SISTEM *MONITORING* LINGKUNGAN RUMAH CERDAS BERBASIS *FOG COMPUTING* DAN NRF24L01

SKRIPSI

Untuk memenuhi sebagai persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Dwiki Nuridhuha
NIM: 165150307111054

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PENGESAHAN

SISTEM *MONITORING* LINGKUNGAN RUMAH CERDAS
BERBASIS *FOG COMPUTING* DAN NRF24L01

SKRIPSI

Diajukan untuk memenuhi sebagai persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Dwiki Nuridhuha
NIM: 165150307111054

Skripsi ini telah diuji dan dinyatakan lulus pada
9 April 2020

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing 1	Dosen Pembimbing 2
Mochammad Hannats Hanafi Ichsan, S.ST, M.T.	Rizal Maulana, S.T, M.T, M.Sc.
NIP: 19881229 201903 1 010	NIK: 201607 891009 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 24 Februari 2020

Dwiki Nuridhuha

NIM: 165150307111054

PRAKATA

Puji syukur kehadiran Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "SIstem Monitoring Lingkungan Rumah Cerdas Berbasis *Fog Computing* Dan NRF24L01" ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Kepada kedua orang tua penulis yang senantiasa memberikan doa serta dukungan selama penulis menempuh pendidikan di Malang.
2. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
3. Bapak Dahnial Syauqy, S.T., M.T., M.Sc. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya Malang.
4. Bapak Mochammad Hannats Hanafi Ichsan, S.ST, M.T., Rizal Maulana, S.T, M.T, M.Sc selaku dosen pembimbing skripsi yang telah dengan sabar dan tekun dalam membimbing jalannya proses skripsi.
5. Adji, Yullius, Arikuloli, Gunawan, Hanifa, Azzam, Satria, Dhiza, Ardi, Labib, Dika, Senna, Yuni, Crisandolin, Chikam, Sabit, Hanif kecil dan besar yang telah membantu dalam proses pengerjaan skripsi.
6. LSO POROS, K-RISMA, ROBOTIIK, BCC dan Lab Dron yang telah memberikan semangat dan tempat dalam pengerjaan skripsi.
7. Seluruh teman-teman geng cimori, atom, kunam, aliansi akatsuki dan Teknik Komputer yang selalu mendukung dan memotivasi penulis dalam mengerjakan skripsi ini.
8. Seluruh pihak yang tidak dapat disebutkan satu persatu yang telah berperan dalam penyelesaian skripsi ini.

Penulis menyadari bahwa laporan skripsi ini masih belum sempurna, sehingga adanya kritik serta saran membangun sangat penulis harapkan agar dapat lebih baik dalam penulisan tugas akhir lainnya. Dengan ini penulis harap agar skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi.

Malang, 24 Februari 2020

Penulis

dwikind@student.ub.ac.id

ABSTRAK

Dwiki Nuridhuha, SIstem *Monitoring* Lingkungan Rumah Cerdas Berbasis *Fog Computing* Dan nRF24L01

Pembimbing: Mochammad Hannats Hanafi Ichsan, S.ST, M.T dan Rizal Maulana, S.T, M.T, M.Sc

Rumah cerdas merupakan penerapan teknologi yang penting bagi manusia karena dapat menyediakan kenyamanan, akan tetapi banyaknya perangkat cerdas pada rumah cerdas dapat mempengaruhi pertukaran data setiap perangkat yang akan berakibat menurunkan kinerja transmisi dan distribusi data ke ranah internet atau ke perangkat cerdas laninya. Berdasarkan permasalahan tersebut penulis menerapkan *fog computing* untuk melakukan pengolahan data diranah sumber data untuk mengatasi kendala dari internet atau *cloud* dan nRF24l01 untuk pertukaran data antar perangkat cerdas pada rumah cerdas. Perangkat cerdas pada penelitian ini terdiri dari dua *node sensor* untuk melakukan *sensing* lingkungan, satu *node sink* untuk meneruskan data *node sensor* ke *node fog,* dan satu *node fog* untuk melakukan pengumpulan data dan *backup* data ke *cloud*, kontrol manual, dan visualisasi data. Fokus penelitian ini mengukur kinerja *delay*, *throughput*, *packet loss*, dan RTT(*round trip-time*) pada modul komunikasi nRF24l01 dalam lingkungan *fog computing* dan mengukur variasi waktu pengiriman data *sensing* ke *cloud*. Pengujian kinerja pada penelitian ini yaitu dengan menaruh setiap *node* dengan jarak kelipan 4 meter dengan kondisi lingkungan tanpa anda halangan dan dengan ada penghalang. Hasil yang didapat dari penelitian ini bahwa jarak terjauh pengiriman data sebesar 44 meter jika tanpa penghalang, akan tetapi jika ada penghalang jarak pengiriman terbesar 28 meter, sedangkan hasil pengujian kinerja QoS(*Quality of Service*) pada kondisi lingkungan tanpa penghalang dan ada penghalang besar kecilnya nilai *delay*, *throughput*, RTT(*round trip-time*) tidak terpengaruhi dengan jarak pengiriman, akan tetapi nilai *packet loss* akan naik jika jarak pengiriman data antar node semakin jauh. Sedangkan pada variasi waktu pengiriman data ke *cloud* tidak mengalami banyak perubahan waktu pengiriman baik ada penghalang atau pun tidak ada penghalang dan jarak pengiriman data antar *node*.

