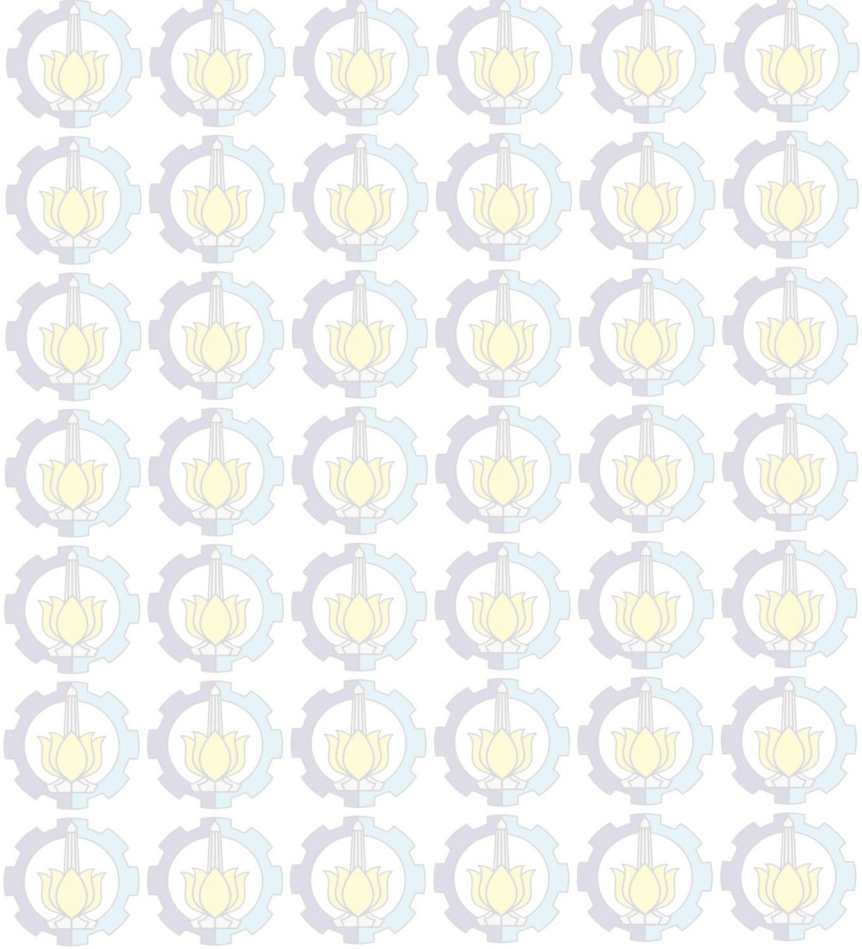
6.2 Saran

Saran yang diberikan untuk pengembangan *picture streaming* pada jaringan *mesh* berbasis protokol Babel menggunakan Raspberry Pi:

1. Untuk pengembangan lebih lanjut, diharapkan menambah aspek keamanan jaringan sehingga tidak sembarang orang bisa mengakses perangkat *node* pemantau (*server*).

2. Untuk pengembangan lebih lanjut, diharapkan menggunakan kamera dengan kecepatan lebih tinggi sehingga dapat menangkap gambar lebih banyak setiap detiknya.

Untuk pengembangan penelitian ini ada baiknya dibuatkan suatu sistem yang *user friendly* sehingga pihak-pihak yang memiliki keperluan dapat menerapkan sistem ini dalam kehidupan mereka.



DAFTAR PUSTAKA

- [1] JaringanKomputer.org, “Topologi Mesh - Kelebihan Dan Kekurangan Topologi jaringan Mesh,” [Online]. Available: <http://www.jaringankomputer.org/topologimesh-kelebihan-dan-kekurangan-topologijaringanmesh/>. [Diakses 25 Februari 2014].
- [2] J. Chroboczek, “The Babel Routing Protocol,” [Online]. Available: <http://tools.ietf.org/html/rfc6126>. [Diakses 9 Maret 2014].
- [3] RaspberryPi, “Raspberry Pi: FAQs,” [Online]. Available: <http://www.raspberrypi.org/faqs#introWhatIs>. [Diakses 24 Februari 2014].
- [4] D. Ross, “HowStuffWorks.com,” 20 Juni 2007. [Online]. Available: <http://computer.howstuffworks.com/how-wireless-mesh-networks-work.htm>. [Diakses 31 Mei 2014].
- [5] J. Chroboczek, “Babel -- a loop-avoiding distance-vector routing protocol,” [Online]. Available: <http://www.pps.univ-paris-diderot.fr/~jch/software/babel/>. [Diakses 25 Februari 2014].

