TUGAS AKHIR – KI141502

PENGEMBANGAN SISTEM PENERANGAN LAMPU JALAN ADAPTIF BERDASARKAN METODE TALISMAN UNTUK STUDI KASUS MULTI-PEDESTRIAN DETECTOR

MUHAMMAD IQBAL TANJUNG
NRP 5112 100 069

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2016

TUGAS AKHIR – KI141502

PENGEMBANGAN SISTEM PENERANGAN LAMPU JALAN ADAPTIF BERDASARKAN METODE TALISMAN UNTUK STUDI KASUS MULTI-PEDESTRIAN DETECTOR

MUHAMMAD IQBAL TANJUNG
NRP 5112 100 069

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

(Halaman ini sengaja dikosongkan)

FINAL PROJECT– KI141502

DEVELOPMENT OF TALISMAN BASED ADAPTIVE STREET LIGHTING SYSTEM IN MULTI-PEDESTRIAN DETECTOR CASE STUDY

MUHAMMAD IQBAL TANJUNG
NRP 5112 100 069

First Advisor
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Second Advisor
Ir. Muchammad Husni, M.Kom

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2016

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENGEMBANGAN SISTEM PENERANGAN LAMPU JALAN ADAPTIF BERDASARKAN METODE TALISMAN UNTUK STUDI KASUS MULTI-PEDESTRIAN DETECTOR

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:
MUHAMMAD IQBAL TANJUNG
NRP. 5112 100 069

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Waskitho Wibisono, S.Kom., M.Eng., Ph.D. .....
NIP: 197410222000031001 (pembimbing 1)

Ir. Muchammad Husni, M.Kom. .....
NIP: 196002211984031001 (pembimbing 2)

SURABAYA
JUNI, 2016

(Halaman ini sengaja dikosongkan)

PENGEMBANGAN SISTEM PENERANGAN LAMPU JALAN ADAPTIF BERDASARKAN METODE TALISMAN UNTUK STUDI KASUS MULTI-PEDESTRIAN DETECTOR

Nama Mahasiswa : Muhammad Iqbal Tanjung
NRP : 5112 100 069
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
Dosen Pembimbing II : Ir. Muchammad Husni, M.Kom.

ABSTRAK

*Lampu penerangan jalan umum yang ada pada saat ini akan tetap menyala dengan tingkat pencahayaan 100% walaupun tidak ada pejalan kaki ataupun kendaraan umum yang melewati sistem. Hal tersebut dapat menyebabkan borosnya daya listrik dan umur dari lampu. Metode TALiSMaN (**T**raffic-**A**ware **L**ighting **S**cheme **M**anagement **N**etwork) adalah metode pengaturan kecerahan pada lampu jalan berdasarkan ada tidaknya objek yang melewati lampu penerangan jalan raya. Metode ini dibuat berdasarkan penelitian bagaimana kebutuhan dari pengguna jalan akan kenyamanan dan keamanan ketika melewati jalan yang diterangi oleh lampu penerangan jalan. Set dari beberapa lampu penerangan jalan akan menyala hanya ketika ada manusia yang melewati salah satu lampu.*

Pada penelitian sebelumnya pengiriman data antar node dilakukan menggunakan jaringan GSM dengan satu master utama untuk mengatur tingkat pencahayaan node-node sistem. Pada penelitian ini penulis, ingin membuat sistem penerangan jalan adaptif yang dapat beradaptasi apabila terdapat lebih dari 1 node yang mendeteksi pejalan kaki. Untuk membantu pengendalian tersebut diperlukan teknologi mikrokontroler pada tiap node dan modul radio dan sensor. Pengiriman data antar node menggunakan modul radio nRF24L01+ *dengan pengiriman data*

multi hop. Untuk mendeteksi keberadaan manusia menggunakan sensor PIR, sehingga tiap node bisa mendeteksi adanya pejalan kaki yang melewati sistem. Pencahayaan pada sistem akan berubah apabila ada pejalan kaki terdeteksi pada sebuah node. Tingkat pencahaayaan tersebut adalah, node yang mendeteksi S_n menyala 100%, node S_{n-1} dan S_{n+1} menyala 67%, dan node S_{n-2} dan S_{n+2} menyala 30% .

Sistem akan diuji coba, baik dari fungsionalitas dan performa menggunakan skenario-skenario yang telah ditentukan. Berdasarkan hasil uji coba, fungsionalitas sistem berjalan dengan baik dan akurasi dari pengiriman dan pengiriman data secara muti hop sebesar 75%, single hop dua arah sebesar 90% serta single hop satu arah dengan jarak 10 m adalah 98%, 20 m adalah 98% dan 30 m adalah 97%.

Kata kunci**: **Pejalan Kaki, Lampu Penerangan Jalan, Mikrokontroler, TALiSMaN,* **nRF24L01+

DEVELOPMENT OF TALISMAN BASED ADAPTIVE STREET LIGHTING SYSTEM IN MULTI-PEDESTRIAN DETECTOR CASE STUDY

Student Name	: Muhammad Iqbal Tanjung
NRP	: 5112 100 069
Major	: Teknik Informatika FTIf-ITS
Advisor I	: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
Advisor II	: Ir. Muchammad Husni, M.Kom.

ABSTRACT

Present public street lighting will remain lit 100% although no pedestrian or public transport passed through the system can lead to inefficient use of electrical power and the shorten lifespan of the lamp. TALiSMaN method (Traffic-Aware Lighting Scheme Management Network) is a method to adjust the brightness of street lights based on the presence of an object passing through street lighting system. Based on the study of how the needs of road users comfort and safety when passing the street lights illuminated road. Set of street lights will be lit only there is human that passes through one of the lamps node.

In previous studies data transmission between nodes done using GSM network with one main master to adjust the lighting levels of the system. In this study, the author wanted to create an adaptive street lighting systems which can adapt when more than one node detects pedestrians. Microcontroller required at each node, consist with a radio module and sensor. Data delivery between nodes handled by nRF24l01+ module with multi-hop data transmission. To detect human presence, PIR sensor used, so each node can detect the presence of pedestrians that pass through the system. The lighting system can be altered if there is a pedestrian

is detected at a node. Nodes that detect pedestrians Sn lit 100%, Sn-1 and node Sn+1 lit 67%, and the node Sn-2 and Sn+2 lit 30%.

The system will be tested, both in functionality and performance using the scenarios that have been determined. Based on trial results, the functionality of the system is running well and accuracy of shipment and delivery data muti-hop by 75%, single hop two directions by 90% as well as a single hop in one direction with a distance of 10m is 98%, 20m is 98% and 30m is 97%.

Keywords**: **Pedestrian, Street Light, Microcontroller, TALiSMaN,* nRF24L01+

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji dan syukur, kehadirat Allah Subhanahu wa ta'ala yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Pengembangan Sistem Penerangan Lampu Jalan Adaptif Berdasarkan Metode TALiSMaN Untuk Studi Kasus Multi-Pedestrian Detector".

Pengerjaan Tugas Akhir ini adalah momen bagi penulis untuk mengeluarkan seluruh kemampuan, hasrat, dan keinginan yang terpendam di dalam hati mulai dari masuk kuliah hingga lulus sekarang ini, lebih tepatnya di jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak. Melalui lembar ini, penulis ingin secara khusus menyampaikan ucapan terima kasih kepada:

1. Allah SWT yang telah melimpahkan rahmat, hidayah, dan inayah-Nya sehingga penulis mampu menyelesaikan Tugas Akhir dengan baik.
2. Junjungan kita Nabi Muhammad SAW yang telah menjadi inspirasi, contoh yang baik bagi penulis sehingga tetap termotivasi dalam mengerjakan Tugas Akhir.
3. Kedua orang tua penulis yang telah mencurahkan kasih sayang, perhatian, dan doa kepada penulis.
4. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku dosen pembimbing pertama, yang telah memberikan kepercayaan, dukungan, bimbingan, nasehat, perhatian, serta semua yang telah diberikan kepada penulis.

5. Bapak Ir. Muchammad Husni, M.Kom. selaku dosen pembimbing kedua, atas bimbingan, arahan, bantuan serta ide untuk menyelesaikan Tugas Akhir ini.
6. Bapak Dr.Eng Darlis Herumurti, S.Kom.,M.Kom. selaku ketua jurusan Teknik Informatika ITS, Bapak Dwi Sunaryono, S.Kom.,M.Kom. selaku dosen wali penulis, Bapak Dr. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator TA dan koordinator KP dan segenap Bapak/Ibu dosen Teknik Informatika yang telah memberikan ilmunya kepada penulis.
7. Seluruh teman Teknik Informatika ITS angkatan 2012 yang telah menemani dan memberi pengalaman berharga bagi penulis sejak maba sampai lulus.
8. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu terselesaikannya Tugas Akhir ini, penulis mengucapkan terima kasih.

Penulis telah berusaha sebaik mungkin dalam menyusun Tugas Akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, 21 Juni 2016

Penulis

Muhammad Iqbal Tanjung

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dengan jumlah instalasi lampu penerangan jalan yang mendekati 90 juta instalasi di seluruh dunia, lampu penerangan jalan merupakan alat yang dibutuhkan di mana-mana, terutama pada daerah urban[1]. Pencahayaan jalan yang efektif dapat menurunkan tingkat kecelakaan serta menurunkan tingkat kriminalitas. Namun penerangan jalan juga menjadi masalah tersendiri untuk pemerintah lokal baik secara finansial maupun dampaknya terhadap lingkungan. Pemerintah Prancis melaporkan bahwa lampu penerangan jalan mengkonsumsi daya listrik sejumlah 119 TW dan melepaskan emisi sebesar 69 ton CO_2 ke atmosfer [1]. Dengan tingginya tingkat urbanisasi, dan masih banyaknya jalan yang memiliki lampu penerangan jalan, beban finansial dan enviromental diprediksi akan terus meningkat seiring dengan peningkatan jumlah lampu listrik yang mencapai 300% pada sepuluh tahun mendatang[2].

Pada lampu penerangan jalan konvensional, biasanya lampu akan mulai menyala pada malam hari. Permulaan lampu untuk menyala dan padam biasanya dipicu oleh sebuah jam yang sudah diatur jadwalnya atau menggunakan sensor cahaya. Namun lampu penerangan jalan yang terus menyala pada jam operasional dapat menyebabkan pemborosan energi. Contohnya saat tengah malam di mana sedikit manusia yang melintas, maka jalan tidak membutuhkan penerangan dengan tingkat kecerahan maksimal. Beberapa teknologi lampu penerangan jalan yang menggunakan *time based dimming scheme* mampu mengurangi penggunaan daya yang signifikan dengan mengatur keredupan lampu sesuai dengan jadwal yang ditentukan. Peredupan lampu pada saat tertentu dapat mengurangi keamanan saat melewati jalan dan rasa aman dari tindak kriminalitas.

Sudah ada beberapa penelitian yang mengembangkan sistem penerangan jalan yang adaptif terhadap manusia yang melewati sebuah jalan. Beberapa penelitian menggunakan sistem kendali lampu yang terpusat pada sebuah *control center*. Penelitian tersebut

menggunakan jaringan internet untuk mengirimkan data dari user kepada *control center* dan menggunakan GPS pada *smartphone* sebagai *tracking device* lokasi pejalan kaki [3]. Namun penerapan seperti itu hanya berlaku pada pejalan kaki yang memiliki perangkat *smartphone* dan harus terhubung dengan jaringan Internet. Penggunaan jaringan internet juga memiliki *delay* yang tinggi, tentunya ini tidak sesuai dengan maksud dari lampu jalan itu sendiri, yaitu selalu tersedia walaupun tanpa usaha dari pejalan kaki. Karena kekurangan tersebut metode TALiSMaN diadaptasi, sebuah algoritma distribusi manajemen skema jaringan untuk penerangan jalan yang *aware* terhadap situasi jalan [4]. Pada metode tersebut, digunakan algoritma peredupan lampu sesuai dengan pejalan kaki yang melewati sebuah area sensor. Tidak lagi menggunakan jaringan yang terpusat pada sebuah *control centre* metode ini didesain untuk beroperasi *autonomously* dengan *short-range mesh network*. Tetapi pada metode TALiSMaN itu sendiri masih terdapat beberapa kekurangan, yaitu tidak mampu menangani keadaan jalan saat ada lebih dari 1 pejalan kaki yang melewati beberapa *node* lampu penerangan jalan. Tugas akhir ini bertujuan untuk merancang sistem lampu penerangan jalan yang mampu menangani lebih dari satu pejalan kaki dengan mengadaptasi metode TALiSMaN.

Sistem ini diharapkan mampu mendeteksi pejalan kaki dan mengatur kondisi lampu penerangan jalan dengan waktu kurang dari 1 detik. Karena ada beberapa faktor yang mempengaruhi proses pengondisian lampu jalan, pengiriman *command* antar *node* lampu penerangan jalan, jumlah pejalan kaki dan kemampuan sensor mendeteksi pejalan kaki.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana cara mengondisikan tingkat pencahayaan lampu sistem apabila ada satu *node* lampu mendeteksi pejalan kaki?

2. Bagaimana cara mengondisikan tingkat pencahayaan lampu sistem apabila ada lebih dari satu *node* lampu mendeteksi pejalan kaki?
3. Bagaimana pengaturan *node* lampu apabila ada pejalan kaki yang melewati area *sensing* sensor?
4. Bagaimana pengaturan *node* lampu apabila pejalan kaki berada di luar area *sensing* sensor namun masih berada di antara dua *node* lampu?
5. Bagaimana tingkat ke-akurasian sistem yang dibangun?

1.3 Batasan Masalah

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

1. Sistem ini menggunakan Arduino sebagai mikrokontroler.
2. Aktuator lampu jalan disimulasikan menggunakan rangkaian lampu LED.
3. Menggunakan bahasa pemrograman C++ dengan *library* Arduino.
4. Aplikasi dibangun dengan menggunakan Arduino IDE.
5. Pengujian dilakukan dengan prototype berupa 4 buah Arduino dan lampu, dimana 4 orang akan melewatinya dengan jarak antar Arduino 1 meter.
6. Sensor yang digunakan untuk mendeteksi kehadiran manusia adalah sensor *passive infrared.*
7. Sistem mendeteksi pejalan kaki berdasarkan sensor *passive infrared* yang dipasang pada tiap lampu jalan
8. Objek yang dideteksi adalah manusia/pejalan kaki.
9. Tiap sensor pada *node* lampu hanya bisa mendeteksi keberadaan pejalan kaki bukan jumlah pejalan kaki.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini antara lain:

1. Aplikasi bertujuan untuk menangkap *input* nilai dari sensor *passive infrared* sebagai pesan dari pengiriman antar *node.*

2. Aplikasi bertujuan untuk menangkap *input* nilai dari sensor *passive infrared* sebagai *output* yang dipresentasikan oleh lampu.
3. Aplikasi bertujuan untuk mengatur tingkat pencahayaan lampu pada tiap *node* dalam jaringan lampu penerangan jalan raya.
4. Aplikasi bertujuan untuk mengolah *trigger* dari pejalan kaki yang melewati sistem.

1.5 Manfaat

Manfaat dari hasil pembuatan Tugas Akhir ini adalah :

1. Sistem yang dibuat pada Tugas Akhir ini diharapkan dapat menghemat penggunaan listrik dari lampu penerangan jalan.
2. Mengatur komunikasi antar lampu jalan secara nirkabel.
3. Memberikan solusi suatu sistem yang adaptif terhadap jumlah manusia yang melewati jalan.

1.6 Metodologi

Adapun langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal tugas akhir

Tahap awal untuk memulai pengerjaan TA adalah penyusunan proposal Tugas Akhir. Pada proposal tersebut dijelaskan secara garis besar tentang metode TALiSMaN, mekanisme sistem penerangan lampu yang berjalan secara adaptif dan mekanisme pengiriman data.

2. Studi literatur

Tahap ini merupakan tahap pengumpulan dan pemahaman informasi yang diperlukan untuk pengerjaan tugas akhir. Mulai dari pengumpulan literatur, diskusi, serta pemahaman topik tugas akhir di antaranya tentang:

a. Perancangan perangkat keras yang akan diintegrasikan dengan nRF24L01+ untuk berkomunikasi antar perangkat.

b. Penggunaan dari *library* yang digunakan untuk nRF24L01+ untuk berkomunikasi antar perangkat

3. Perancangan sistem

Tahap ini merupakan perancangan sistem dengan menggunakan studi literatur dan mempelajari konsep aplikasi yang akan dibuat. Dengan berbekal teori, metode dan informasi yang sudah terkumpul pada tahap sebelumnya diharapkan dapat membantu dalam proses perancangan sistem.

4. Implementasi perangkat lunak dan perangkat keras

Tahap ini merupakan implementasi rancangan sistem yang telah dibuat. Tahap ini merealisasikan apa yang terdapat pada tahapan perancangan yang telah dibuat sebelumnya sehingga menjadi sebuah 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 perencanaan. 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 tahap ini disusun laporan tugas akhir sebagai dokumentasi pelaksanaan tugas akhir, yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini terdiri dari beberapa bab, yang dijelaskan sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisi latar belakang, permasalahan, tujuan, batasan permasalahan, metodologi, dan sistematika penulisan.

BAB II. TINJAUAN PUSTAKA
Bab ini berisi dasar teori yang mendukung pembahasan tugas akhir ini. Dasar teori yang dibahas meliputi perangkat mikrokontroler, *networking* dan sensor, beserta metode TALiSMaN.

BAB III. ANALISIS PERANCANGAN
Bab ini berisi tentang perancangan sistem, *flowchart,* dan perancangan perangkat keras yang akan dibuat. Perancangan yang dibahas meliputi perancangan sifat adaptif sistem, implementasi metode TALiSMaN, dan protokol komunikasi antar perangkat.