Kata kunci: *fog computing, r*umah cerdas, nRF24l01*, cloud, quality of service*

ABSTRACT

Dwiki Nuridhuha, ***System monitoring environment smart home based fog computing and nRF24L01***

Supervisors: Mochammad Hannats Hanafi Ichsan, S.ST, M.T dan Rizal Maulana, S.T, M.T, M.Sc

Smart home is an important technology implementation for human beings because it can provide comfort, but many smart home devices can affect the data exchange of each device that will result in the performance of data transmission and distribution of the Internet or to its mobile device. Based on the problem the authors apply fog computing to perform data processing in the data source to overcome the constraints of the Internet or cloud and nRF24l01 to exchange data between smart devices on the smart home. The smart device in the study consisted of two sensor nodes for environmental sensing, one sink node for forwarding the sensor node data to the Fog node, and one fog node for data collection and data backup to the cloud, manual control, and data visualization. The focus of this research measures the performance of delay, throughput, packet loss, and RTT (round trip-time) in the communication module nRF24l01 in a fog computing environment and measures the variation in time of sensing data delivery to the cloud. Performance testing in this research is by placing each node with a spaced distance of 4 meters with environmental conditions without obstruction and with a barrier. The results obtained from this research that the farthest data delivery amount is 44 meters if without barrier, but if there is a barrier delivery distance is 28 meters largest, while the results of performance testing QoS (Quality of Service) in environmental conditions without obstructions and there is a large barrier of the value of delay, throughput, RTT (round trip-time) is not affected by the shipping distance, but the value of the packet loss will increase if the data transfer distance between nodes further. While the time variation of data transfer to the cloud does not undergo a lot of changes in the delivery time either there is barrier or no barrier and data transmission distance between nodes.

Keywords: *fog computing, smart home, nrf24l01, cloud, quality of service*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Rumah merupakan salah satu kebutuhan utama yang harus di penuhi oleh semua orang dan rumah yang ideal harus menyediakan kenyamanan, dan keamanan, untuk mewujudkan kenyamanan dan keamanan bagi penghuni rumah dapat menerapkan teknologi rumah cerdas (Siregar R, 2019). Penerapan teknologi rumah cerdas membuat peralatan rumah seperti pendingin rumah, TV, sistem audio dan video, pencahayaan, pemanas, komputer, dan peralatan - peralatan rumah lain mampu saling bertukar informasi satu sama lain dan dapat dikendalikan sesuai kebutuhan penghuni rumah, adanya teknologi rumah cerdas memungkinkan pemilik rumah dapat mengamati atau mengontrol kenyamanan dan keamanan dalam rumah (Tian A, 2018). Peralatan-peralatan rumah sangat penting, oleh sebab itu diperlukan sensor dan perangkat keras yang disebut juga perangkat cerdas untuk memantau setiap peralatan rumah (Ichsan M.H.H, 2018). Banyaknya perangkat *internet of things* pada rumah cerdas akan berpengaruh pada kinerja pertukaran data dari setiap perangkat, hal ini di sebabkan setiap perangkat mengirimkan data terus menerus yang mengakibatkan trafik menuju internet menjadi ramai, dan di waktu yang bersamaan penghuni rumah juga mengakses internet, hal ini dapat memengaruhi konsumsi energi, transmisi dan distribusi data, yang akan berakibat menurunkan kinerja setiap perangkat pada rumah cerdas padahal penerapan rumah cerdas harus dapat melakukan integrasi pada perangkat lain dan memberikan informasi ke penghuni rumah secara akurat dan cepat (EditorialTeam, 2019). Penerapan teknologi *fog computing* dan jaringan sensor nirkabel dapat membantu memberikan informasi ke pengguna walaupun kondisi jaringan internet sedang tidak baik secara cepat.

Monitoring kondisi lingkungan rumah cerdas dengan menerapkan teknologi *fog computing* dan jaringan sensor nirkabel telah dilakukan oleh (Francisco, et al., 2018) dalam penelitiannya *fog computing* digunakan sebagai analisis data, penyimpanan data, kendali perangkat pada rumah cerdas, dan *cloud gateway*. Akan tetapi pada penelitian tersebut tidak ada mekanisme menampilkan data pada lingkungan lokal. Sedangkan jaringan sensor nirkabel menggunakan protokol komunikasi mqtt dan menggunakan Wi-Fi sebagai media transmisi data. Protokol mqtt sendiri memiliki kelebihan *high-latency* dan *low bandwidth* yang cocok diterapkan pada *device* dengan *resources* terbatas (Rakhman, M.H, 2018) dan media transmisi *Wi-Fi* memiliki kelebihan *transfer rate* yang tinggi (Pratama I.P.A.E, 2015). Selain mempunyai kelebihan baik *protokol* mqtt dan *Wi-Fi* memiliki kekurangan yaitu ketika broker atau *router Wi-Fi* berhenti maka setiap perangkat tidak dapat mengirim data sampai broker atau *router Wi-Fi* bekerja kembali (Rakhman, M.H, 2018). Oleh sebab itu dibutuhkan mekanisme *monitoring* lingkungan rumah cerdas pada jaringan lokal dan dibutuhkan metode pertukaran data lain yang dapat membuat media transmisi sendiri dalam melakukan pertukaran data seperti modul komunikasi nRF24l01.