BIODATA PENULIS



Muhtarom Widodo, lahir di Surabaya, pada tanggal 4 Oktober 1991. Penulis menempuh pendidikan mulai dari SDN Kalirungkut I (1998-2004), SMPN 12 Surabaya (2004-2007), SMAN 16 Surabaya (2007-2010) dan S1 Teknik Informatika ITS (2010-2014).

Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC). Diantaranya adalah menjadi staff departemen Pengembangan Sumber Daya Manusia dan sebagai Kepala Departemen Hubungan Luar Himpunan

Mahasiswa Teknik Computer-Informatika ITS. Penulis juga aktif dalam organisasi Badan Eksekutif Mahasiswa (BEM) ITS sebagai staff Hubungan Luar.

Selama kuliah di teknik informatika ITS penulis mengambil bidang minat Komputasi Berbasis Jaringan (KBJ). Komunikasi dengan penulis melalui : muhtaromw@live.com.

LAMPIRAN

1. Memasang Sistem Operasi Raspbian Wheezy

Memasang sistem operasi pada mini komputer Raspberry Pi adalah hal pertama yang wajib dilakukan karena tanpa adanya sistem operasi maka perangkat Raspberry Pi tidak akan dapat bekerja sebagai mana mestinya. Untuk melakukan instalasi sistem operasi Raspbian Wheezy pada Raspberry Pi akan dijelaskan melalui langkah-langkah sebagai berikut:

1. Siapkan sebuah SDHC card dengan ukuran minimal 4GB.
2. Unduh *image* sistem operasi Raspberry Pi, yaitu Raspbian Wheezy dengan tanggal rilis 26 juli 2013.
3. Ekstrak file yang telah diunduh, sehingga terdapat *file* dengan format *.img*.
4. Unduh aplikasi Win32 Disk Imager untuk melakukan instalasi Raspbian Wheezy pada SDHC *card*.
5. Ekstrak aplikasi Win32 Disk Imager.



Gambar A.1 Aplikasi Win32 Disk Imager

6. Pada Gambar diatas ditunjukkan tampilan aplikasi Win32 Disk Imager, pilih *drive* SD *card* yang ingin di-format dan sorot berkas *image* dari sistem operasi Raspbian Wheezy.

7. Klik *Write* dan tunggu sampai prosesnya selesai dan SD *card* siap digunakan untuk menjalankan sistem operasi Raspberry Pi.

2. Inisialisasi Modul Kamera

Dikarenakan modul kamera adalah alat yang paling diperlukan pada sistem ini karena fungsinya yang sangat dibutuhkan untuk melakukan pemantauan jalan. Sebelumnya maka dari itu dilakukan proses inisialisasi modul kamera terhadap mini komputer Raspberry Pi agar nantinya dapat berjalan dengan sempurna. Untuk melakukan pengecekan apakah modul sudah terpasang dengan benar, lakukan perintah berikut ini:

- Ketikkan perintah berikut pada konsol terminal `$raspistill -o gambar.jpg`
- Kemudian jika kamera sudah terpasang dengan benar, maka akan ada *preview* pada layar monitor sebelum gambar ditangkap oleh kamera. Gambar 4.4 merupakan hasil tangkapan kamera mini komputer Raspberry Pi.