BAB IV. IMPLEMENTASI
Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *pseudocode* dari aplikasi yang diimplementasikan pada perangkat keras.

BAB V. EVALUASI DAN UJI COBA
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. Skenario yang dievaluasi adalah sifat adaptif dari sistem. Pengujian performa merupakan pengujian sifat adaptif aplikasi terhadap beberapa pembanding aplikasi yang lain.

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

2.1 NRF24L01+

nRF24L01+ seperti yang ditunjukkan pada Gambar2.1 adalah sebuah *chip transciever* 2.4Ghz dengan *baseband protocol engine* (Enhanced ShockBurst™) yang tertanam di dalamnya, sesuai dengan implementasi pada sistem yang membutuhkan daya yang rendah. nRF24L01+ dirancang untuk standart operasi pada range pita frekuensi ISM 2.400 - 2.4835GHz. Perancangan sebuah sistem dengan menggunakan nRF24L01+ cukup menggunakan sebuah mikrokontroler dan beberapa komponen pasif. nRF24L01+ dapat dioperasikan dan dikonfigurasi melalui *port Serial Peripheral Interface* (SPI). *Register map* yang dapat diakses melalui port SPI, berisi semua konfigurasi *registeri* pada nRF24L01+ dan dapat diakses pada semua mode operasi pada *chip*[10].

Protokol Enhanced ShockBurst™ adalah protokol yang berdasarkan pada *packet communicaion* dan mendukung berbagai mode dari operasi manual hingga protokol otonom yang kompleks. Adanya internal FIFO menjamin aliran data antara *radio front end* dan sistem mikrokontroler. Enhanced ShockBurst™ mengurangi *system cost* dengan mengelola semua *link layer operation* kecepatan tinggi[10].

nRF24L01+ adalah modul *radio transciever* yang hemat daya daripada ZigBee dan memiliki harga yang lebih murah. Jangkauan dari nRF24L01+ sebesar ±1000 meter sedangkan ZigBee ± 50 meter. Namun *node* pada ZigBee dikonfigurasi menggunakan AT *command*, atau aplikasi windows yang terpisah. Node pada nRF24L01+ dikonfigurasi dengan meng-*compile firmware* langsung pada EEPROM[10].

Fitur yang dimiliki oleh modul nRF24L01+ adalah sebagai berikut:

1. Radio
   - Operasi pita 2.4GHz ISM

- 126 RF channel
- Antarmuka RX dan TX umum .
- Modulasi GFSK
- Kecepatan data 250kbps 1 and 2Mbps
- 1MHz *non-overlapping* 9 *el spacing* pada 1Mbps
- 2MHz *non-overlapping channel spacing* pada 2Mbps

2. Transmitter
   - Output daya yang dapat diprogram: 0, -6, -12 or -18dBm
   - Output power 11.3mA pada 0dBm.
3. Receiver
   - Fast AGC untuk meningkatkan *dynamic range*
   - Filter *channel* yang terintegrasi
   - 13.5mA pada 2Mbps
   - Sensivitas -82dBm pada 2Mbps
   - Sensivitas -85dBm pada 1Mbps
   - Sensivitas -94dBm pada 250kbps
4. RF Synthesizer
   - *Synthesizer* yang terintegerasi secara penuh
   - Tanpa *loop filer* eksternal, VCO varactor diode atau resonator
   - Menerima *low cost* ±60ppm kristal 16MHz
5. Enhanced ShockBurst™
   - Panjang payload dinamis adalah 1 to 32 bytes
   - Penanganan paket secara otomatis
   - Penanganan transaksi paket secara otomatis
   - 6 *pipe* data MultiCeiver™ untuk 1:6 jaringan star
6. Manajemen Daya
   - Regulator tengangan terintegerasi
   - 1.9 to 3.6V rentang tegangan sumber
   - *Idle modes* dengan *start-up* cepat untuk manajemen daya tingkat lanjut
   - Mode standby-I 6μA, mode *power down*900nA
   - 1.5ms waktu *start-up* maksimal dari mode power down
   - 130us waktu *start-up* dari mode standby-I
7. Antarmuka *Host*

- 4-pin hardware SPI
- 10Mbps maksimal
- 3 TX dan RX FIFOs 32 bytes yang terpisah

Gambar 2.1 Perangkat nRF24L01+

2.2 Lampu Penerangan Jalan

Lampu jalan atau dikenal juga sebagai Penerangan Jalan Umum (PJU) adalah lampu yang digunakan untuk penerangan jalan dimalam hari sehingga mempermudah pejalan kaki, pesepeda dan pengendara kendaraan dapat melihat dengan lebih jelas jalan/medan yang akan dilalui pada malam hari, sehingga dapat meningkatkan keselamatan lalu lintas dan keamanan dari para pejalan kaki dari kegiatan/aksi kriminal. Clarke mengatakan penerangan (jalan) yang lebih baik akan menghalangi penyerang yang mengambil manfaat dari kegelapan malam[6]. Pada sistem yang dibuat pada tugas akhir ini tipe lampu yang digunakan adalah lampu penerangan jalan umum *one side single arm* yang ditunjukkan pada Gambar 2.2.

Gambar 2.2 Lampu Penerangan Jalan Umum[15]

Penerangan jalan di kawasan perkotaan mempunyai fungsi antara lain :

1. Menghasilkan kekontrasan antara obyek dan permukaan jalan;
2. Sebagai alat bantu navigasi pejalan kaki;
3. Meningkatkan keselamatan dan kenyamanan pejalan kaki, khususnya pada malam hari;
4. Mendukung keamanan lingkungan;
5. Memberikan keindahan lingkungan jalan

Perencanaan penerangan jalan terkait dengan kriteria sebagai berikut ini:

1. Volume lalu-lintas, baik kendaraan maupun lingkungan yang bersinggungan seperti pejalan kaki, pengayuh sepeda, dll;
2. Tipikal potongan melintang jalan, situasi (*lay out*) jalan dan persimpangan jalan;
3. Geometri jalan, seperti alinyemen horisontal, alinyemen vertikal, dll;
4. Tekstur perkerasan dan jenis perkerasan yang mempengaruhi pantulan cahaya lampu penerangan;
5. Pemilihan jenis dan kualitas sumber cahaya/lampu, data fotometrik lampu dan lokasi sumber listrik;
6. Tingkat kebutuhan, biaya operasi, biaya pemeliharaan, dan lain-lain, agar perencanaan sistem lampu penerangan efektif dan ekonomis;

7. Rencana jangka panjang pengembangan jalan dan pengembangan daerah sekitarnya;
8. Data kecelakaan dan kerawanan di lokasi.

Beberapa tempat yang memerlukan perhatian khusus dalam perencanaan penerangan jalan antara lain sebagai berikut :

1. Lebar ruang milik jalan yang bervariasi dalam satu ruas jalan;
2. Tempat-tempat di mana kondisi lengkung horizontal (tikungan) tajam;
3. Tempat yang luas seperti persimpangan, interchange, tempat parkir, dan lain sebagainya;
4. Jalan-jalan berpohon;
5. Jalan-jalan dengan lebar median yang sempit, terutama untuk pemasangan lampu di bagian median;
6. Jembatan sempit/panjang, jalan layang dan jalan bawah tanah (terowongan);
7. Tempat-tempat lain di mana lingkungan jalan banyak berinterferensi dengan jalannya.

Tabel 2.1 Rumah Lampu Tipe A

Jenis lampu	Tinggi lampu (m)	Lebar jalan (m)								Tingkat pencahayaan
		4	5	6	7	8	9	10	11	
35W SOX	4	32	32	32	-	-	-	-	-	3,5 LUX
	5	35	35	35	35	35	34	32	-	
	6	42	40	38	36	33	31	30	29	
55W SOX	6	42	40	38	36	33	32	30	29	6,0 LUX
90W SOX	8	60	60	58	55	52	50	48	46	
90W SOX	8	36	35	35	33	31	30	29	28	10,0 LUX
135W SOX	10	46	45	45	44	43	41	40	39	
135W SOX	10	-	-	25	24	23	22	21	20	20,0 LUX
180W SOX	10	-	-	37	36	35	33	32	31	
180W SOX	10	-	-	-	-	22	21	20	20	30,0 LUX

Tabel 2.2 Rumah Lampu Tipe A

Jenis lampu	Tinggi lampu (m)	Lebar jalan (m)								Tingkat pencahayaan
		4	5	6	7	8	9	10	11	
50W SON / 80W MBF/U	4	31	30	29	28	26	-	-	-	3,5 LUX
	5	33	32	32	31	30	29	28	27	
70W SON / 80W MBF/U	6	48	47	46	44	43	41	39	37	6,0 LUX
70W SON / 80W MBF/U	6	34	33	32	31	30	28	26	24	
100W SON	6	48	47	45	42	40	38	36	34	10,0 LUX
150W SON / 250W MBF/U	8	-	-	48	47	45	43	41	39	
100W SON	6	-	-	28	26	23	-	-	-	
250W SON / 400W MBFU	10	-	-	-	-	55	53	50	47	
250W SON / 400W MBFU	10	-	-	36	35	33	32	30	28	20,0 LUX
400W SON	12	-	-	-	-	39	38	37	36	30 LUX

Untuk jarak antar lampu penerangan jalan disesuaikan berdasarkan tingkat pencahayaan, tinggi lampu, rumah lampu dan lebar dari jalan yang akan diterangi oleh lampu. Pada Tabel 2.1 menunjukkan jarak antar lampu yang ditetapkan oleh Badan Standarisasi Nasional SNI 7391:2008 untuk rumah lampu tipe A dan Tabel 2.2 untuk rumah lampu tipe B. Rumah lampu tipe A memiliki penyebaran sorotan cahaya lebih luas karena lampunya berjenis gas sodium bertekanan rendah, sedangkan lampu B penyebaran sorotan

cahaya lebih ringan berjenis lampu merkuri atau sodium bertekanan tinggi[13].

2.3 Mikrokontroler Arduino

Mikrokontroler atau *embedded controller* adalah suatu sistem yang mengandung masukan/keluaran, memori dan prosesor yang digunakan pada sistem yang sederhana seperti pendingin udara, mesin cuci dan lain sebagainya. Pada prinsipnya mikrokontroler adalah sebuah komputer berukuran kecil yang dapat digunakan untuk mengambil keputusan dalam proses perulangan dengan berbagai jenis alat masukan[11].

Arduino adalah sebuah board yang berisi mikrokontroler yang bersifat *open-source*. Arduino dirancang untuk memudahkan prototyping sebuah sistem untuk tujuan penelitian ataupun pre-produksi dari sebuah produk yang tertanam mikrokontroler di dalamnya. Mikrokontroler yang digunakan pada board Arduino berjenis AVR dari perusahaan Atmel. Arduino tidak membutuhkan flash programmer external karena di dalam chip mikrokontroler Arduino telah diisi dengan bootloader yang membuat proses upload menjadi lebih sederhana. Dalam sebuah mikrokontroler Arduino dapat pula ditanamkan berbagai macam *library* maupun metode selama kapasitas memori dari sebuah mikrokontroler mencukupi. Arduino dapat dipasangkan dengan berbagai sensor, *network pheriperal*, modul radio, layar LCD dan bahkan keyboard sebagai alat masukan dan keluaran[11].

2.4 Arduino Uno Revision 3

Arduino Uno merupakan mikrokontroler yang paling umum digunakan dalam keluarga papan mikrokontroler Arduino. Arduino Uno menggunakan prosesor ATmega328. Perbedaan antara Arduino Uno dengan Arduino Uno Rev 3 adalah pada mikrokontroler kedua yang menangani protokol komunikasi USB. Pada Arduino Uno masih menggunakan chip ATmega8U2 sedangkan Arduino Uno Rev 3 telah menggunakan chip ATmega16U2. Arduino memiliki 14 *pin* (termasuk 6 *pin* untuk *PWM output*), 6 *analog input*, sebuah koneksi

USB, sebuah *power jack*, ISCP *header*, dan tombol *reset*. Arduino dihubungkan dengan kabel USB atau menggunakan AC-to-DC *adapter* (baterai) untuk awalan menggunakan Arduino. Program diunggah menggunakan kabel USB. Arduino Uno dapat dioperasikan dengan *power* 5V (*vcc*). Kapasitas memori Arduino Uno adalah 32 kb dan setengah dari kapasitas memori digunakan untuk *bootloader*. Gambar 2.3 menunjukkan penjelasan singkat komponen yang ada pada Arduino Uno Revision3[11].

1. USB *Connector*

   USB *connector* adalah *port* Arduino Uno yang penghubung antara PC dengan mikrokontroler Arduino Uno dengan sebuah kabel USB A-B. USB *connector* berfungsi untuk mengunggah *sketch* program dari PC ke dalam mikrokontroler serta sebagai sumber daya dari Arduino Uno[9].
2. Pin I/O Digital

   Pin I/O Digital adalah pin yang digunakan untuk menerima atau mengirim isyarat digital. Isyarat biner 1 atau HIGH direpresentasikan dalam bentuk tegangan tertentu dan isyarat biner 0 atau LOW direpresentasikan dalam bentuk tegangan 0V. Beberapa pin digital yang dapat digunakan sebagai keluaran analog, dinamakan pin *PWM* ditandai dengan simbol "~". Pin *PWM* tersebut adalah pin nomor 2, 5, 6, 9, 10 dan 11[11].
3. *VCC Pin*

   Pin yang memiliki fungsi untuk memberikan daya dengan tegangan tertentu (3,3V dan 5V) pada modul yang terpasang pada Arduino Uno[11].
4. *Ground (GND) Pin*

   Pin yang memiliki fungsi untuk menghubungkan *pin ground* (*GND*) dari komponen (seperti rangkaian sensor) ke mikrokontroler Arduino[11].
5. *Analog Input Pins*

   Pin yang digunakan untuk menerima nilai analog.
6. *Reset Button*

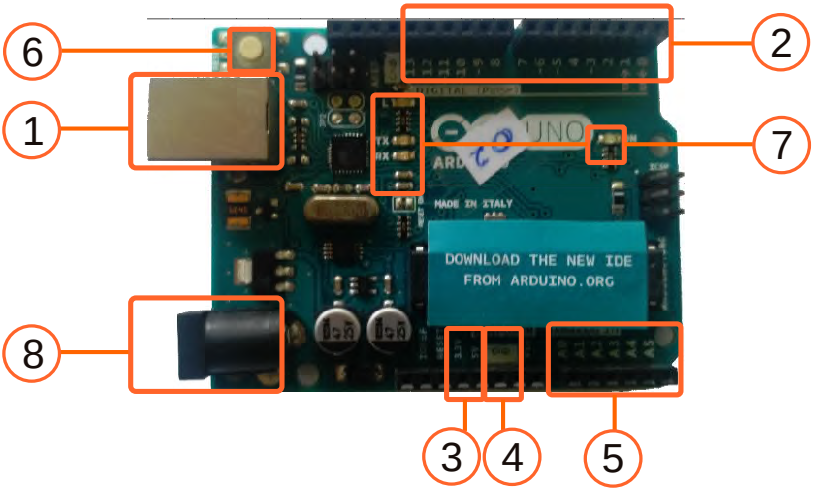
Tombol reset akan membuat *sketch* yang telah diunggah dijalankan ulang.

7. LED

Empat lampu LED yang terdiri dari ON menandakan Arduino terhubung pada sumber daya, RX TX menandakan proses pengiriman dan penerimaan pesan berjalan, dan indikator upload *script* pada Arduino.

8. *Power Adaptor*

Power adaptor berfungsi sebagai penghubung ke sumber tegangan eksternal apabila Arduino tidak terhubung dengan PC. Adaptor AC-DC atau baterai dengan tegangan antara 5V hingga 12 V[11].



Gambar 2.3 Arduino Uno R3

2.5 Sensor PIR

Sensor *Passive Infrared* atau disingkat PIR adalah sensor yang mendeteksi adanya perubahan gerakan manusia atau objek yang memiliki panas dalam jangkauannya[12]. PIR memilki ukuran yang

kecil, dengan harga yang murah, hemat daya, mudah digunakan, serta tidak mudah rusak. PIR banyak digunakan dalam perlengkapan sehari-hari, seperti pendingin udara.

Pada dasarnya PIR terbuat dari *pyroelectric sensor* yang dapat mendeteksi level dari radiasi inframerah. Semua benda memancarkan readiasi pada level yang rendah, dan dengan semakin panas suatu benda maka radiasi yang dipancarkan juga semakin besar. Sensor PIR memiliki dua slot di dalamnya, setiap slot dibuat dari material khusus yang sensitif terhadap inframerah[13]. Ketika sensor pada kondisi *idle,* kedua slot mendeteksi kadar radiasi inframerah yang sama, radiasi yang dipancarkan dari lingkungan sekelilingnya. Ketika sebuah objek hangat, seperti tubuh manusia atau hewan melewatinya, objek tersebut melewati salah sati slot dari sensor PIR, yang menyebabkan sebuah perubahan differensial positif diantara slot satu dan slot yang lain. Ketika objek keluar dari area sensor, hal sebaliknya terjadi, dimana sensor mengeluarkan sebuah perubahan diferensial negatif. Gambar 2.4 menunjukkan bentuk dari sensor PIR HC-SR501[13].