Gambar A.2 Hasil tangkapan kamera Raspberry Pi

3. Mengkonfigurasi Wifi

Setelah instalasi sistem operasi Raspbian Wheezy pada SD card, perlu untuk mendapatkan koneksi internet karena dibutuhkan paket-paket yang harus diunduh. Untuk melakukan proses ini harus mengatur WiFi dongle Edimax terlebih dahulu. Berikut ini adalah cara setting wifi untuk membuat koneksi dengan internet:

- Setelah booting, login ke console menggunakan *username* pi dan kata sandi raspberry.
- Pada konsol ketik perintah berikut:
\$sudo nano /etc/network/interfaces
- Pada nano text editor, akan dilihat konfigurasi awal sebagai berikut:

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp
```

Gambar A.3 Konfigurasi dasar ethernet

- Pada Gambar diatas merupakan konfigurasi dasar untuk menghubungkan ke Ethernet, sehingga perlu ditambahkan sedikit konfigurasi agar bisa menggunakan WiFi dongle. Tambahkan konfigurasi seperti pada gambar A.4. kemudian simpan konfigurasi tersebut dengan menekan CTRL+X.
- Kemudian pada konsol ketik perintah berikut:
\$sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

```
allow-hotplug wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Gambar A.4 Konfigurasi WiFi dongle

- Jika file yang dibuka kosong maka ketikkan konfigurasi seperti pada Gambar A.5. Konfigurasi ini untuk menghubungkan WiFi dongle dengan SSID tertentu. Jika sudah maka jangan lupa untuk menyimpan berkas dengan menggunakan perintah CTRL + X kemudian tekan enter. Setelah itu pada konsol lakukan reboot dengan perintah sudo reboot.

```
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev
update_config=1
network={
ssid="YOURSSID"
psk="YOURPASSWORD"
#tipe protokol: RSN untuk WP2
#dan WPA untuk WP1
proto=WPA
# tipe manajemen kunci, bisa WPA-PSK atau
#WPA-EAP (Pre-Shared or Enterprise)
key_mgmt=WPA-PSK
# Pairwise bisa CCMP
# atau TKIP (for WPA2 or WPA1)
pairwise=TKIP
auth_alg=OPEN
}
```

Gambar A.5 Konfigurasi WiFi

4. Instalasi PiCamera

Picamera dapat digunakan untuk berbagai macam keperluan yang menggunakan keberadaan dari modul kamera Raspberry Pi. Picamera adalah paket yang menyediakan Python *interface* untuk modul kamera Raspberry Pi dengan basis Python versi 2.7 atau versi di atasnya dan Python versi 3.2 atau versi di atasnya. Dengan

menggunakan Picamera maka perintah modul kamera Raspberry Pi yang berbasis *command line* pada terminal dapat dieksekusi melalui Python *script*.

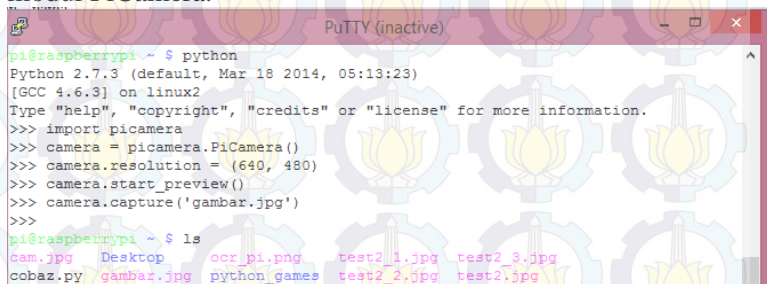
Untuk memasang PiCamera tuliskan perintah berikut:

- \$sudo apt-get install python-pip
Python pip adalah paket dari Python untuk digunakan memasang modul-modul python dengan mudah. Pip berfungsi layaknya seperti apt-get.
- \$sudo pip install picamera

Setelah modul PiCamera sudah terpasang lakukan ujicoba dengan mengetikkan langkah-langkah berikut pada terminal:

```
$python
>>>import picamera
>>>camera = picamera.PiCamera()
>>>camera.resolution = (640, 480)
>>>camera.start_preview()
>>>camera.capture('gambar.jpg')
```

Jika tidak ada *error* saat *import* modul picamera maka modul sudah terpasang dengan benar. *Script* di atas akan menyimpan hasil tangkapan kamera dengan nama gambar.jpg. Pada Gambar berikut merupakan gambar dari contoh penggunaan modul PiCamera.