Gambar 2.4 Sensor PIR HC-SR501

2.6 Metode TALiSMaN

Metode TALiSMaN (***T**raffic-**A**ware **L**ighting **S**cheme **M**anagement **N**etwork*) adalah metode pengaturan kecerahan pada lampu jalan berdasarkan ada tidaknya objek yang melewati lampu penerangan jalan raya[4]. Metode ini dibuat berdasarkan penelitian

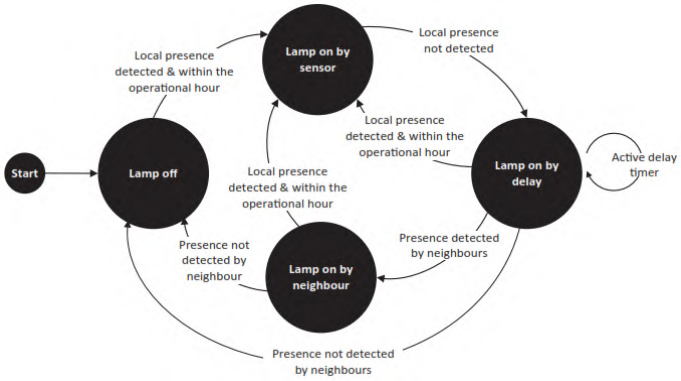
bagaimana kebutuhan dari pengguna jalan akan kenyamanan dan keamanan ketika melewati jalan yang diterangi oleh lampu penerangan jalan. Set dari beberapa lampu penerangan jalan akan menyala hanya ketika ada manusia yang melewati salah satu lampu. Tujuan utamanya adalah untuk efisiensi daya pada lampu penerangan jalan.

Berdasarkan perspektif pejalan kaki, panjang jalan yang dibutuhkan untuk diterangi sejauh 150 m dan dalam rentang jalan tersebut dibutuhkan pola penerangan yang berbeda. TALiSMaN memanfaatkan hal tersebut dengan mengatur kecerahan lampu sesuai dengan kebutuhan pengguna jalan dan meningkatkan efisiensi daya dengan mengurangi penggunaan listrik. TALiSMaN mendeteksi pengguna jalan dan membagikan informasi tersebut pada lampu penerangan jalan terdekat[4].

Alih-alih bergantung pada sebuah kontrol kendali yang terpusat untuk mengelola operasi lampu penerangan jalan, TALiSMaN menggunakan WSN otonom. Pada setiap lampu memiliki sebuah *node* sensor nirkabel dengan sebuah modul komunikasi nirkabel, sensor pendeteksi pengguna jalan, dan pengontrol lampu di dalamnya. Hal tersebut memungkinkan terbentuknya *multi-hop* WSN antar lampu peneranan jalan untuk pertukaran informasi. Jaringan lampu tersebut *time-syncronised*, dan tiap lamu telah terprogram dengan informasi lokasi dan id masing-masing. Kontroler lampu akam mengatur kecerahan masing-masing lampu berdasarkan pengguna jalan yang terdeteksi oleh sensor[4].

Default lampu penerangan jalan adalah padam (kecerahan 0%), ketika seorang pejalan kaki berada pada daerah *sensing* sensor maka lampu akan menyala (*Lamp on by sensor*) dari keadaan atau *state* padam(*Lamp off*). Jika pejalan kaki berada di luar daerah *sensing* lampu akan menyala dengan *delay* tertentu hingga pejalan kaki berada pada area *sensing* lampu awal atau lampu sebelahnya (*Lamp on by neighbour*) jika telah melewati waktu *delay* maka lampu akan mati kembali (*Lamp off*). Dibawah ini akan disertakan gambar ilustrasi bagaimana sistem mengatasi jika ada satu orang yang berjalan melalui

sistem. *State chart* dari sistem penerangan lampu jalan raya untuk tiap pejalan kaki yang melewati sistem ditunjukkan pada Gambar 2.5.



Gambar 2.5 State Chart Sistem Penerangan Lampu TALiSMaN[4]

Pada Gambar 2.77 dan Gambar 2.6 menunjukkan perkembangan pola penerangan yang dibutuhkan setelah pejalan kaki terdeteksi oleh sensor pada lampu s_2. Setelah informasi keberadaan pejalan kaki dibagikan kepada seluruh *node* sensor pada lampu s_1 hingga lampu s_8, pencahayaan pada masing-masing lampu, L_{ped} secara progresif termodulasi berdasar algoritma berikut:

```
if operation state is 'Lamp off' then
      Lped = 0
if operation state is 'Lamp on by sensor' or
'by neighbour' then
      Lped = 1 - 0.2 Zped(daprox)
if operation state is 'Lamp on by delay' then
      current Lped(daprox) remains
```

Gambar 2.6 Algoritma Modulasi Tiap Lampu[4]

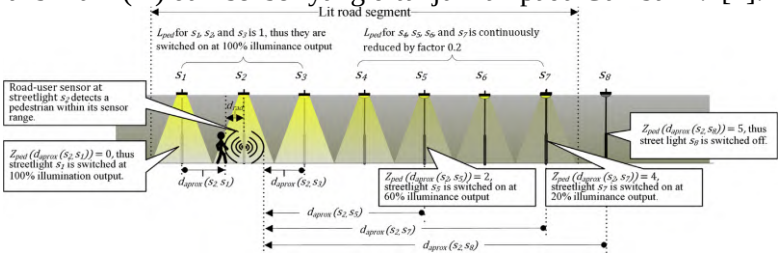
Berdasarkan Gambar 2.6, dimana L_{ped} adalah pencahayaan yang dibutuhkan pada lampu penerangan jalan berdasarkan jarak relatif (m) terhadap pejalan kaki yang terdeteksi, dan d_{approx} dan Z_{ped} menentukan zona pencahayaan menurut d_{approx}[4].

Lima zona pencahayaan dibuat untuk pejalan kaki, dan pencahayaan (L_{ped}) yang dibutuhkan pada tiap zona dikurangi 20%. Karena jarak antar zona pencahayaan adalah 30 meter dan pejalan kaki membutuhkan penerangan dengan rentang jarak 150m, maka nilai maksimum dari Z_{ped} adalah lima. Zona pencahayaan dari lampu penerangan jalan, Z_{ped} berkenaan dengan d_{approx} adalah sebagai berikut:

$$Z_{ped}(d_{approx}) = \begin{cases} 0, & \left\lfloor \frac{d_{approx}}{30} \right\rfloor = 0 \\ \left\lfloor \frac{d_{approx}}{30} \right\rfloor, & 0 < \left\lfloor \frac{d_{approx}}{30} \right\rfloor \leq 5 \\ 5, & \left\lfloor \frac{d_{approx}}{30} \right\rfloor > 5 \end{cases} \quad (2.1)$$

$$d_{approx} = \begin{cases} 0, & d_{rad} \geq d_{det} \\ d_{det} - d_{rad}, & d_{rad} < d_{det} \end{cases} \quad (2.2)$$

Dimana d_{det} adalah jarak *euclidian* (m) ke sensor terdekat yang mendeteksi pejalan kaki dan d_{rad} adalah jarak deteksi maksimum (m) dari sensor yang ditunjukkan pada Gambar 2.7[4].



Gambar 2.7 Jarak Euclidian ke Sensor[4]

Daerah kosong antara jangkauan sensor pada tiap lampu dapat menimbulkan perubahan yang tidak dibutuhkan pada pencahayaan lampu jalan. Untuk mengurangi hal tersebut, *state delay counter* diterapkan untuk memperpanjang state operasi TALiSMaN pada state '*Lamp on by delay*' sampai pejalan kaki benar-benar telah melewati daerah kosong tersebut. Fitur ini juga meringankan latensi komunikasi jaringan, saat sensor sedang mendeteksi keberadaan

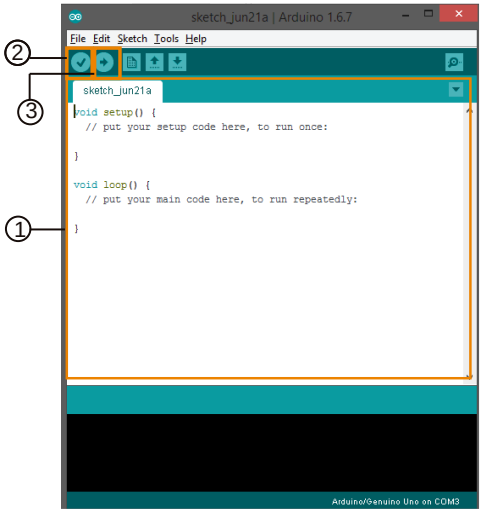
pejalan kaki secara terus menerus dan pada saat yang bersamaan informasi disebarkan kepada *node* lainnya. Untuk menentukan waktu kadaluarsa *state delay counter*, t_{exp} ditentukan dengan persamaan berikut,

$$t_{exp} = \frac{d_{adj} - 2\, d_{rad}}{v}\ ,\ d_{adj} \geq d_{rad} \qquad (2.3)$$

dimana d_{adj} adalah jarak terjauh (m) antara lampu yang bersebelahan, d_{rad} adalah jangkauan dari sensor dan v adalah ekspektasi kelajuan terlambat (m/s) dari pengguna jalan[4].

2.7 Arduino IDE

Arduino IDE yang ditunjukkan pada Gambar 2.8 merupakan software IDE *open source*, sehingga bisa memiliki *resource* untuk *developer* yang terbuka dan gratis. Bahasa pemrograman yang digunakan adalah bahasa C/C++ dengan dukungan *library* Arduino[10].



Gambar 2.8 Arduino IDE

Struktur perintah pada Arduino secara garis besar terdiri dari dua bagian, yaitu void setup dan void loop. Void setup berisi perintah

yang dieksekusi sekali sejak arduino dinyalakan. Sedangkan void loop berisi perintah yang dilakukan berulang-ulang selama Arduino dinyalakan. Aplikasi perangkat lunak Arduino IDE terdiri dari 3 bagian, yaitu:

1. Editor program
2. *Compiler*
3. *Uploader*

2.8 Jaringan Sensor Nirkabel

Secara umum, *Wireless Sensor Network* (WSN) didefinisikan sebagai salah satu jenis dari jaringan nirkabel terdistribusi, yang memanfaatkan teknologi *Embeded System* dan seperangkat *node* sensor, untuk melakukan proses sensor, pengiriman data, monitoring, dan penyajian data ke pengguna melalui komunikasi internet. Setiap jenis sensor memiliki perangkat lunak (aplikasi, sistem operasi) dan perangkat keras masing-masing yang kemudian akan digabungkan dan dijalankan ke dalam sistem *Wireless Sensor Network*[8].

BAB III
ANALISIS DAN PERANCANGAN

Analisis dan perancangan merupakan bagian penting dari pembuatan perangkat lunak dan perangkat keras yang berupa perencanaan-perencanaan secara teknis aplikasi yang dibuat, sehingga bab ini secara khusus akan menjelaskan analisis dan perancangan sistem yang dibuat dalam Tugas Akhir ini. Berawal dari deskripsi umum aplikasi hingga perancangan proses, alur dan implementasinya.

3.1 Deskripsi Umum Sistem

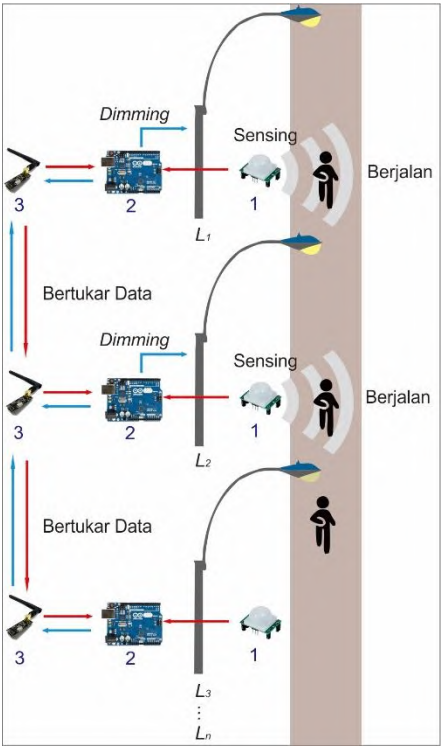
Pada Tugas Akhir ini akan dibangun suatu sistem penerangan jalan untuk pejalan kaki secara adaptif terhadap banyaknya *node* lampu yang mendeteksi pejalan kaki dengan menggunakan mikrokontroler Arduino. Sistem penerangan jalan adaptif dengan studi kasus adanya lebih dari satu *node* lampu mendeteksi pejalan kaki yang melewati sistem. Sistem akan mengatur tingkat pencahayaan seluruh lampu dalam sistem sesuai dengan keberadaan dan jumlah pejalan kaki.

Tiap *prototype* lampu penerangan terdiri dari Arduino UNO R3, modul *transciever* nRF24L01+, rangkaian lampu LED, dan sebuah sensor PIR HC-SR501. Sebagai *controller* dari sistem, Arduino mengatur masukan dari sensor PIR dan informasi dari nRF24L01+, dan keluaran berupa tingkat pencahayaan lampu dan *command* yang dikirim pada lampu lain melalui nRF24L01+. nRF24L01+ berfungsi sebagai alat komunikasi antar *node* lampu melalui jaringan frekuensi pita ISM 2.4 GHZ menggunakan library RF24Network. Sensor diletakkan di tempat yang memiliki keuntungan untuk menjangkau pejalan kaki secara maksimal, sebuah sensor PIR HC-SR501 mampu mendeteksi gerakan manusia dalam jangkauan sebesar ±6 m. Rangkaian *dimmer controller* lampu untuk mengatur tingkat pencahayaan tiap lampu diilustrasikan menggunakan tiga buah lampu LED.. Tiap *prototype*

lampu penerangan tersebut akan melakukan *monitoring* keberadaan manusia pada sekitarnya. Ketika ada gerakan manusia yang terdeteksi pada *node* lampu. *Node* lampu tersebut akan mengirim data bahwa *node* lampu mendeteksi pejalan kaki, dan tingkat pencahayaan pada masing-masing lampu lain di dalam sistem. Sehingga tiap lampu memiliki tingkat pencahayaan sesuai kebutuhan pejalan kaki.

3.2 Arsitektur Umum Sistem

Rancangan arsitektur dari sistem yang dibuat dapat dilihat pada Gambar 3.1.



Gambar 3.1 Arsitektur Umum Sistem

Berdasarkan perancangan arsitektur sistem pada Gambar 3.1, tiap *node* lampu terdiri dari sensor PIR, Arduino, *dimmer*, lampu, dan *transciever* nRF24L01+. Pengambilan data berupa sinyal digital dari sensor yang dilewati oleh pejalan kaki dilakukan oleh sensor PIR, pada jaringan sensor nirkabel berbasis nRF24L01+. Jaringan sensor nirkabel penerangan jalan ini terdiri dari minimal dua *node* lampu yang masing-masing terintegerasi dengan sensor PIR untuk mendeteksi adanya pejalan kaki, serta saling terhubung dengan koneksi jaringan nirkabel nRF24L01+. Masing-masing *node* lampu memiliki *physical address* yang tetap, sesuai dengan posisinya pada sistem. Data yang diperoleh akan diolah pada Arduino di tiap *node* secara otonom. Lalu data yang diolah pada Arduino di tiap *node* tersebut akan dikirim kepada *node* lain sebagai pendukung keputusan dalam pengaturan lama dan tingkat pencahayaan lampu.

Sifat adaptif pada sistem lampu penerangan jalan ini terdapat pada pengolahan dan pengiriman data antar *node* setelah adanya *trigger* dari sensor PIR setelah mendeteksi pejalan kaki. Pengolahan dan pengiriman data untuk mengatur tingkat pencahayaan dilakukan dengan menggunakan metode TALiSMaN. Metode TALiSMaN akan menentukan tingkat dan lama pencahayaan lampu, pada tiap *node* lampu dalam kelompok alamat yang telah ditentukan.

3.3 Perancangan Perangkat Keras Sistem

Sistem penerangan jalan ini terdiri dari dua bagian, yaitu bagian utama dan bagian *controlled light dimmer*. Pada bagian utama merupakan gabungan dari modul sensor PIR, *transciever* nRF24L01+, mikrokontroler Arduino, dan rangkaian *controlled light dimmer*. Sedangkan bagian *controlled light dimmer* adalah rangkaian komponen elektronik untuk mengatur tingkat pencahayaan lampu pada masing-masing *node*.

3.3.1 Perancangan Controlled Light Dimmer

Rangkaian *controlled light dimmer* ini adalah *output-interface* yang menggunakan bola lampu 220v sebagai

aktuatornya. Rangkaian ini mengolah masukan daya dari listrik PLN lalu mengolahnya sehingga bisa mengatur tingkat pencahayaan dari bola lampu 220V menggunakan salah satu metode *Phase Cutting*, yaitu *Leading Edge Cutting*. TRIAC melakukan *phase control*, dimana TRIAC terbuka sepenuhnya namun pada bagian tertentu pada gelombang sinus. Untuk mengurangi *error* yang tidak diinginkan pada bagian gelombang sinus saat TRIAC terbuka, dan juga level *dimming* tidak bisa diprediksi maka detektor *zero crossing* dibutuhkan. Sirkuit ini memberikan informasi saat gelombang sinus sednag melewati sumbu y nol dan juga memeberikan titik yang ditentukan pada gelombang sinus tersebut. Lampu yang dapat digunakan pada sistem ini adalah lampu halogen, filamen dan *dimming* LED. Rangkaian *controlled light dimmer* ini diilustrasikan menggunakan 3 buah lampu LED, dimana jumlah LED yang menyala menggambarkan tingkat pencahayaan dari *controlled light dimmer*.

3.3.2 Perancangan Rangkaian Utama

Pada sistem ini, perangkat keras yang digunakan sebagai berikut :

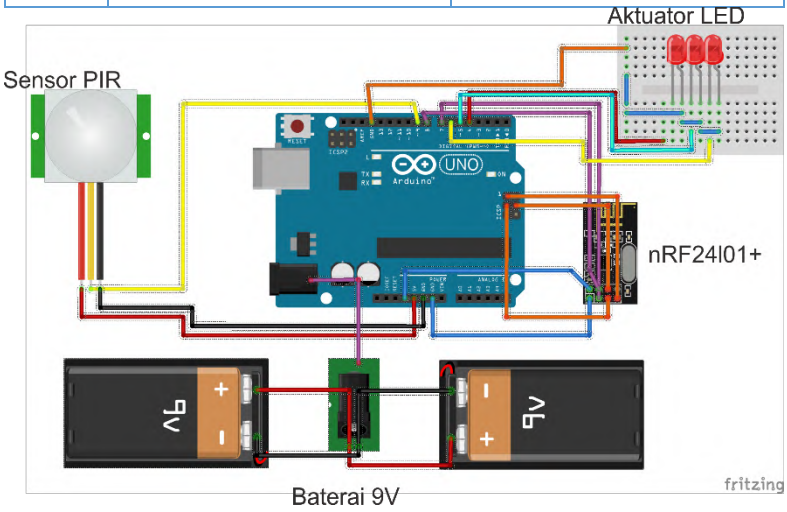
- Satu buah mikrokontroler Arduino Uno Rev 3,
- Satu buah sensor PIR
- Satu buah modul *transciever* nRF24L01+
- Rangkaian *Controlled Dimmer Lamp*
- 2 Baterai 9 V

Rangkaian perangkat sensor ditunjukkan pada Gambar 3.2. Pada perangkat utama seluruh modul digabungkan. Mikrokontroler Arduino, sensor PIR, nRF24L01+, dan *controlled dimmer light* dihubungkan. Untuk *pin* yang menghubungkan antara modul-modul yang ada dengan Arduino, dijelaskan pada Tabel 3.1.

Tabel 3.1 Penjelasan koneksi Pin Antar Arduino dan Perangkat

PIN	Modul	Deskripsi
D4	*Controlled Dimmer Light*	Sebagai aktuator LED 1
D5	*Controlled Dimmer Light*	Sebagai aktuator LED 2

D6	*Controlled Dimmer Light*	Sebagai aktuator LED 3
D7	nRF24L01+	Merubah mode TX atau RX
D8	nRF24L01+	SPI Chip Select
D9	HC SR501	Masukan dari PIR
ISCP Serial		
MISO	nRF24L01+	SPI Slave Data Output, with tri-state option
MOSI	nRF24L01+	SPI Slave Data Input
SCK	nRF24L01+	SPI Clock



Gambar 3.2 Rangkaian *Controlled Light Dimmer*

Pada rangkaian di atas, mikrokontroler berfungsi sebagai "otak" dari perangkat sensor yang akan mengontrol kerja perangkat sensor. Program diunggah di mikrokontroler ini. Program tersebut akan dijalankan sesuai dengan kebutuhan sistem ketika mengimplementasikan metode TALiSMaN.

Komponen lainnya adalah sensor PIR, modul *transciever* nRF24L01+, dan *controled dimmer light*. Sensor PIR akan mendeteksi gerakan dari manusia yang melewati sistem. Modul

transciever akan menjadi penghubung antar *node* satu dengan yang lain. *Controled dimmer light* berfungsi sebagai pengatur pencahayaan dari lampu yang ada pada tiap *node.*

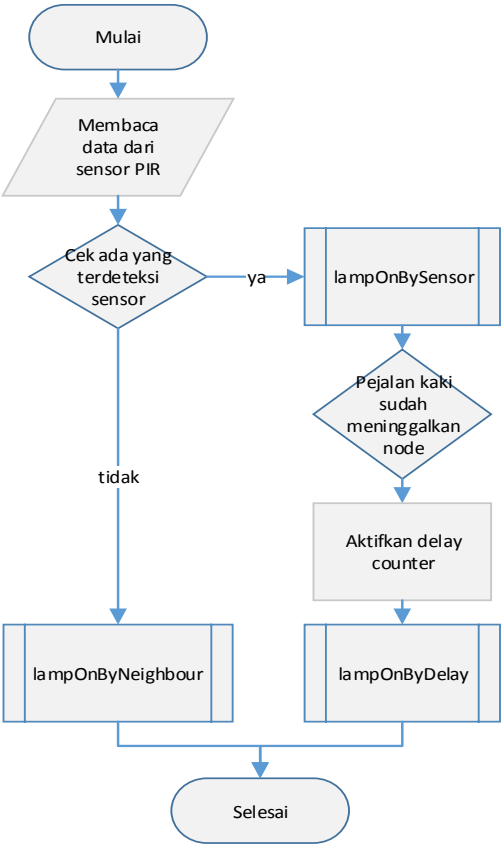
3.4 Diagram Alir Aplikasi Sistem

Pada subbab ini akan dibahas alur sistem secara keseluruhan yang melibatkan seluruh *node-node* lampu yang ada pada sistem. Perancangan alir data keseluruhan sistem dilakukan untuk dapat lebih memahami dan memberikan pandangan secara menyeluruh mengenai sistem yang dibangun, serta menunjukkan tentang fungsi-fungsi utama atau proses yang ada dan aliran logika dari jalannya sistem.

Sistem ini terbagi dari beberapa subproses di dalamnya. Yaitu *lampOnBySensor, lampOnByNeighbour* dan *lampOnByDelay* mengikuti *state chart* pada metode TALiSMaN. Pada Gambar 3.3 menjelaskan garis besar dari alur jalannya sistem secara keseluruhan berdasarkan subproses yang ada. Sebelum sistem dimulai alamat dari masing-masing *node lampu* harus ditentukan terlebih dahulu. Alamat pada tiap *node* lampu ditetapkan berdasarkan urutan dan keberadaannya dalam sistem. Jalannya sistem diawali dengan pembacaan data dari sensor PIR adakah pejalan kaki yang masuk dalam jangkauan sensor. Jika tidak ada maka subproses *lampOnByNeighbour* akan dijalankan. Subproses *lampOnByNeighbour* akan menjalankan proses bagaimana tiap *node* mengatur pencahayaan dari data yang dikirim oleh *node* sumber ketika sistem dalam keadaan *default* atau saat tidak mendeteksi pejalan kaki.

Apabila masukan dari proses membaca sensor PIR menunjukkan adanya pejalan kaki yang terdeteksi dalam jangkauan sensor, maka subproses *lampOnBySensor* dijalankan. Pada subproses *lampOnBySensor* dijalankan proses bagaimana tiap *node* mengatur pencahayaan ketika ada pejalan kaki melewati sensornya dan mengirimkan pesan untuk mengatur pencahayaan lampu pada *node* lain. Setelah subproses *lampOnBySensor* dijalankan, sensor membaca lingkungan sekitarnya apakah pejalan

telah meninggalkan *node* tersebut. Setelah pejalan kaki meninggalkan *node* tersebut dan menuju node lainnya *delay counter* diaktifkan. *Delay counter* dijalankan untuk mengatasi keterbatasan jangkauan dari sensor, untuk mengatasi daerah antar *node* yang tidak terdeteksi oleh sensor. Dalam masa *delaycounter* tersebut subproses *lampOnByDelay* dijalankan. Pada subproses ini, sistem memeriksa apakah pejalan kaki telah tiba pada *node* lain di sebelahnya.

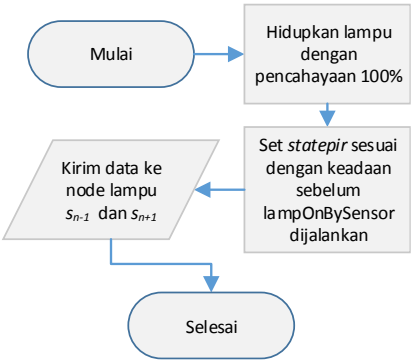


Gambar 3.3 Diagram Alir Sistem

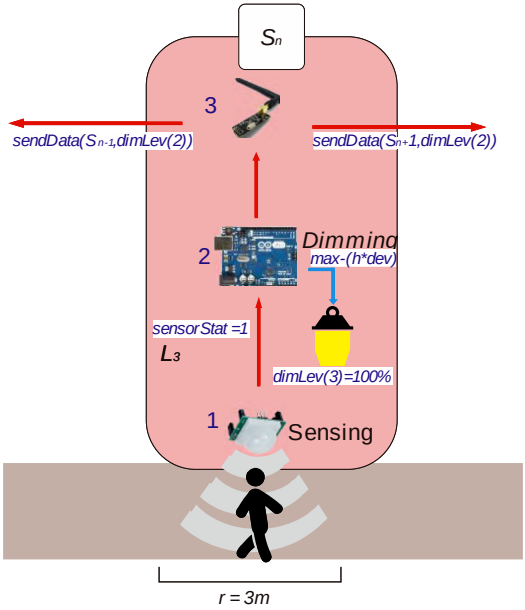
3.4.1 Diagram Alir Subproses *LampOnBySensor*

Seperti yang dijelaskan sebelumnya pada subproses *lampOnBySensor* dijalankan proses bagaimana tiap *node* mengatur pencahayaan ketika ada pejalan kaki melewati sensornya dan bagaimana pengiriman pesan untuk mengatur pencahayaan lampu pada *node* lain. Pada diagram alir yang ditunjukkan Gambar 3.4 dan ilustrasi Gambar 3.5 jika ada pejalan kaki yang terdeteksi, lampu pada *node* (S_n) yang mendeteksi tersebut akan menyala dengan tingkat pencahayaan D_{level} = 100% dan status bahwa *node* mendeteksi pejalan kaki ditetapkan *statepir=1*. Setelah itu *node* S_n akan mengirim data kepada *node* yang berada disebelahnya, *node* S_{n-1} dan *node* S_{n+1} terus menerus dengan jeda waktu 0,01 detik. Jeda dibuat secepat mungkin sesuai dengan spesifikasi yang dimiliki oleh modul nRF24L01+ dan mengurangi *data loss* ketika pengiriman data. Data yang dikirimkan berisi informasi tentang alamat *node* pengirim paket, alamat *node* yang dituju setelahnya, jumlah hop (*h*) yang tersisa dimana pada awal pengiriman diatur nilainya *h=3* , dan status sensor ketika data dikirim.

Ketika *node* dalam mode *lampOnBySensor*, node akan mengabaikan paket dari *node* lain, karena *node* diatur hanya untuk mengirimkan data saja atau mode TX. Hal itu dilakukan karena *node* memegang peran utama dalam sistem saat menjalankan subproses *lampOnBySensor*.



Gambar 3.4 Diagram Alir Subproses *LampOnBySensor*



Gambar 3.5 Ilustrasi berjalannya subproses *lampOnBySensor*

3.4.2 Diagram Alir Subproses *LampOnByNeighbour*

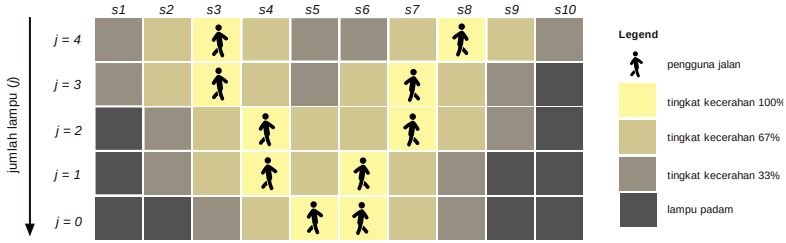
Subproses *lampOnByNeighbour* akan menjalankan proses bagaimana tiap *node* mengatur pencahayaan dari data yang dikirim oleh *node* sumber ketika sistem dalam keadaan *default*. Keadaan default adalah disaat *node* lampu tidak mendeteksi adanya pejalan kaki dalam jangkauan sensor dan lampu dalam kondisi padam.

Ketika *node* lain yang berada pada $S_{n-2} < S_n < S_{n+2}$ menjalankan proses *lampOnBySensor* maka ada data yang dikirim pada *node* S_n. *Node* S_n akan memeriksa adakah data yang ditujukan padanya. Jika tidak ada, maka *node* lampu S_n akan menjadi mode *default*. Jika ada, data akan dibaca dan disimpan sementara. Pada kondisi *best case* dimana data yang masuk berasal dari 1 *node*, *node* S_n akan menyala dengan tingkat pencahayaan sesuai dengan D_{level} yang ditentukan oleh sisa hop (h) ke-berapa data diterima (*dimLevel(h)*) dengan persamaan berikut:

max = 100% *tingkat pencahayaan maksimum
dev = 33,33% *deviasi pencahayaan
dimLevel(h) = max – (h x dev) (3.1)

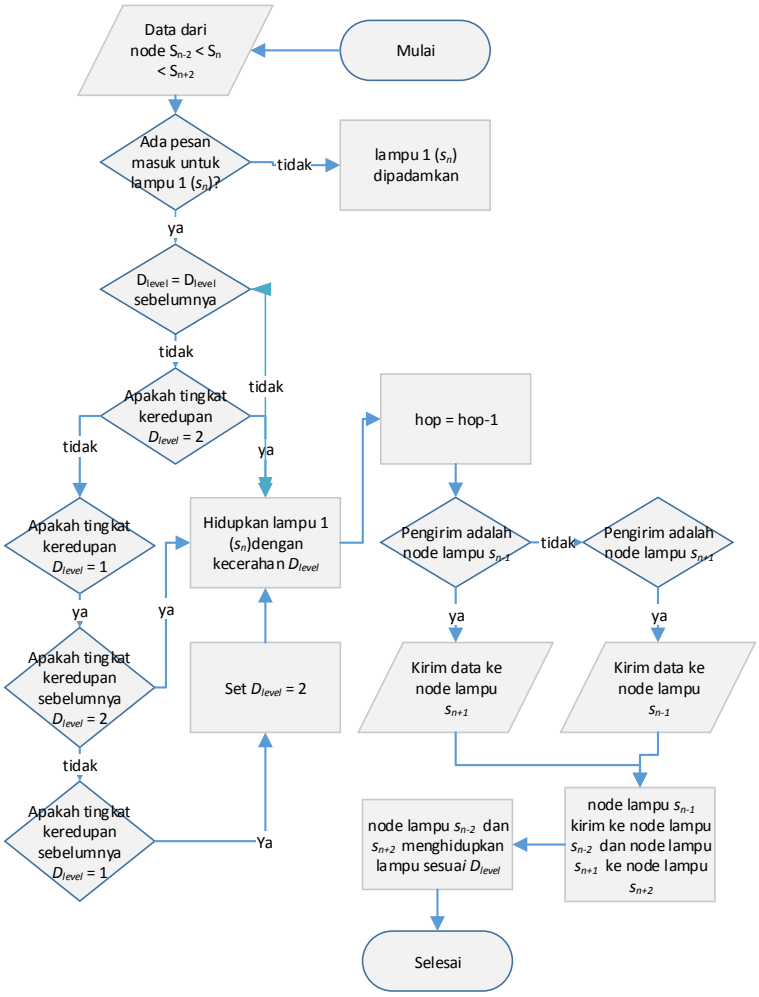
Namun apabila ada lebih dari satu *node* sumber data maka akan perlu dilakukan penyesuaian untuk mengurangi potensi masalah. Dalam kondisi tersbut akan banyak kemungkinan yang berpotensi menganggu jalannnya sistem. Seperti yang kita ketahui berdasarkan Gambar 3.66 potensi masalah adalah sebagai berikut:

a. S_5 ketika $j = 3$; S_5 akan menerima data dari *node* S_3 dan S_7 dimana sama-sama menerima D_{level} = 1 (33%),
b. S_5 dan S_6 ketika $j = 2$; S_5 akan menerima data D_{level} = 1(33%) dari *node* S_7 dan D_{level} = 2 (67%) dari *node* S_4 begitu pula S_6 akan menerima data D_{level} = 1(33%) dari *node* S_4 dan D_{level} = 2 (67%) dari *node* S_7,
c. S_5 ketika $j = 1$; S_5 akan menerima data dari *node* S_4 dan S_6 dimana sama-sama menerima D_{level} = 2 (67%).



Gambar 3.6 Ilustrasi jalannya sistem

Pada point a dan c dimana D_{level} dari masing-masing data yang diterima dari dua *node* sumber sama, bisa diabaikan karena pada akhirnya *node* S_n menyala sesuai dengan tingkat pencahayaan yang dibutuhkan sistem.



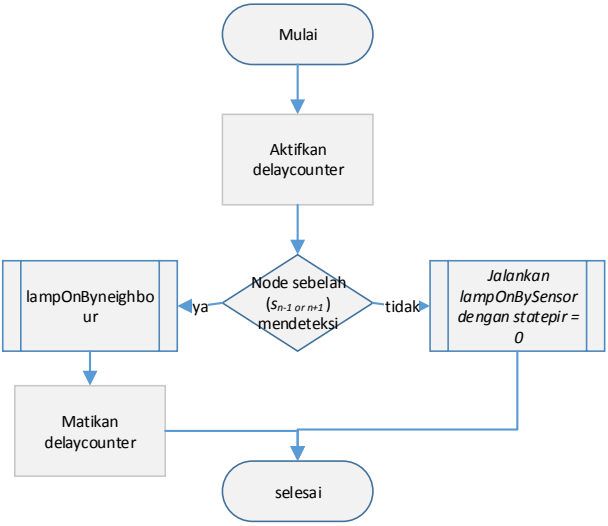
Gambar 3.7 Diagram Alir Subproses *LampOnByNeighbour*

Namun pada point b dimana pada S_5 dan S_6 menerima D_{level} yang berbeda, akan menyebabkan lampu berkedip-kedip karena D_{level} yang diterima berbeda. Untuk menanganinya, ketika data diambil status D_{level} terbesar dengan memeriksa status D_{level}

sebelumya, seperti yang ditunjukkan pada Gambar 3.77. Lalu nyalakan lampu pada *node* dengan D_{level} yang sesuai. Setelah itu *node* lampu S_n mengirimkan kembali data pada S_{n+1} atau S_{n-1} sesuai dengan alamat *node* sumber.