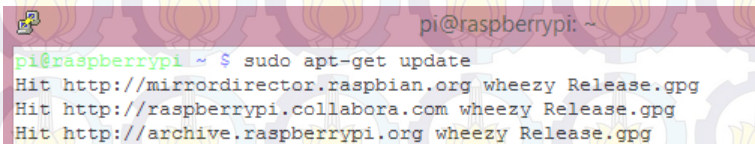
```
pi@raspberrypi ~ $ python
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import picamera
>>> camera = picamera.PiCamera()
>>> camera.resolution = (640, 480)
>>> camera.start_preview()
>>> camera.capture('gambar.jpg')
>>>
pi@raspberrypi ~ $ ls
cam.jpg  Desktop  ocr_pi.png  test2_1.jpg  test2_3.jpg
cobaz.py gambar.jpg python_games test2_2.jpg  test2.jpg
```

Gambar A.6 Contoh penggunaan PiCamera

- Kemudian setelah itu pasang modul pillow, modul pillow adalah modul untuk pengolahan gambar yang kompatibel dengan Python. Ketikkan perintah berikut untuk memasang pillow:
\$sudo pip install pillow
- Lalu pasang modul libjpeg-dev sebagai decoder jpeg dengan mengetikkan perintah berikut
\$sudo apt-get install libjpeg-dev

5. Implementasi protokol Babel

Untuk mengimplementasikan protokol Babel maka akan penulis gunakan paket bernama Babeld, yaitu paket daemon untuk mengimplementasikan protokol routing Babel. Paket ini dibuat oleh perancang protokol Babel yaitu Julius Chrobocek. Untuk memasang paket Babel pertama kali yang dilakukan adalah update paket-paket pada Raspbian Wheezy terlebih dahulu, ketikkan perintah seperti pada Gambar A.7.



```
pi@raspberrypi ~ $ sudo apt-get update
Hit http://mirrordirector.raspbian.org wheezy Release.gpg
Hit http://raspberrypi.collabora.com wheezy Release.gpg
Hit http://archive.raspberrypi.org wheezy Release.gpg
```

Gambar A.7 Perintah *update repository*

Setelah melakukan *update repository* install Babel dengan perintah seperti pada Gambar A.8.

```
$sudo apt-get install babeld
```

Gambar A.8 Perintah instalasi Babeld

- **Konfigurasi protokol Babel**
Protokol routing Babel pada masing-masing *node* menggunakan program shell script untuk berjalan secara otomatis

setiap kali *node* menyala atau dijalankan. Agar Babel dapat berjalan dalam jaringan mesh, akan dijelaskan seperti berikut ini.

1. Setting ulang konfigurasi wifi pada `/etc/network/interfaces` untuk membuat jaringan *ad-hoc* dengan mengetik perintah pada konsol **\$sudo nano /etc/network/interfaces**.

```
auto wlan0
iface wlan0 inet manual
address 172.168.x.x
netmask 255.255.255.255
wireless-channel 1
wireless-ssid NamaAdHoc
wireless-mode ad-hoc
```

Gambar A.9 Konfigurasi jaringan Ad-Hoc

Kemudian lakukan konfigurasi seperti Gambar 4.5 dan kemudian lakukan *reboot*.

2. Aktifkan modul IPv6 dengan mengubah konfigurasi pada konfigurasi modprobe dengan mengetikkan perintah pada konsol sebagai berikut.

\$sudo nano /etc/modprobe.d/ipv6.conf

Lalu konfigurasi seperti pada gambar berikut.

```
#alias net-pf-10 off
#alias ipv6 off
```

Gambar A.10 Konfigurasi IPv6 modprobe

3. Konfigurasi seperti pada Gambar A.9 juga bisa dilakukan dengan mengetikkan perintah pada konsol seperti pada Gambar A.11.

```
$sudo iwconfig wlan0 mode ad-hoc channel 1
      essid NamaAdhoc
$sudo ip addr add 172.168.x.x/32 dev wlan0
```