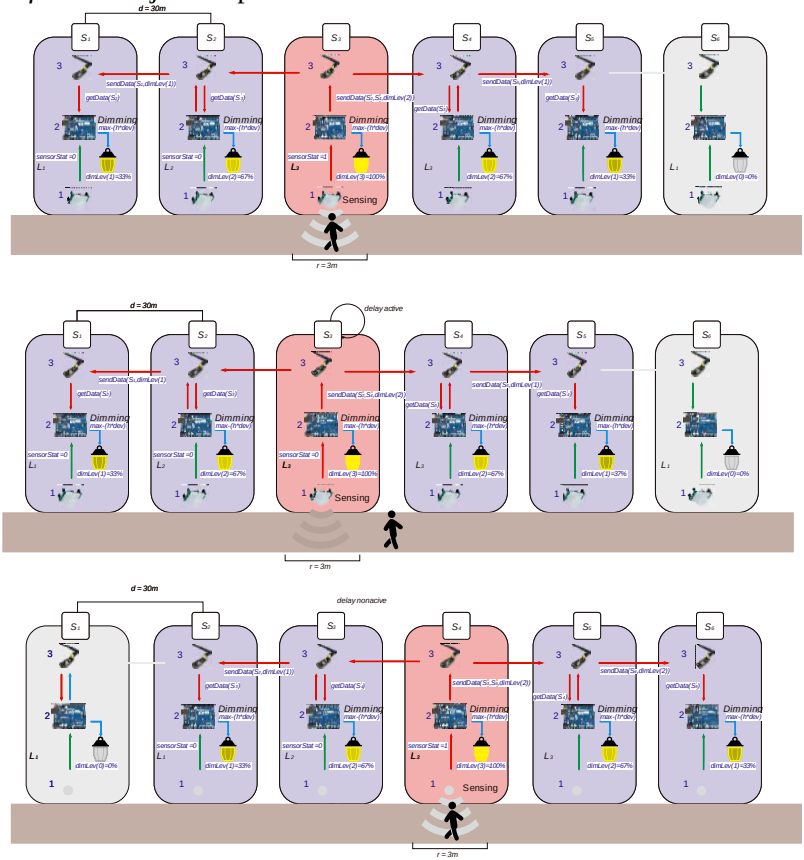
3.4.3 Diagram Alir Subproses *LampOnByDelay*

Blank zone yaitu area diantara dua *node* dimana sensor tidak bisa menjangkau pejalan kaki di dalam sistem. Ketika pejalan kaki berada pada *blank zone*, apabila sistem berjalan hanya berdasarkan status sensor, maka disaat pejalan kaki berjalan dari *node* satu ke lainnya lampu akan padam. *Delaycounter* dimaksudkan untuk memberikan waktu pejalan kaki menyelesaikan perjalanannya menuju *node* lain. Untuk menentukan lama *delaycounter* digunakan persamaan pada metode TALiSMaN, dengan jarak antar lampu sebesar d_{adj} = 30 m dan *range* sensor PIR d_{rad} = 3 m, dan kecepatan minimal pejalan kaki v = 1,4 m/s didapatkan lama delay adalah 17,4 detik. Diagram alir fungsi *lampOnByDelay* ditunjukkan pada Gambar 3.8.



Gambar 3.8 Diagram Alir Subproses *LampOnByDelay*

Dalam masa berjalannya *delaycounter* proses *lampOnByDelay* mendeteksi apakah pejalan kaki telah sampai di *node* lain di sebelahnya ($S_{n-1\ or\ n+1}$). Jika pejalan kaki telah terdeteksi oleh sensor pada *node* S_{n-1} atau S_{n+1} maka proses *lampOnByNeighbour* dijalankan dan *delaycounter* dibatalkan. Jika pejalan kaki tidak terdeteksi oleh sensor pada *node* S_{n-1} atau S_{n+1} hingga masa *delaycounter* telah habis, *node* kembali pada mode *default* ditunjukkan pada Gambar 3.9.



Gambar 3.9 Ilustrasi Jalannya Subproses *LampOnByDelay*

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan perangkat lunak dan perangkat keras pada sistem. Cakupan implementasi sistem meliputi perangkat sistem penerangan lampu adaptif dan proses berjalannya sistem.

4.1 Lingkungan Implementasi

Karena sistem merupakan sistem *embedded* pada perangkat mikrokontroler, maka lingkungan implementasi dibagi menjadi dua bagian, yaitu lingkungan implementasi perangkat keras dan implementasi perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Lingkungan Implementasi erangkat keras yang digunakan dalam implementasi sistem penerangan jalan adaptif dijelaskan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

Perangkat *Embedding* (Komputer)	Prosesor: • Intel(R) Core(TM) i5-6500 CPU @ 3.46GHz Memori : • 8 GB Sistem Operasi: • Microsoft Windows 8.1 Embedded 64-bit Perangkat Sistem : - Arduino Uno R3 - Sensor PIR HC-SR501 - Transciever nRF24L01+
Perangkat *Embedded* (Mikrokontroler)	Mikrokontroler: • ATmega328 Tegangan:

	• 5V Memori *Flash*: • 32 KB SRAM • 2 KB

4.1.2 Lingkungan Implementasi Perangkat Lunak

Lingkungan Implementasi erangkat keras yang digunakan dalam implementasi sistem penerangan jalan adaptif dijelaskan pada Tabel 4.2.

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak

Perangkat *Embedding* (Komputer)	Sistem Operasi: • Microsoft Windows 8.1 Embedded 64-bit Software: • Arduino IDE 1.6.7 • Fritzing 0.9.3b 64 bit

4.2 Implementasi Perangkat Keras

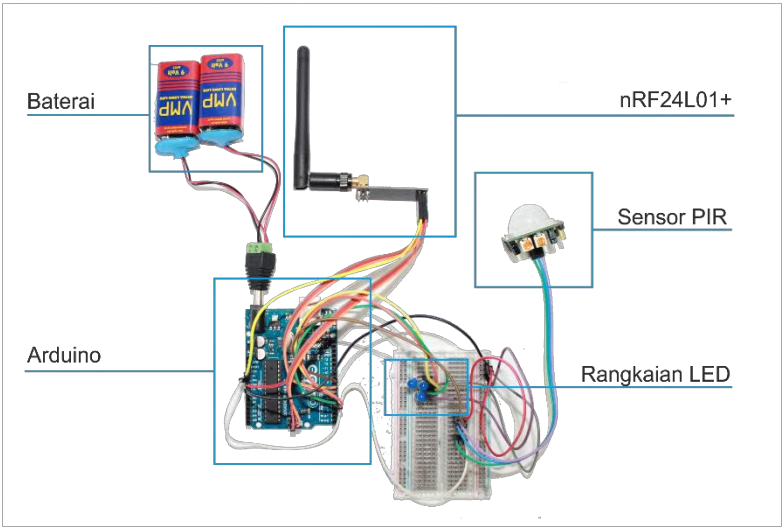
Seperti yang dijelaskan pada bab Analisis dan Perancangan, perangkat keras diimplementasikan pada papan mikrokontroler Arduino Uno R3 yang dihubungkan dengan beberapa modul perndukung. Pada bab Analisis dan Perancangan, aktuator dari sistem adalah *controlled dimmer light* yang berupa rangkaian elektronik untuk mengatur pencahayaan lampu dengan mengatur gelombang listrik dari masukan daya listrik PLN. Akan tetapi implementasi yang digunakan adalah menggunakan lampu LED (*Light Emitting Dioda*) yang dirangkai pada *breadboard* untuk mempermudah pengujian. Penjelasan lebih lanjut tentang perangkat yang digunakan dijelaskan pada Tabel 4.3**.**

Tabel 4.3 Perangkat Keras yang Digunakan

Perangkat Keras		Jumlah
Mikrokontroler	Arduino Uno Rev 3	Lima buah
Sensor	Sensor PIR HC-SR501	Lima buah

Kabel	Kabel USB	Lima buah
	Kabel *jumper*	Empat set
Board	*Breadboard*	Lima buah
Baterai	*Baterai 9 volt*	10 buah
Komunikasi	*Transciever* nRF24L01+	Lima buah
Aktuator	Lampu LED	15 buah

Perangkat keras yang digunakan pada tiap *node* sama satu dan lainnya. Dalam realisasinya tiap *node* harus berurutan sesuai dengan alamatnya. Fungsi dari masing-masing *node* lampu adalah untuk mendeteksi pejalan kaki yang melewati sistem dan sebagai alat komunikasi antar *node* pada sistem. Gambar rangkaian ditunjukkan pada Gambar 4.1.



Gambar 4.1 Desain Rangkaian

Gambar 4.1 menunjukkan dalam implementasi perangkat keras, sumber daya berasal dari dua buah baterai 9 volt dan aktuatornya adalah tiga lampu LED. Tiga buah LED pada *breadboard* akan mensimulasikan pengaturan tingkat pencahayaan

lampu *controlled dimmer light* pada sistem. Tabel 4.4 menunjukkan pin koneksi dengan Arduino dan deskripsi dari tiap lampu LED.

Tabel 4.4 Koneksi Pin Antar Arduino dan Aktuator LED

PIN	Modul	Deskripsi
D4	*LED 1*	Sebagai aktuator LED 1
D5	*LED 2*	Sebagai aktuator LED 2
D6	*LED 3*	Sebagai aktuator LED 3

Pengaturan tingkat pencahayaan *controlled dimmer light* seperti yang dijelaskan pada Tabel 4.5 menunjukkan ada 4 tingkat pencahayaan lampu pada sistem dengan deviasi *dev=33,33%*, yaitu *dimLev(0)* = 0%, *dimLev(1)* = 33%, *dimLev(2*) = 67%, dan *dimLev(3)* = 100%. Tabel 4.5 menggambarkan simulasikan tingkat pencahayaan tersebut pada lampu LED.

Tabel 4.5 Tingkat pencahayaan dan LED

dimLevel()	LED
dimLevel(0) = 0%	LED 1 padam, LED 2 padam, LED 3 padam
dimLevel(1) =33%	LED 1 menyala, LED 2 padam, LED 3 padam
dimLevel(2) =67%	LED 1 menyala, LED 2 menyala, LED 3 padam
dimLevel(3) =100%	LED 1 menyala, LED 2 menyala, LED 3 menyala

4.3 Implementasi Perangkat Lunak

Perangkat lunak yang diimplementasikan pada sistem penerangan jalan adaptif, terbagi dari 4 fungsi dari 4 sub-proses yang telah dijelaskan pada bab analisis dan perancangan sistem. Fungsi yang akan dibahas pada bab ini adalah *lampOnBySensor*, *lampOnByNeighbour* dan *lampOnByDelay*. Penjelasan setiap bagian fungsi akan dijelaskan lebih lanjut pada masing-masing subbab.

4.3.1 Inisialisasi Sistem

Proses inisialisasi pada Arduino merupakan suatu hal penting. Inisialisasi dilakukan hanya di awal ketika perangkat sensor pertama kali dinyalakan. Inisialisasi yang dilakukan adalah inisialisasi fungsi *setup()* dan variabel global. Inisialisasi fungsi *setup()* Arduino seperti yang ditunjukkan pada Gambar 4.2 dapat dijelaskan melalui beberapa tahapan sebagai berikut:

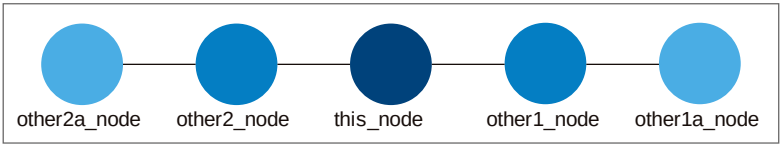
1. Buka *serial port* dan inisialisasi *baud rate.*
2. Inisialisasi *pin mode* sensor PIR .
3. Inisialisasi *pin mode* LED, terdapat 3 buah LED untuk mensimulasikan pengaturan pencahayaan.
4. Set sensor PIR pada kondisi LOW atau padam.
5. Kalibrasi sensor PIR sebelum dijalankan selama 10 detik
6. Inisialisasi pin yang digunakan sebagai radio pada RF24 dengan input *pin radio* pada objek "radio".
7. Tambahkan objek "network" untuk jaringan RF24network dengan dasar objek "radio".
8. Buka SPI port untuk komunikasi nRF24L01+ dengan Arduino
9. nRF24L01+ dijalankan.
10. Jalankan jaringan RF24 dengan *channel* dan alamat *node* yang sudah ditentukan.

```
1  Open Serial.begin(baud rate)
2  Initialize pinMode(PIR pin, INPUT)
3  Initialize pinMode(LED pin1, OUTPUT)
4  Initialize pinMode(LED pin2, OUTPUT)
5  Initialize pinMode(LED pin3, OUTPUT)
6  Set digitalWrite(PIR pin, LOW)
7  While callibrationTime = 10 second
8    calibratePIR
9  EndWhile
10 Set RF24 radio(Radio pin)
11 Add RF24Network network(radio)
12 Open SPI.begin
13 Start radio.begin
14 Start network.begin(channel,this_nodeaddress)
```

Gambar 4.2 Inisialisasi serial port, pin sensor, dan pin radio

Proses inisialisasi variabel global dilakukan di awal sistem berjalan, untuk menentukan variabel-variabel yang dapat digunakan pada keseluruhan sistem. Inisialisasi variabel global penting sistem pada Gambar 4.4 dijelaskan sebagai berikut:

1. Inisialisasi alamat pada tiap *node* yang berhubungan pada sistem yang diilustrasikan pada Gambar 4.3.
   a. this_nodeaddress, adalah alamat *node* S_n RF24 dimana program dijalankan.
   b. other1_nodeaddress, adalah alamat *node* RF24 pada *node* S_{n+2} yang bersebelahan langsung *node* S_{n+1}.
   c. other1a_nodeaddress, adalah alamat *node* RF24 pada *node* S_{n-1} yang bersebelahan langsung *node* S_n.
   d. other2_nodeaddress, adalah alamat *node* RF24 pada *node* S_{n-1} yang bersebelahan langsung *node* S_n.
   e. other2a_nodeaddress, adalah alamat *node* RF24 pada *node* S_{n-2} yang bersebelahan langsung *node* S_{n-1}.



Gambar 4.3 Ilustrasi pengalamatan *node*

2. Inisialisasi *payload* pada paket pesan di jaringan, payload berisi beberapa data yang digunakan sebagai salah satu masukan dalam pengambilan keputusan. Isi dari payload tersebut adalah:
   a. Data alamat *node* sumber pengirim data.
   b. Data alamat *node* tujuan yang bersebelahan langsung (*node* S_{n-1} atau S_{n+1}) dengan *node Sn*.
   c. Data status sensor pada *node* pengirim data.
   d. Data jumlah sisa *hop* pengiriman data yang harus dilakukan lagi pada *node* selanjutnya.

```
Set nodeAdress(this_nodeadress)
Set nodeAdress(other1_nodeadress)
Set nodeAdress(other1a_nodeadress)
Set nodeAdress(other2_nodeadress)
Set nodeAdress(other2a_nodeadress)
Initialize Struct payload
  source address
  other_nodeaddress
  sensor state
  hop remaining
```

Gambar 4.4 Inisialisasi *payload* dan alamat pada tiap *node*

4.3.2 Sistem Utama Dalam Fungsi Loop

Proses berjalannya sistem utama terdapat di dalam fungsi *loop()* pada Arduino. Sistem utama dijalankan terus menerus selama sistem berjalan di dalam fungsi *loop()*. Sistem utama adalah implementasi dari rancangan sistem penerangan jalan adaptif yang telah dibahas pada bab sebelumnya. Pada sistem utama ini fungsi-fungsi subproses sistem diimplementasikan dengan menerapkan metode TALiSMaN. Penjelasan proses berjalannya sistem utama yang ditunjukkan pada Gambar 4.5 dijelaskan secara bertahap sebagai berikut:

1. Jalannya sistem diawali dengan menyimpan *timestamp* dari awal sistem dijalankan, untuk membantu menentukan lama *delay counter* dalam fungsi *lampOnByDelay*.
2. Pada sistem utama terdapat beberapa *flag* yang menunjukkan status dari sensor dan berjalannya sebuah fungsi
   a. *Flag statePIR* menyimpan status bahwa *node* telah mendeteksi pejalan kaki.
   b. *Flag stateDelay* menandakan bahwa *delay counter* telah diaktifkan.
3. Pada kasus pertama, jika sensor mendeteksi keberadaan pejalan kaki, namun *node* masih dalam keadaan default dimana *delay counter* belum diaktifkan dan *node* belum mendeteksi adanya pejalan kaki sebelumnya maka fungsi *lampOnBySensor* dipanggil.

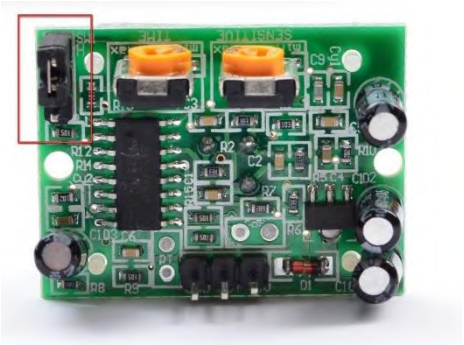
4. Fungsi *lampOnBySensor* menjalankan proses bagaimana tiap *node* mengatur pencahayaan ketika ada pejalan kaki terdeteksi sensor dan mengirimkan pesan untuk mengatur pencahayaan lampu pada *node* lain.
5. Pada kasus kedua, jika pejalan kaki meninggalkan *node*, atau tidak terdeteksi lagi oleh sensor dimana *statePIR* berubah dari 0 menjadi 1 dan fungsi *ceksensor()* mengembalikan nilai 0, dan *delay counter* belum diaktifkan, maka *delay counter* diaktifkan dan *timestamp* awal berjalannya *delay counter* disimpan untuk membantu menentukan lama *delay counter* dalam fungsi *lampOnByDelay*.
6. Pada kasus ketiga, jika *delay counter* telah diaktifkan yang ditandai dengan perubahan *stateDelay* dari 0 menjadi 1 saat sensor telah ditinggalkan pejalan kaki, maka fungsi *lampOnByDelay* dipanggil.
7. *Delay counter* dijalankan untuk mengatasi keterbatasan jangkauan dari sensor, untuk mengatasi daerah antar *node* yang tidak terdeteksi oleh sensor dengan menjaga *node* lampu tetap menyala adalah waktu yang ditentukan.
8. Fungsi *lampOnByDelay* memeriksa status data dari *node* lain apakah *node* yang besebelahan langsung S_{n+1} atau *S*n-1 telah mendeteksi pejalan kaki dalam masa *delay counter* diaktifkan.
9. Jika *node* yang besebelahan langsung S_{n+1} atau *S*n-1 tidak mendeteksi adanya pejalan kaki maka *delay counter* dibatalkan, dan *node* lampu dikembalikan pada mode *default*.
10. Pada kasus keempat, jika dalam masa *delay counter* diaktifkan ada pejalan kaki yang terdeteksi sensor, maka *delay counter* dibatalkan, dan *node* lampu dikembalikan pada mode *default*.
11. Pada kasus kelima, jika tidak ada pejalan kaki yang terdeteksi sensor dan *node* pada mode default, maka fungsi *lampOnByNeighbour* dipanggil
12. Fungsi *lampOnByNeighbour* menjalankan proses bagaimana tiap *node* mengatur pencahayaan dari data yang dikirim oleh *node* sumber.

```
1   Start network.update()
2   Set currenttime = millis()
3   Set max hop
    //case 1
4   If sensorCheck() = TRUE And stateDelay = 0
     And ( statePIR = 0 Or statePIR = 1 )
5     Call lampOnBySensor(statePIR,max hop)
6     Set statePIR = 1
    End If
    //case 2
7   If sensorCheck() = FALSE And statePIR = 1
     And stateDelay = 0
8     Set start = millis()
9     Set stateDelay = 1
    End If
    //case 3
10  If sensorCheck() = FALSE And statePIR = 1
     And stateDelay = 1
12    If Call lampOnByDelay() = TRUE
13      Set stateDelay=1
      End If
14    Else
15      Set stateDelay = 0
16      Set statePIR = 0
17      Set start =0
      End If
    End If
    //case 4
18  If sensorCheck() = TRUE And statePIR = 1
     And stateDelay = 1
19    Set stateDelay = 0
20    Set statePIR = 0
21    Set start = 0
    End If
    //case 5
22  If sensorCheck() = FALSE And statePIR = 0
     And stateDelay = 0
23    Call lampOnByNeighbour()
    End If
```

Gambar 4.5 Inisialisasi *payload* dan alamat pada tiap *node*

Fungsi *ceksensor()* memeriksa masukan dari sensor PIR HC-SR501 pada pin digital. Untuk memastikan masukan dari sensor

PIR HC-SR501 tetap memberikan nilai HIGH selama ada gerakan yang terdeteksi atau *retriggering*, maka *jumper* pada sensor diposisikan pada posisi H seperti yang ditunjukkan pada Gambar 4.6.



Gambar 4.6 Jumper diposisikan pada posisi H

Nilai kembalian pada fungsi ceksensor adalah 1 untuk menandakan sensor PIR pada kondisi HIGH dan 0 untuk menandakan sensor PIR pada kondisi LOW ditunjukkan pada Gambar 4.7.

```
1  If digitalRead(PIR pin)= HIGH)
2    Set sensorstate  = 1
3  End If
4  If digitalRead(PIR pin)= HIGH)
5    Set sensorstate  = 0
6  End If
7  Set return value = sensorstate
```

Gambar 4.7 Fungsi *ceksensor()*

4.3.3 Proses LampOnBySensor

Fungsi *lampOnBySensor* dipanggil ketika *node* lampu dalam keadaan telah mendeteksi pejalan kaki dari keadaan *default* atau ketika *delay counter* diaktifkan dan *node* lampu S_n tidak menerima *trigger*, baik dari *node* lain S_{n+1} atau S_{n-1} ketika mendeteksi pejalan kaki atau pejalan kaki yang melewati *node* lampu S_n tersebut. Penjelasan proses berjalannya fungsi *lampOnBySensor* yang

ditunjukkan pada Gambar 4.8 dijelaskan secara bertahap sebagai berikut:

1. Parameter pada fungsi *lampOnBySensor* ada dua, yaitu:
   a. Integer *statesensor*, adalah status deteksi sensor pada *node* lampu S_n yang dikirimkan kepada *node* lampu lain S_{n+1} atau S_{n-1}, apabila sensor *node* lampu S_n mendeteksi pejalan kaki *statesensor = 1* jika tidak *statesensor = 0*
   b. Integer *hopNum*, adalah jumlah *hop* maksimal terhitung dari *node* lampu S_n, dikirim pada *node* lampu S_{n+1} atau S_{n-1}, pada sistem ini terdapat 2 hop maka *hopNum = hop +1*, setelah dikirim nilai *hopNum* di-*decrement* dengan 1 sehingga ketika pesan diteruskan *node* lampu S_{n+2} atau S_{n-2} sisa hop = 0 maka tidak bisa mengirim data pada *node* lampu berikutnya
2. Ketika fungsi dipanggil, maka tingkat pencahayaan diubah menjadi *dimLevel(hopNum)*, pada *node* S_n *hopNum = 3*, maka *dimLevel(3) = 100%*.
3. Jeda pengiriman data diatur berdasarkan *interval* yang ditentukan, pada sistem ini diatur dengan lama 50 milisekon, agar adanya *packet loss* dapat segera ditangani oleh fungsi *auto resend* pada *library* RF24Network.
4. Inisialisasi nilai pada *payload* untuk pengiriman data ke *node* lampu S_{n+2} dan S_{n-2} sesuai dengan yang sudah dijelaskan sebelumnya pada Gambar 4.3.

```
1  Funtion lampOnBySensor(statesensor, hopNum)
2   Start network.update()
3   Set hopNum = hopNum - 1
4   Call dimLevel(hopNum)
5   Set now = (millis()
6   If now - last_sent >= interval
      Set payload1 =
7      {this_node,other_node1,hopNum,statesensor}
8     Set RF24NetworkHeader header1(other1_node)
      Start network.write
9      (header1,&payload1, sizeof(payload1))
      Set payload2 =
```

```
10      {this_node,other_node2,hopNum,statesensor}
11    Set RF24NetworkHeader header2(other2_node)
      Start network.write
12     (header2,&payload2, sizeof(payload2))
      Set last_sent = now
13   End IF
End of Funtion
```

Gambar 4.8 Fungsi *lampOnBySensor*

5. *Header* diatur sesuai dengan alamat *node* S_{n+2} dan S_{n-2}.
6. Paket data dikirimkan dengan menggunkan fungsi *network.write* setelah mengatur alamat tujuan, isi data dan panjang data yang dikirim.

4.3.4 Proses LampOnByNeighbour

Fungsi *LampOnByneighbour* dipanggil ketika *node* lampu dalam keadaan tidak sedang mendeteksi pejalan kaki dan *delay* counter dalam keadaan tidak aktif. Fungsi *LampOnByneighbour* mengatur tingkat pencahayaan lampu pada *node* lampu S_n berdasarkan paket data yang diterima dari *node* yang bersebelahan langsung dengannya S_{n+1} atau S_{n-1}. kaki atau pejalan kaki yang melewati *node* lampu S_n tersebut. Penjelasan proses berjalannya fungsi *lampOnBySensor* yang ditunjukkan pada Gambar 4.9 dijelaskan secara bertahap sebagai berikut:

1. Dengan memangil fungsi *network.available()* RF24 diatur pada mode RX memeriksa adanya paket data yang diterima oleh *node* lampu S_n .
2. Jika ada, data *header* diambil untuk kemudian paket data yang siterima dan disimpan dalam *payload1*.
3. Jika paket data yang diterima sesuai dengan alamat *node* lampu S_n maka *flag statepacket = 1*, *flag statepacket* adalah *flag* yang menandai adanya data yang diterima *node* lampu S_n, jika tidak kembalikan *node* lampu pada mode *default*.
4. Alamat pengirim paket data diperiksa untuk menentukan alamat *node* lampu berikutnya data dikirim.
5. Lampu pada *node* lampu S_n dinyalakan dengan tingkat pencahayaan sesuai dengan nilai *sisa hop* yang diterima, pada

bagian ini fungsi *dimlevelHandling()* yang dipanggil untuk mengatasi permasalahan pada BAB 3.4.2, yang akan dijelaskan selanjutnya.

6. Nilai sisa hop yang diterima diperiksa, jika tidak ada hop yang tersisa maka paket data tidak akan dikirim pada *node* lampu selanjutnya.
7. Jika tidak ada data yang diterima pada *node* lampu S_n maka *node* lampu kembali pada mode *default* dan semua *flag* nilainya dirubah kembali menjadi 0.

```
1   Function lampOnByNeighbour()
2    Start network.update()
3    If network.available()
4      Get RF24NetworkHeader header
5      Get network.read
         (header,&payload1, sizeof(payload1))
6      If payload1.other_node = this_node
7        Set statepacket = 1
8        Add levellight = payload1.hop
       End If
9      Else
10       Set statepacket = 0
11       Set levellight = 0
       End If
12     If payload1.source = other_node2
13       Add toadress = other_node1
       End If
14     Else If payload3.source = other_node1
15       Add toadress = other_node1
       End If
16     Call dimlevelHandling(levellight)
17     Add hops = payload1.hopNum
18     If hops > 0 && hops <= 3
19       Set hops= hops-1
20       Set payload2 =
           (payload1.source,other_node2,hops,0)
21       Set RF24NetworkHeader header(other_node2)
22       Start network.write
           (header,&payload2, sizeof(payload2))
       End If
23     Else
24       print "This is the last node"
```

```
    End If
  End If
25 Else
26   Set statepacket = 0
27   Set dimLevel(0)
28   Set last = 0
29   Set levellight = 0
  End If
End Of Function
```

Gambar 4.9 Fungsi *lampOnByNeighbour*

Seperti yang dijelaskan sebelumnya pada BAB 3.4.2, pada proses *lampOnByNeighbour* diperlukan penanganan khusus ketika ada lebih dari satu *node* lampu yang mendeteksi adanya pejalan kaki dalam jangkauan $S_{n+5} > S_n > S_{n-2}$. Karena dalam mode *lampOnByNeighbour node* lampu dalam keadaan RX, maka akan ada kemungkinan *node* lampu mendapatkan *command* yang berbeda dalam satu waktu. Fungsi *dimlevelHandle()* ditujukan untuk mengatasi masalah tersebut. Berikut penjelasan tentang fungsi *dimlevelHandle()* yang ditunjukkan oleh gambar:

1. Flag *statepacket* = 1 menandakan adanya paket data yang diterima pada *node* lampu
2. Jika tidak ada paket yang diterima maka tingkay pencahayaan kembali pada mode *default* yaitu 0%
3. *Flag last* menyimpan status tingkat pencahayaan sebelumnya.
4. Jika tingkat pencahayaan sebelumya sama dengan tingkat pencahayaan yang diterima, maka tingkat pencahaayan tetap menggunakan tingkat pencahayaan sebelumnya.
5. Jika status tingkat pencahayaan yang diterima pada saat ini *level = 2* maka tingkat pencahayaan diatur *dimLevel(2)*.
6. Jika status tingkat pencahayaan yang diterima pada saat ini *level = 1* , diperiksa kembali *last state* dari lampu, apabila *last = 2* maka tingkat pencahayaan diatur *dimLevel(2)* dan *last state* diperbaharui menjadi *last = 2*. Namun apabila *last = 1* maka tingkat pencahayaan diatur mana yang lebih besar antara *last state* dan *current state*, *dimLevel(2)* dan *last state* diperbaharui menjadi *last = 1*.

```
1   Function dimlevel(level)
2    If statepacket = 1
3      If level = 2
4        Call dimLevel(2)
5        Set last = 2
       End If
6      If level = 1
7        If last = 2
8          Call dimLevel(2)
         End If
9        Else If last = 1
10         Call dimLevel(1)
         End If
11       Set last = 1
       End If
12     Else last = 0
     End If
13   Else
14     Call dimLevel(0)
     End If
   End Of Funtion
```

Gambar 4.10 Fungsi *dimlevelHandle()*

Untuk mengatur tingkat pencahayaan lampu pada tiap *node* menggunakan aktuator 3 buah LED digunakan fungsi *dimLevel()*. Pada sistem terdapat 4 tingkat pencahayaan, level 0 hingga 3. Fungsi mengatur jumlah LED yang menyalaberdasarkan masukan pada parameter *level* ditunjukkan pada gambar.

```
1   Function dimlevelact(int level)
2     If level = 3
3       Set digitalWrite(LED pin1, HIGH);
4       Set digitalWrite(LED pin2, HIGH);
5       Set digitalWrite(LED pin3, HIGH);
      End If
6     If level = 2
7       Set digitalWrite(LED pin1, HIGH);
8       Set digitalWrite(LED pin2, HIGH);
        Set digitalWrite(LED pin3, LOW);
9     End If
10    Else If level = 1
        Set digitalWrite(LED pin1, HIGH);
11      Set digitalWrite(LED pin2, LOW);
```

```
      Set digitalWrite(LED pin3, LOW);
12  End If
    Else If level = 0
13    Set digitalWrite(LED pin1, LOW);
14    Set digitalWrite(LED pin2, LOW);
      Set digitalWrite(LED pin3, LOW);
    End If
End Of Function
```

Gambar 4.11 Fungsi *dimLevel ()*

4.3.5 Proses LampOnByDelay

Ketika pejalan kaki meninggalkan area jangakauan sensor pada *node* lampu maka *flag statepir* akan berubah dari 0 menjadi 1. Pada saat itu *delay counter* diaktifkan dan fungsi *LampOnByDelay* dipanggil. Penjelasan proses berjalannya fungsi *lampOnByDelay* yang ditunjukkan pada Gambar 4.9 dijelaskan secara bertahap sebagai berikut:

1. Variabel *lodStat* adalah *flag* dari status berjalannya fungsi *LampOnByDelay*.
2. *ctrDelay* adalah lama berjalannya *delay counter*, waktu dihitung berdasarkan selisih waktu sekarang dengan *timestamp* pertama kali *flag stateDelay* berubah dari 0 menjadi 1. Apabila *ctrDelay* > 17 detik maka *lodStat = 0*, fungsi *LampOnByDelay* dan *delay counter* dibatalkan.
3. Jika dalam masa *delay counter* terdapat paket data yang ditujukan pada *node* lampu, maka paket dibaca dan diperiksa apakah data menujukkan *node* yang bersebelahan langsung mendeteksi adanya pejalan kaki, jika iya maka fungsi *LampOnByNeighbour* dijalankan dan fungsi *LampOnByDelay* dibatalkan serta status *lodStat = 1*, jika paket data menunjukkan *node* yang bersebelahan langsung tidak mendeteksi pejalan kaki maka fungsi *LampOnBySensor* dijalankan dengan parameter *statussensor* = 0 dan *max hop = 3* serta status *lodStat = 1*.
4. Jika tidak ada maka fungsi *LampOnBySensor* dijalankan dengan parameter *statussensor* = 0 dan *max hop = 3*, serta status *lodStat = 0;*

```
1   Function lampOnByDelay()
2     Start network.update()
3     Initialize lodStat = 0
4     Add ctrDly  = current - start
5     If ctrDly>0 And ctrDly<=17000 milisekon
        If network.available()
6         Get RF24NetworkHeader header
7         Get network.read
8          (header,&payload,sizeof(payload))
          If payload5.pirstatus = 1
9           Call lampOnByNeighbour()
10          Set lodStat = 0
          End If
11        Else If payload.pirstatus = 0
            Call lampOnBySensor(0,3)
12          Set lodStat=1
          End If
13      End If
14      Else
          Call lampOnBySensor(0,3)
          Set lodStat = 1
        End If
      End If
      Else
        Set lodStat = 0
      End If
      Return lodStat
    End Of Function
```

Gambar 4.12 Fungsi *lampOnByDelay*

(Halaman ini sengaja dikosongkan)

BAB V
PENGUJIAN DAN EVALUASI

Bab ini berisi bahasan mengenai uji coba dan evaluasi sistem penerangan jalan adaptif untuk pejalan kaki yang dikembangkan. Sistem akan diujicoba fungsionalitasnya dengan menjalankan rangkaian skenario pengujian yang telah ditentukan. Hasil evaluasi menjelaskan tentang rangkuman dari hasil pengujian, yang akan dijelaskan pada akhir bab ini. Uji coba fungsionaltas sistem meliputi uji coba reaksi sistem terhadap berbagai kondisi yang dilakukan pejalan kaki kepada sistem.

5.1 Uji Coba Fungsionalitas

Uji coba fungsionalitas merupakan sebuah pengujian terhadap jalannya fungsi-fungsi utama yang ada pada sistem. Fungsi utama pada sistem meliputi cara bagaimana tiap node beradaptasi terhadap pejalan kaki yang terdeteksi pada sistem, dan bagaimana sistem mengatur *node-node* lampu yang ada di dalamnya.