Gambar A.11 Konfigurasi ad-hoc pada terminal

4. Kemudian jalankan Babel dengan perintah
\$sudo babeld -D wlan

Ulangi perintah-perintah diatas untuk setiap *node* yang akan diimplementasikan dengan menggunakan Babel protokol. Pada Gambar A.12 adalah keadaan saat Babeld dijalankan.

```
Terminal
File Edit View Search Terminal Tabs Help
Terminal x Terminal x Terminal x
172.168.0.1/32 metric 581 (532) refmetric 0 id 02:1f:02:ff:fe:21:21:d5 seqno 583
09 age 25 via wlan0 neigh fe80::821f:2ff:fe21:21d5 nexthop 172.168.0.1 (installe
d)
(flushing 24 buffered bytes on wlan0)

My id 02:21:5d:ff:fe:0f:f7:2e seqno 8979
Neighbour fe80::821f:2ff:fe21:21d5 dev wlan0 reach 7ff8 rxcost 341 txcost 436 ch
an 1.
172.168.0.2/32 metric 0 (exported)
172.168.0.1/32 metric 581 (532) refmetric 0 id 02:1f:02:ff:fe:21:21:d5 seqno 583
09 age 25 via wlan0 neigh fe80::821f:2ff:fe21:21d5 nexthop 172.168.0.1 (installe
d)
Checking neighbours.
Sending self update to wlan0.
Sending update to wlan0 for 172.168.0.2/32.

My id 02:21:5d:ff:fe:0f:f7:2e seqno 8979
Neighbour fe80::821f:2ff:fe21:21d5 dev wlan0 reach 3ffc rxcost 512 txcost 436 ch
an 1.
172.168.0.2/32 metric 0 (exported)
172.168.0.1/32 metric 872 (549) refmetric 0 id 02:1f:02:ff:fe:21:21:d5 seqno 583
09 age 27 via wlan0 neigh fe80::821f:2ff:fe21:21d5 nexthop 172.168.0.1 (installe
d)
█
```

Gambar A.12 Babeld saat dijalankan

- **Pengalamatan *Node***

Pada pengujian akan terdapat 5 *node* yang ditempatkan pada posisi tertentu agar sesuai dengan topologi jaringan yang dikehendaki, empat *node* nantinya akan memiliki pengalamatan IP yang dinamis dan satu *node* sisanya memiliki alamat IP yang statis digunakan sebagai *server* untuk menjalankan *Ad-Hoc Configuration Protocol* (AHCP). AHCP memudahkan kita untuk memberi alamat IP pada jaringan *mesh*, dimana hal itu masih belum bisa dilakukan oleh DHCP.

Langkah-langkah untuk konfigurasi AHCP akan dijabarkan sebagai berikut:

1. Buat *file* dengan nama *ahcpd.conf* pada direktori */etc/ file* ini akan berisi konfigurasi range IP yang diberikan pada *node-node* nantinya. Lakukan dengan perintah berikut
`$sudo nano /etc/ahcpd.conf` ketikkan konfigurasi seperti pada Gambar A.13

```
mode server
prefix 172.168.1.128/25
lease-dir /var/lib/leases
name-server 172.168.1.1
```

Gambar A.13 Konfigurasi AHCP

2. AHCP tidak akan mampu untuk dijalankan jika folder */var/lib/leases* tidak ada. Maka buat direktori baru untuk digunakan menyimpan alamat IP yang sudah diberikan dengan perintah
`$sudo mkdir /var/lib/leases`
3. Kemudian gunakan *chmod* agar script tersebut bisa dieksekusi
`$sudo chmod 755 /etc/ahcpd.conf`
4. Eksekusi program *ahcp* dengan perintah berikut
`$sudo ahcpd -c /etc/ahcpd.conf -D wlan0`

Pada langkah-langkah diatas akan memberikan IP dengan range antara 172.168.1.128 hingga 172.168.1.255. Untuk *node* lainnya jika ingin meminta alamat IP tinggal menjalankan perintah \$sudo ahcpd -D wlan0 pada terminal.