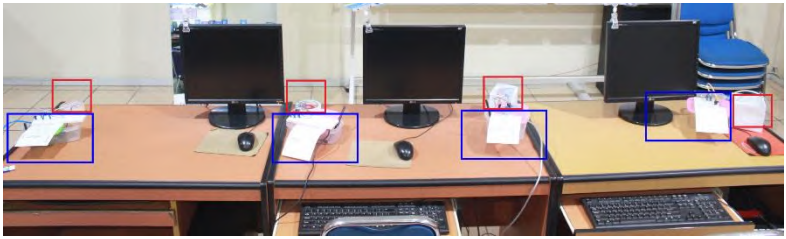
5.1.1 Lingkungan Uji Coba

Proses uji coba dilakukan pada lingkungan yang telah ditentukan. Pada masing masing *node* terdapat seorang responden yang merekam bagaimana reaksi *node* terhada pejalan kaki yang terdeteksi. Pada ujicoba ini untuk beberapa uji coba, responden juga berperan sebagai pejalan kaki dengan melambaikan tangannya ke arah sensor PIR pada tiap *node*. Dengan begitu uji coba *multi pedestrian detector* menjadi lebih mudah dan terkontrol. Lingkungan uji coba memiliki spesifikasi sebagai berikut:

- Lima *node* lampu, yang tiap *node* terdiri dari komponen berikut:
  - Papan mikrokontroler Arduino R3
  - Sensor PIR HC-SR501
  - Modul *transciever* radio nRF24L01+
  - Satu set kabel jumper

  - Kabel USB
- Dua responden untuk mensimulasikan aktivitas pada tiap *node*.
- Laptop Asus A46CM
  - Intel i7-3517U
  - RAM DDR3 4GB
  - Windows 8.1 Embedded Industry
- 3 PC Laboratorium AJK Teknik Informatika ITS.

Spesifikasi yang telah dijabarkan di atas merupakan spesifikasi lingkungan uji coba yang digunakan selama pengerjaan uji coba. Pengujian penerangan jalan adaptif dilakukan melalui aplikasi sistem yang telah dibuat. Gambaran lingkungan uji coba ditunjukkan seperti pada Gambar 5.1.



Gambar 5.1 Lingkungan Uji Coba

Pada Gambar 5.1 kotak berwarna merah adalah wadah berisikan sensor PIR dan kotak berwarna biru adalah rangkaian Arduino berserta LED dan *transciever* nRF24L01+. Sensor PIR diletakkan di dalam wadah untuk menghindari deteksi terhadap gerakan yang tidak diperlukan selama uji coba berlangsung.

5.1.2 Uji Coba Satu Pejalan Kaki Terdeteksi Tiap Node

Pada sistem penerangan jalan adaptif, jika ada satu *node* S_n yang mendeteksi adanya pejalan kaki maka *node* S_{n+1} dan *node* S_{n-1} tingkat pencahayaannya akan berubah menjadi dari padam menjadi *dimLevel(2)* dimana dua LED pada perangkat akan menyala. Begitu pula dengan *node* S_{n+2} dan *node* S_{n-2} tingkat pencahayaannya akan berubah menjadi dari padam menjadi *dimLevel(1)* dimana 1 LED pada perangkat akan menyala, dan

node lainnya tetap padam. Pada uji coba ini akan menguji tiap *node* apakah nyala LED pada tiap *node* sesuai dengan metode sistem. Pejalan kaki (*p*) ditandai dengan *cell* berwarna kuning.

Tabel 5.1 Uji nyala LED pada tiap *node* ketika ada 1 pejalan kaki terdeteksi.

Posisi Node / Posisi Pejalan Kaki	*Node 1*	*Node 2*	*Node 3*	*Node4*
Node 1	*p*			
Node 2		*p*		
Node 3			*p*	
Node 4				*p*

5.1.3 Uji Coba Lebih Dari Satu Pejalan Kaki Terdeteksi Tiap Node

Pada sistem penerangan jalan adaptif, jika lebih dari satu *node* mendeteksi adanya pejalan kaki, maka sistem akan mengatur tingkat pencahayaan pada tiap *node* $S_{n+2} > S_n > S_{n-2}$ dari tiap *node* yang mendeteksi pejalan kaki. Pada uji coba ini sistem akan diuji bagaimana jika ada 2, 3, atau 4 pejalan kaki yang terdeteksi oleh sensor.

Pada pengujian dua *node* lampu mendeteksi adanya pejalan kaki, pejalan kaki pertama (p_1) posisinya tetap berada pada *node* 1 sedangkan pejalan kaki kedua (p_2) berjalan mendekat dari *node* 4 menuju *node 2*. Pejalan kaki pertama (p_1) ditandai dengan *cell* berwarna kuning dan pejalan kaki kedua (p_2) ditandai dengan *cell* berwarna hijau, ditunjukkan pada Tabel 5.2.

Tabel 5.2 Uji nyala LED ketika 2 *node* mendeteksi pejalan kaki

Node 1	*Node 2*	*Node 3*	*Node4*
p_1			p_2
p_1		p_2	
p_1	p_2		

Pada pengujian tiga *node* lampu mendeteksi adanya pejalan kaki, pejalan kaki pertama (p_1) posisinya tetap berada pada *node 1*,

pejalan kaki kedua (p_2) berjalan mendekat dari *node* 3 menuju *node 2, dan* pejalan kaki ketiga (p_3) berjalan mendekat dari *node* 4 menuju *node* 3. Pejalan kaki pertama (p_1) ditandai dengan *cell* berwarna kuning dan pejalan kaki kedua (p_2) ditandai dengan *cell* berwarna hijau,dan dan pejalan kaki ketiga (p_3) ditandai dengan *cell* berwarna ungu, ditunjukkan pada Tabel 5.1.

Tabel 5.3 Uji nyala LED ketika 3 *node* mendeteksi pejalan kaki.

Node 1	*Node 2*	*Node 3*	*Node4*
p_1		p_2	p_3
p_1	p_2		p_3
p_1	p_2	p_3	

Pada pengujian empat *node* lampu mendeteksi adanya pejalan kaki, pejalan kaki pertama (p_1) berada pada *node 1*, pejalan kaki kedua (p_2) berada pada *node* 2, pejalan kaki ketiga (p_3) berada pada *node* 3 , dan pejalan kaki keempat (p_4) berada pada *node* 4. Pejalan kaki pertama (p_1) ditandai dengan *cell* berwarna kuning dan pejalan kaki kedua (p_2) ditandai dengan *cell* berwarna hijau, dan dan pejalan kaki ketiga (p_3) ditandai dengan *cell* berwarna ungu, dan pejalan kaki keempat (p_4) ditandai dengan *cell* berwarna jingga ditunjukkan pada Tabel 5.1.

Tabel 5.4 Uji nyala LED ketika 4 *node* mendeteksi pejalan kaki.

Node 1	*Node 2*	*Node 3*	*Node4*
p_1	p_2	p_3	p_4

5.1.4 Uji Coba Delaycounter

Pada sistem penerangan jalan adaptif, jika pejalan kaki berada diantara dua *node* namum di luar jangkauan sensor pada dua node tersebut, sistem memberikan delay waktu sebesar ± 17 detik sampai pejalan kaki terdeteksi pada *node* tetangga. Pada ujicoba ini akan diuji jalannya *delaycounter* dan fungsi *lampOnByDelay*.

Pengujian yang dilakukan untuk ujicoba *delaycounter* teknisnya hampir sama dengan uji coba satu pejalan kaki terdeteksi

pada tiap *node* seperti yang ditunjukkan pada Tabel 5.1 dan uji coba dari satu pejalan kaki terdeteksi pada setiap *node* seperti yang ditunjukkan pada Tabel 5.2, Tabel 5.3, dan Tabel 5.4 . Namun diantara *node* awal S_n dan *node* tetangganya S_{n-1} atau S_{n+1} pejalalan kaki berada di *blank zone* selama <17 detik dan >17 detik, untuk menguji apakah *node-node* tingkat pencahayaan lampu tetap menyala seperti yang dibutuhkan dalam masa *delaycounter*.

5.1.5 Hasil Dan Evaluasi Uji Coba Fungsionalitas

Pada pengujian pertama dilakukan pengujian terhadap satu pejalan kaki terdeteksi tiap *node*, hasil yang diperoleh ditunjukkan pada Tabel 5.5.

Tabel 5.5 Uji nyala LED ketika 1 *node* mendeteksi pejalan kaki

Posisi Node / Posisi Pejalan Kaki	*Node 1*	*Node 2*	*Node 3*	*Node4*
Node 1	*3*	*2*	*1*	*0*
Node 2	*2*	*3*	*2*	*1*
Node 3	*1*	*2*	*3*	*2*
Node 4	*0*	*1*	*2*	*3*

Pada percobaan tersebut menunjukkan bahwa ketika ada satu pejalan kaki yang terdeteksi pada tiap *node* sistem bisa merubah tingkat pencahayaan lampu yang dibutuhkan. Pada *node* S_n LED yang menyala adalah 3, pada S_{n+1} dan S_{n-1} LED yang menyala adalah 2, pada *node* S_{n+2} dan S_{n-2} LED yang menyala adalah 1 dan pada *node* $S_{n-2} < S_n > S_{n+2}$ LED padam. Pada Gambar 5.2, Gambar 5.3, Gambar 5.4, dan Gambar 5.5 lingkaran merah menandakan pejalan kaki yang disimulasikan dengan menggunakan gerakan tangan pada sensor PIR.

Gambar 5.2 Satu pejalan kaki terdeteksi pada *node* 1

Gambar 5.3 Satu pejalan kaki terdeteksi pada *node* 2

Gambar 5.4 Satu pejalan kaki terdeteksi pada *node* 3

Gambar 5.5 Satu pejalan kaki terdeteksi pada *node* 4

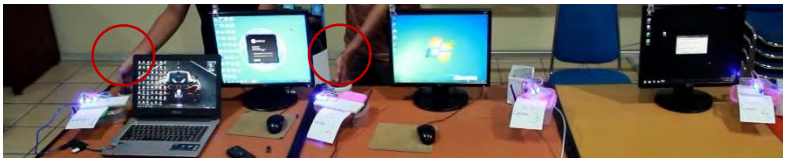
Pada pengujian kedua dilakukan pengujian apabila ada dua *node* pada sistem mendeteksi pejalan kaki, hasil yang diperoleh ditunjukkan pada Tabel 5.6.

Tabel 5.6 Uji nyala LED ketika 2 *node* mendeteksi pejalan kaki

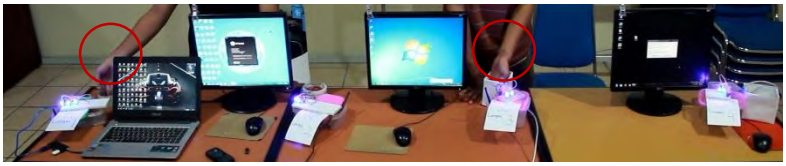
Node 1	*Node 2*	*Node 3*	*Node4*
3	2	2	*3*
3	2	*3*	2
3	*3*	2	1

Pada percobaan tersebut menunjukkan bahwa ketika ada dua pejalan kaki yang terdeteksi pada *node* sistem pengaturan pencahayaan pada tiap *node* berhasil dilakukan. *Node* ke S_{n+2} *atau* S_{n-2} dari masing-masing *node* yang mendeteksi manusia bisa mengatur tingkat pencahayaan dengan 1 LED saja yang dihidupkann sesuai dengan *command* dari masing-masing *node* yang mendeteksi manusia. *Node* ke S_{n+1} dari p_1 yang seharusnya 2 LED menyala, ketika menerima *command* yang ditujukan untuk *node* S_{n-2} dari p_2 dimana hanya 1 LED saja yang menyala berhasil

diatur LED yang menyala mengikuti *command* dari *node* terdekat yaitu 2.



Gambar 5.6 Dua pejalan kaki terdeteksi pada *node* 1 dan *node* 2



Gambar 5.7 Dua pejalan kaki terdeteksi pada *node* 1 dan *node* 3

Gambar 5.8 Dua pejalan kaki terdeteksi pada *node* 1 dan *node* 4

Pada pengujian ketiga dimana ada tiga *node* pada sistem yang mendeteksi pejalan kaki, hasil yang diperoleh ditunjukkan pada Tabel 5.7.

Tabel 5.7 Uji nyala LED ketika 3 *node* mendeteksi pejalan kaki

Node 1	*Node 2*	*Node 3*	*Node4*
3	2	*3*	*3*
3	*3*	2	3
3	*3*	*3*	2

Pada percobaan tersebut menunjukkan ketika ada tiga *node* yang mendeteksi adanya pejalan kaki, pengaturan tingkat pencahayaan lampu adaptif berhasil dijalankan. *Node* ke S_{n+1} dari

p_1 yang seharusnya 2 LED menyala, ketika menerima *command* yang ditujukan untuk *node* S_{n-2} dari p_2 dimana hanya 1 LED saja yang menyala berhasil diatur LED yang menyala mengikuti *command* dari *node* terdekat yaitu 2 hal tersebut berlaku pula pada *node* yang lainnya.

Gambar 5.9 Tiga pejalan kaki terdeteksi pada *node* 1, *node* 2 dan *node* 4

Gambar 5.10 Tiga pejalan kaki terdeteksi pada *node* 1, *node* 3 dan *node* 4

Pada pengujian keempat dimana ada empat *node* pada sistem yang mendeteksi pejalan kaki, hasil yang diperoleh ditunjukkan pada Tabel 5.8.

Tabel 5.8 Uji nyala LED ketika 4 *node* mendeteksi pejalan kaki

Node 1	*Node 2*	*Node 3*	*Node4*
3	*3*	*3*	*3*

Pada percobaan tersebut menunjukkan ketika ada lima *node* yang mendeteksi adanya pejalan kaki, pengaturan tingkat pencahayaan lampu adaptif berhasil dijalankan. Karena semua *node* yang mendeteksi keberadaan pejalan kaki maka seluruh *node* menjalankan fungsi *lampOnBySensor* sehingga jumlah LED yang menyala adalah 3.

Pada percobaan-percobaan yang dilakukan diatas fungsionalitas sistem pengaturan tingkat pencahayaan adaptif *multi pedestrian detector* telah berhasil dilakukan.

5.2 Uji Coba Performa

Pada sistem ini pengiriman data *single* hop dan pengiriman data *multi hop* sangat berperan penting, oleh karena itu perlu dilakukan uji coba performa pada bagian tersebut. Uji coba performa dilakukan untuk mengukur akurasi pengiriman data ketika *node* mendeteksi pejalan kaki pada *multi hop* antar *node* untuk mengirimkan data. Pada sistem dibutuhkan kecepatan pengiriman secepat mungkin kurang dari satu detik untuk pengiriman data, karena apabila ada *node* yang menerima lebih dari satu *command* yang berbeda, bisa mengatur mana *command* yang dijalankan.

5.2.1 Uji Coba Single Hop

Pada uji coba pengiriman data *single hop*, masing-masing *node* S_n akan mengirim data ke *node* yang bersebelahan langsung dengannya S_{n+1} dan S_{n-1} selama 1 menit. Akurasi data dihitung dari presentasi jumlah paket yang seharusnya berhasil dikirim dan diterima dibanding yang gagal dikirim dan diterima.

5.2.2 Uji Coba *Multi Hop*

Pada uji coba pengiriman data *multi hop*, masing-masing *node* S_n akan mengirim data ke *node* S_{n+2} dan S_{n-2} selama 1 menit. Akurasi data dihitung dari presentasi jumlah paket yang seharusnya berhasil dikirim dan diterima dibanding yang gagal dikirm dan diterima.

5.2.3 Uji Coba *Single Hop* dengan Jarak Tertentu

Pada uji coba ini, dilakukan ujicoba pengiriman data *single hop* satu arah. Masing-masing *node* S_n akan mengirim data ke *node* yang bersebelahan langsung dengannya S_{n+1} selama 2 menit. Akurasi data dihitung dari presentasi jumlah paket yang seharusnya

berhasil dikirim dan diterima dibanding yang gagal dikirm dan diterima.

5.2.4 Hasil Dan Evaluasi Uji Coba Performa

Pada hasil uji coba performa dengan jarak antar *node* sebesar 5 meter dan pengiriman data diberi jeda waktu 1 detik. Pengujian pertama adalah pengujian bagaimana akurasi pengiriman data *single hop* hasil yang diperoleh ditunjukkan pada Tabel 5.9.

Tabel 5.9 Uji performa *single hop*

Detik	6	12	18	24	30	36	42	48	54	60
Gagal	0	1	0	0	2	0	1	0	0	2

Pada hasil ujicoba pada Tabel 5.11, menunjukkan dari 60 pengiriman data terdapat 6 pengiriman dan penerimaan data yang gagal dilakukan, sehingga menunjukkan akurasi dari pengiriman data single hop dari sistem adalah 90%.

Pada pengujian kedua, pengujian akurasi pengiriman dan penerimaan data, diuji bagaimana pengiriman data *multi hop node* S_n akan mengirim data ke *node* S_{n+2} dan S_{n-2}, hasil yang diperoleh ditunjukkan pada Tabel 5.10.

Tabel 5.10 Uji performa *multi hop*

Detik	6	12	18	24	30	36	42	48	54	60
Gagal	1	1	2	1	2	2	2	1	1	2

Pada hasil uji coba pada Tabel 5.10, menunjukkan dari 60 pengiriman data terdapat 15 pengiriman dan penerimaan data yang gagal dilakukan, sehingga menunjukkan akurasi dari pengiriman data *multi hop* S_n mengirim data ke *node* S_{n+2} dan S_{n-2}, dari sistem adalah 75%.

Pada pengujian ketiga, pengujian akurasi pengiriman dan penerimaan data, diuji bagaimana pengiriman data *single hop node* S_n satu arah akan mengirim data ke *node* S_{n+1} dengan variasi jarak 10 m, 20 m, dan 30 m. Dari percobaan pengiriman data *single hop* dengan variasi jarak tersebut hasil yang diperoleh ditunjukkan pada Tabel 5.11.

Tabel 5.11 Uji performa *multi hop*

Jarak(m)	*Terkirim*	*Diterima*	*n-loop*	*Gagal*	%
10	123	123	125	2	98%
20	125	125	123	2	98%
30	122	122	125	3	97%

Pada hasil uji coba pengujian ketiga *n-loop* adalah jumlah *loop* pada fungsi *loop* pada Arduino. Presentsai keberhasilan ditunjukkan dengan perbandingan data yang berhasil dan diterima dengan jumlah loop yang berjalan, hal ini dilakukan karena seharusnya jumlah penerimaan dan pengiriman data sesuai dengan jumlah *loop* yang berjalan. Apabila jumlahnya tidak sama maka terjadi permasalahan terhadap penerimaan dan pengiriman data antar *node*.

Pada uji coba pengiriman data dengan jarak 10 m hanya terjadi 2 kali kegagalan penerimaan dan pengiriman dengan presentasi keberhasilan sebesar 98%. Pada uji coba 20 m ada dua *loop* yang mana menerima dua kali data dalam satu loop sehingga menyebabkan jumlah loop menjadi 2 lebih sedikit dari jumlah pengiriman dan penerimaan data. Hasil yang diperoleh pada percobaan dengan jarak 20 m menunjukkan presentasi keberhasilan sebesar 98%. Pada percobaan dengan jarak sebesar 30 m hanya terjadi 3 kali kegagalan penerimaan dan pengiriman dengan prsesntasi keberhasilan sebesar 97%.

Dari hasil uji coba performa diatas menunjukkan pada pengujian *single hop* memiliki akurasi pengiriman yang sangat baik dan pada *multi hop* memiliki akurasi pengiriman yang cukup baik untuk digunakan pada sistem penerangan jalan adaptif *multi pedestrian detector*.

LAMPIRAN

```
#include <RF24Network.h>
#include <RF24Network_config.h>
#include <Sync.h>
#include <nRF24L01.h>
#include <printf.h>
#include <SPI.h>
#include <RF24.h>
#include <RF24_config.h>


RF24 radio(7,8); //radio pin
int pirPin = 9;     //the digital pin connected to
the PIR sensor's output
int ledPin = 4;
int ledPin2 = 5;
int ledPin3 = 6;

const uint16_t this_node = 01; //this node address

const uint16_t other_node1 = 011; //address for
leftside node

const uint16_t other_node2 = 00; ////address for
rightside node

const unsigned long interval=25; //packet sent
schedule
const unsigned long delayctr=17000;// delay counter
unsigned long last_sent; //when last packet sent
unsigned long packets_sent; //how many packet sent

unsigned long currenttime =0;
unsigned long start =0;
unsigned long current =0;

int statedelay =0;
//flag for last dimming level
int last = 0;
int lastfromsource = 0;
```

```
int levellight = 0;
//flag is there any packet for the node
int statepacket = 0;
int statepir = 0;
//address source

#define DATARATE RF24_2MBPS

//data sent to leftside node
struct payload_light1
{
  uint16_t source;
  uint16_t light1;

  int pirstatus;
  int hop;
};

//data sent to rightside node
struct payload_light2
{
  uint16_t source;
  uint16_t light2;

  int pirstatus;
  int hop;
};

struct payload_light3
{
  uint16_t source;
  uint16_t light2;

  int pirstatus;
  int hop;
};

struct payload_light4
{
  uint16_t source;
  uint16_t light2;

  int pirstatus;
  int hop;
```

```
};

struct payload_light5
{
  uint16_t source;
  uint16_t light2;

  int pirstatus;
  int hop;
};

/////////////////////////////
//VARS
//the time we give the sensor to calibrate (10-60
secs according to the datasheet)
int calibrationTime = 30;

//the time when the sensor outputs a low impulse
long unsigned int lowIn;
RF24Network network(radio);
//the amount of milliseconds the sensor has to be
low
//before we assume all motion has stopped
long unsigned int pause = 5000;

boolean lockLow = true;
boolean takeLowTime;

/////////////////////////////
//SETUP
void setup(void)
{
  Serial.begin(57600);
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
  digitalWrite(pirPin, LOW);

  //give the sensor some time to calibrate
  Serial.print("calibrating sensor ");
    for(int i = 0; i < calibrationTime; i++)
    {
      Serial.print(".");
```

```
      delay(400);
    }
    Serial.println(" done");
    Serial.println("SENSOR ACTIVE");

    SPI.begin();
    radio.begin();
    network.begin(90,this_node);
}


void dimlevelact(int level)
{
  if(level == 3)
  {
    Serial.println("lampu 3");
    digitalWrite(ledPin, HIGH);
    digitalWrite(ledPin2, HIGH);
    digitalWrite(ledPin3, HIGH);
  }
  if(level == 2)
  {
    Serial.println("lampu 2");
    digitalWrite(ledPin, HIGH);
    digitalWrite(ledPin2, HIGH);
    digitalWrite(ledPin3, LOW);
  }
  else if(level == 1)
  {
    Serial.println("lampu 1");
    digitalWrite(ledPin, HIGH);
    digitalWrite(ledPin2, LOW);
    digitalWrite(ledPin3, LOW);
  }
  else if(level == 0)
  {
    Serial.println("lampu mati");
    digitalWrite(ledPin, LOW);
    digitalWrite(ledPin2, LOW);
    digitalWrite(ledPin3, LOW);
  }
}

void dimlevel(int level)
```

```
{
  if(statepacket == 1)
  {
    if(level == 2)
    {
      dimlevelact(2);
      last = 2;
      Serial.println("level 2");
    }
    if(level == 1)
    {
      if(last == 2)
      {
        dimlevelact(2);
        Serial.println("level 2  dari 1");
      }
      else if(last == 1)
      {
        dimlevelact(1);
        Serial.println("level 1 dari 1");
      }
      last = 1;
    }
    else last=0;
  }
  else
  {
    Serial.println("level 0");
    dimlevelact(0);
  }
}

void lampOnBySensor(int statesensor, int hopnum)
{
   //network.update();
   Serial.println("lamponbysensor running.");
   dimlevelact(hopnum);   //the led visualizes the
sensors output pin state
   Serial.print("ini jumlah hop tersisa: ");
   Serial.print(hopnum);
   int lompat =0 ;
   int lompats = 0;
   lompat = hopnum;
   hopnum = hopnum - 1;
```

```
   Serial.print("ini jumlah hop tersisa: ");
   Serial.println(hopnum);
   unsigned long now (millis());
   unsigned long cekwaktu = 0;
   cekwaktu = now-last_sent;
   Serial.println(cekwaktu);
   if(now - last_sent >= interval)
   {
      last_sent = now;
      Serial.print("Sending       payload       to
othernode1... ");
      Serial.println(other_node1);
      payload_light1          payload1          =
{this_node,other_node1,statesensor,hopnum};
      Serial.print("Sending       payload       to
othernode2... ");
      Serial.println(other_node2);
      payload_light2          payload2          =
{this_node,other_node2,statesensor,hopnum};

      RF24NetworkHeader header1(other_node1);
        RF24NetworkHeader header2(other_node2);

      bool ok1 = network.write(header1,&payload1,
sizeof(payload1));
      if(ok1)
        Serial.println("othernode1 succes.");
      else
        Serial.println("data    sent    fail    on
othernode1.");
      bool ok2 = network.write(header2,&payload2,
sizeof(payload2));
      if(ok2)
        Serial.println("othernode2 succes");
      else
        Serial.println("data    sent    fail    on
othernode2.");
   }
}

void lampOnByNeighbour()
{
  network.update();
  Serial.println("lamponbyneighbour running.");
```

```
  if(network.available())
  {
    while(network.available())
    {
        RF24NetworkHeader header; //if so, get the
data and config the lamp
        payload_light3 payload3;
        network.read(header,&payload3,
sizeof(payload3));

        if(payload3.light2 == this_node)
        {
          statepacket = 1;
          levellight = payload3.hop;
          Serial.print("levelight awal: ");
          Serial.println(levellight);
        }
        else
        {
          statepacket = 0;
          levellight = 0;
        }
        uint16_t toadress = 0;
        if(payload3.source == other_node2)
        {
          toadress = other_node1;
        }
        else if(payload3.source == other_node1)
        {
          toadress = other_node2;
        }
        Serial.print("levelight: ");
        Serial.println(levellight);
        Serial.print("ini alamat tujuan: ");
        Serial.println(toadress);
        Serial.println("---------------------");
        dimlevel(levellight);

        int hops = 0;
        hops = payload3.hop;

        if(hops >0 && hops<=3)
        {
          hops= hops-1;
```

```
          payload_light4         payload4         =
{payload3.source,toadress,0,hops};
          RF24NetworkHeader header3(toadress);
          bool                ok3                =
network.write(header3,&payload4,
sizeof(payload4));
          if(ok3)
            Serial.println("data  sent  to  second
hop.");
          else
            Serial.println("failed  sent  data  to
second hop.");
        }
        else
        {
          Serial.print("this is the last node");
        }
    }
  }
  else
  {
    Serial.println("ga ada data masuk");
    statepacket = 0;
    dimlevelact(0);
    last = 0;
    levellight = 0;
  }
}

int lampOnByDelay()
{
  network.update();
  Serial.println("lamponbydelay running.");
  int lodStat = 0;
  if(currenttime<=start)
  {
    currenttime = currenttime+100;
  }
  unsigned long ctrDly  = currenttime-start;
  if(ctrDly == 4294967295)
  {
    ctrDly = 1;
  }
  Serial.println(currenttime);
```

```
  Serial.println(start);
  Serial.println(ctrDly);
  if(ctrDly>=0 && ctrDly<=17000)
  {
    if(network.available())
    {
      RF24NetworkHeader header; //if so, get the
data and config the lamp
      payload_light5 payload5;
      network.read(header,&payload5,
sizeof(payload5));

      if(payload5.pirstatus == 1)
      {
        Serial.println("   delay   counter   mati,
disebelah ada orang");
        lampOnByNeighbour();
        lodStat = 0;
      }
      else if(payload5.pirstatus == 0)
      {
        Serial.println("tetangga   ga   ada   orang,
delay counter");
        lampOnBySensor(0,3);
        lodStat=1;
      }
    }
    else
    {
      Serial.println("ga ada data masuk waktu delay
counter");
      lampOnBySensor(0,3);
      lodStat = 1;
    }
  }
  else
  {
    Serial.println("diluar delay counter");
    lodStat = 0;
  }
  return lodStat;
}
```

```
int ceksensor()
{
  int sensorstate = 0;
  if(digitalRead(pirPin) == HIGH)
  {
    if(lockLow)
     {
       //makes sure we wait for a transition to LOW
before any further output is made:
       lockLow = false;
       Serial.println("---");
       Serial.print("motion detected at ");
       Serial.print(millis()/1000);
       Serial.println(" sec");
       delay(50);
     }
   takeLowTime = true;
    sensorstate = 1;
  }

  else if(digitalRead(pirPin) == LOW)
  {
    if(takeLowTime)
    {
      lowIn = millis();          //save the time of
the transition from high to LOW
      takeLowTime = false;        //make sure this
is only done at the start of a LOW phase
    }
    sensorstate = 0;
     //if the sensor is low for more than the given
pause,
     //we assume that no more motion is going to
happen
     if(!lockLow && millis() - lowIn > pause)
     {
       //makes sure this block of code is only
executed again after
       //a new motion sequence has been detected
       lockLow = true;
       Serial.print("motion    ended    at    ");
//output
       Serial.print((millis() - pause)/1000);
       Serial.println(" sec");
```

```
        delay(50);
      }

  }
  return sensorstate;
}

////////////////////////////
//LOOP
void loop(void)
{
  network.update();
  currenttime = millis();
  Serial.println(currenttime);
  Serial.println("starting loop");
  int bos = ceksensor();
  Serial.println(bos);
  if(ceksensor()==1 && statedelay==0 && (statepir
== 0 || statepir == 1 ))
  {
    Serial.println("1 0 0 1.");
    lampOnBySensor(1,3);
    statepir = 1;
  }
  if(ceksensor()==0   &&   statepir   ==   1   &&
statedelay==0)
  {
    Serial.println("0 1 0 .");
    start = millis();
    Serial.println(start);
    statedelay=1;
  }
  if(ceksensor()==0   &&   statepir   ==   1   &&
statedelay==1)
  {
    Serial.println("0 1 1 .");
    if(lampOnByDelay()==1)
    {
      Serial.println("delay 1.");
      statedelay=1;
    }
    else
    {
      Serial.println("delay 0.");
```

```
      statedelay=0;
      statepir=0;
      start =0;
    }
  }
  if(ceksensor()==1    &&    statepir    ==    1    &&
statedelay==1)
  {
    Serial.println("1 1 1 .");
    statedelay=0;
    statepir=0;
    start =0;
  }
  if(ceksensor()==0    &&    statepir    ==    0    &&
statedelay==0)
  {
    Serial.println("0 0 0 .");
    lampOnByNeighbour();
  }
  delay(1000);
}
```

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, terdapat pula saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1. Kesimpulan

Dalam proses pengerjaan Tugas Akhir mulai dari tahap analisis, desain, implementasi, hingga pengujian didapatkan kesimpulan sebagai berikut:

1. Apabila terdapat 1 *node* pada sistem yang mendeteksi adanya pejalan kaki maka tingkat pencahayaan lampu pada tiap nodenya akan diatur, *node* S_n lampu akan menyala pada tingkat pencahayaan 100%, pada *node* S_{n-1} dan S_{n+1} lampu akan menyala dengan tingkat pencahayaan sebesar 67% dan pada *node* S_{n-2} dan S_{n+2} tingkat pencahayaan akan diatur sebesar 33%.
2. Apabila terdapat lebih dari 1 *node* mendeteksi adanya pejalan kaki, tiap *node* yang mendeteksi akan mengatur sendiri, namun pada *node-node* yang tidak mendeteksi pejalan kaki namun menerima *command* yang berbeda tingkat pencahayaannya akan mengikuti *command* tingkat pencahayaan terbesar.
3. Apabila pejalan kaki berada pada area *blank zone* pada sistem, sistem akan memberikan *delay* pada *node* lampu agar tetap menyala selama 17 detik, hingga pejalan kaki tiba pada *node* lampu berikutnya atau keluar dari sistem.
4. Tingkat keakurasian sistem pada *single hop* dengan dua arah pengirman adalah sebesar 90% dan pengiriman data *multi hop* dengan satu arah pengiriman adalah 75%.

5. Tingkat keakurasian pengiriman data pada sistem pada *single hop* satu arah dengan variasi jarak 10 m adalah 98%, 20 m adalah 98 % dan 30 m adalah 97%.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang, berdasarkan pada hasil perancangan, implementasi dan uji coba yang telah dilakukan.

1. Pada penelitian selanjutnya bisa menggunakan sensor untuk mendeteksi tidak hanya gerakan manusia, namun juga gerakan kendaraaan umum.
2. Menggunakan sebuah *master node* yang bisa dipasang di ujung sistem untuk mengatur dan mengkoleksi berapa banyak pejalan kaki yang melewati sistem.
3. Tingkat pencahayaan pada tiap *node* diatur juga berdasarkan jumlah pejalan kaki yang ada pada lokasi *node* lampu

DAFTAR PUSTAKA

[1] International Energy Agency. Light's labour's lost: policies for energy-efficient lighting. France: IEA; 2006

[2] Northeast Group, LLC. Global LED and smart street lighting: market forecast (2014–2025). Washington: Northeast Group, LLC; 2014.

[3] Müllner R, Riener A. An energy efficient pedestrian aware smart street lighting system. Int J Pervasive Comput Commun 2011;7(2):147–61.

[4] Lau SP, Merrett GV, White NM, Weddell AS. Traffic-Aware Street Lighting Scheme for Smart Cities Using Autonomous Networked Sensors. Computers and Electrical Engineering 45 (2015) 192–207

[5] Ronald V. Clark, Improving Street Lighting to Reduce Crime in Residential Area, Problem-Oriented Guides for Polic, Washington, 2008.

[6] "Rekayasa Lalu Lintas/Penerangan Jalan" [Online]. Tersedia: https://id.wikibooks.org/wiki/Rekayasa_Lalu_Lintas/Penerangan_jalan [diakses 17 Desember 2015].

[7] Pyroelectric Infrared Radial Sensor Manual Model:D203B. PIR Sensor CO.,LTD.

[8] Pratama, I Putu Eka., Suakanto, Sinung. 2015. Wireless Sensor Network. Bandung: Informatika.

[9] Sulaiman, A. (2012, February 1). "ARDUINO: Mikrokontroler bagi Pemula hingga Mahir" [Online]. Tersedia:http://buletin.balaielektronika.com/?p=163 [diakses 17 Desember 2015]

[10] Semiconductor,NORDIC.2008. nRF24L01+ Single Chip 2.4 GHZ Transciever Product Specificaion v1.0. Norwegia: NORDIC Semiconductor.

[11] Kadir, Abdul.2014. Buku Pintar Pemrograman Arduino. Yogyakarta: Mediakom.
[12] "HC-SR501 PIR Motion Detector Datasheet" [Online]. Tersedia: https://www.mysensors.org/motion/HCSR501.pdf [diakses 2 Februari 2016]
[13] Ada, Lady. "PIR Motion Sensor" [Online]. Tersedia: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor [diakses 2 Februari 2016]
[14] Badan Standarisasi Nasional.2008. Standart Penerangan Jalan SNI-7391-2008. Jakarta: BSN.
[15] "Single Arm street Light Pole" [Online]. Tersedia: http://gxtxsb.net/wproducts_content-67529.html [diakses 21 Juli 20016]

BIODATA PENULIS

Penulis lahir di Jember, 28 Maret 1994, merupakan anak pertama dari 2 bersaudara. Dalam perjalanan hidupnya penulis pernah menempuh pendidikan di SDN Kebonsari 2 Jember, SMPN 2 Jember, SMAN 1 Jember dan S1 Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS) pada rumpun Komputasi berbasis Jaringan (KBJ).

Selama menjadi penulis pernah bergelut di beberapa organisasi seperti HMTC ITS, KMI ITS, dan Himpunan Mahasiswa Jember Surabaya. Selain itu penulis juga pernah kerja praktik di PT Telekomunikasi Indonesia Malang dan *freelance* di Aksamedia Group.

Penulis dapat dihubungi melalui alamat surel m.iqbaltanjung@gmail.com