- **Penjelasan Babel Log**

Pada Gambar A.14 merupakan log dari Babel ketika Babel dijalankan.

```
My id ba:27:eb:ff:fe:54:48:c5 seqno 47376
192.168.1.102/32 metric 0 (exported)
Creating neighbour fe80::ba76:3fff:feca:cb49 on eth0.
Sending hello 11468 (400) to eth0.
Sending ihu 65535 on eth0 to fe80::ba76:3fff:feca:cb49.
Received hello 3261 (400) from fe80::ba76:3fff:feca:cb49 on eth0.
(flushing 24 buffered bytes on eth0)
Sending hello 11469 (400) to eth0.
Sending ihu 65535 on eth0 to fe80::ba76:3fff:feca:cb49.
Received request for any from fe80::ba76:3fff:feca:cb49 on eth0.
Sending ihu 65535 on eth0 to fe80::ba76:3fff:feca:cb49.
```

Gambar A.14 Babel log saat daemon berjalan

Berikut adalah penjelasan dari log saat Babel daemon berjalan.

- “My id ba:27:eb:ff:fe:54:48:c5” merupakan ID yang diambil oleh Babel melalui IPv6 dari link local interface yang sedang menjalankan Babel, pada kasus ini adalah mengambil pada interface eth0.
- “Creating neighbour fe80:ba76:3fff:feca:cb49 on eth0” merupakan proses mengetahui node tetangga yang terhubung melalui jaringan ad-hoc.
- “Sending hello 11468 to eth0” merupakan proses mengirimkan pesan pada tetangga melalui eth0. 11468 adalah merupakan *sequence number* yang dikirimkan ke tetangga beserta pesan *hello* tersebut.

- “Sending ihu 65535 on eth0 to fe80:ba76:3fff:feca:cb49” berarti node ini telah menerima pesan hello dari fe80:ba76:3fff:feca:cb49 dan kemudian mengirim pesan ihu (*I heard you*) dengan perhitungan metric 65535.
- Pada Gambar A.15 merupakan log dari iterasi kedua, terlihat rxcost dan txcost bernilai 65535 sesuai dengan metrik yang dikirim pada pesan ihu. Kemudian node ini menerima pesan hello dengan sequence number 38495 dan membalas dengan mengirim ihu dengan perhitungan metrik yang baru yaitu 96

```
My id ba:27:eb:ff:fe:54:48:c5 seqno 47378
Neighbour fe80::ba76:3fff:feca:cb49 dev eth0 reach 8000 rxcost 65535 txcost 65535
chan -2.
192.168.1.102/32 metric 0 (exported)
Received hello 38495 (400) from fe80::ba76:3fff:feca:cb49 on eth0.
(flushing 40 buffered bytes on eth0)
Sending hello 47479 (400) to eth0.
Sending ihu 96 on eth0 to fe80::ba76:3fff:feca:cb49.
```

Gambar A.15 Log saat menerima pesan hello kedua

- Pada Gambar A.16 berikut adalah saat node memperbarui rutenya dan memasukkan alamat 192.168.1.105 sebagai *gateway* pada saat menerima *router-id*.

```
-----
Received hello 47914 (400) from fe80::ba76:3fff:feca:cb49 on eth0.
Received update/id for ::/0 from fe80::ba76:3fff:feca:cb49 on eth0.
Received nh 192.168.1.105 (1) from fe80::ba76:3fff:feca:cb49 on eth0.
Received router-id ba:76:3f:ff:fe:ca:cb:49 from fe80::ba76:3fff:feca:cb49 on eth0.
Received update for 192.168.1.105/32 from fe80::ba76:3fff:feca:cb49 on eth0.
Sending update to eth0 for 192.168.1.105/32
Received request for any from fe80::ba76:3fff:feca:cb49 on eth0.
Sending unicast ihu 96 on eth0 to fe80::ba76:3fff:feca:cb49.
```

Gambar A.16 Log saat memperbarui rute

- Sebelum menerima *update* dari *node* tetangga untuk memperbarui *routing table*, maka *node* akan menerima hingga 7 *hello message*. Dengan rincian *hello* sebagai berikut ini:
 - *Hello* pertama adalah untuk inisialisasi tetangga dengan menggunakan ipv6.
 - Pesan *hello* kedua dan ketiga digunakan untuk mengetahui *cost* dari *node*, pada saat ini *cost* masih belum *feasible* yaitu 65535.
 - Pesan *hello* keempat hingga keenam, *cost* sudah berubah menjadi *feasible* dengan *cost* < 65535.
 - Pesan *hello* ketujuh menerima *update* untuk memasukkan rute *node* (ipv4) ke dalam *routing table*.