Berdasarkan kebutuhan akan ketersediaan *monitoring* lingkungan rumah cerdas pada jaringan lokal dalam menangani *cloud* atau akses internet yang terbatas pada penelitian ini akan merancang sistem rumah cerdas berbasis *fog computing* sebagai penanganan akses internet atau *cloud* dengan mengolah data pada lingkungan sumber data berasal dan modul komunikasi nRF24l01 sebagai media pertukaran data antar perangkat. Pada penelitian ini sistem *monitoring* lingkungan pada rumah cerdas ini terdiri dari dua *node sensor,* seperangkat *node sink*, dan seperangkat *node fog.* Komponen utama *node sensor* terdiri atas mikrokontroller, sensor, dan modul komunikasi. *Node sensor* ditempatkan pada bagian - bagian rumah secara spesifik, seperti menempatkan *node sensor* yang terpasang sensor suhu pada ruangan tamu untuk *monitoring* kondisi suhu pada ruangan tamu. Secara periodik *node sensor* akan mengambil data dari hasil pengamatan pada setiap ruangan, kemudian data dari pengamatan sensor dikirim ke *node sink* melalui modul komunikasi. Untuk komponen utama *node sink* terdiri dari mikrokontroller, dan modul komunikasi. *Node sink* bertugas untuk melakukan pengumpulan data dari setiap *node sensor* dan meneruskan ke *node fog*. Sedangkan *node fog* memiliki komponen utama yang terdiri dari *mini PC*, dan modul komunikasi. *Node fog* akan menerima data dari *node sink* yang kemudian data akan ditampilkan atau diolah atau dianalisis, supaya *bandwidth* pengiriman ke *cloud* lebih efisien dan memperingan kinerja pada *cloud*, selain itu pada jaringan lokal dapat memperkecil *latency*, dan dapat meningkatkan *Quality of Servis* pada jaringan lokal yang tidak stabil (OpenFog, 2015). Modul komunikasi yang digunakan pada penelitian ini adalah modul komunikasi nrf24l0l yang dapat mengkonsumsi arus dengan daya rendah hal ini disebabkan nrf24lo1 menggunakan *Enhanced Shockburst Protocol* (Ichsan M.H.H, 2019). Selain itu nr24l01 dapat digunakan pada *topology mesh* di jaringan *sensor* nirkabel dengan menggunakan RF24Mesh supaya setiap *node sensor* dapat berkomunikasi secara langsung untuk sinkronisasi waktu (Afif T, 2019). Aplikasi web pada penelitian ini dibuat dengan menggunakan flask, flask merupakan *framework* web yang ringan sehingga cocok digunakan untuk *monitoring* pada perangkat tertanam berbasis *mini pc* (Rovai M, 2018). Pada media penyimpanan yang digunakan pada penelitian ini menggunakan SQLite untuk database lokal di *node fog*, SQLite sendiri merupakan database SQL yang ringan yang cocok diterapkan pada perangkat yang memiliki spesifikasi rendah (ScienceProg, 2019). Sedangkan untuk database di cloud menggunakan mongoDB yang merupakan database NoSQL yang menawarkan availability dan skalabilitas supaya kinerja dari perangkat IoT menjadi lebih baik (Ibrahim D.M, 2018).

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang tersebut, maka terdapat beberapa rumusan masalah yang diangkat, antara lain:

1. Bagaimana mengimplementasikan *fog computing* pada rumah cerdas?
2. Bagaimana Kinerja *Delay* pengiriman nRF24l01 dalam mengirimkan data dari *node sensor* sampai *node fog?*

3. Bagaimana Kinerja RTT(*round trip-time*) pengiriman nRF24l01 dalam mengirimkan data dari *node sensor* sampai *node fog?*
4. Bagaimana Kinerja *Throughput* pengiriman nRF24l01 dalam mengirimkan data dari *node sensor* sampai *node fog?*
5. Bagaimana Kinerja *Packet Loss* pengiriman nRF24l01 dalam mengirimkan data dari *node sensor* sampai *node fog*?
6. Bagaimana variasi waktu pengiriman data dari *node sensor* sampai ke *cloud*?

1.3 Tujuan

Berdasarkan pada rumusan masalah, maka tujuan dari penelitian ini adalah sebagai berikut:

1. Dapat mengimplementasikan *fog computing* pada rumah cerdas.
2. Dapat mengukur kinerja *Delay* pengiriman nRF24l01 dalam mengirimkan data dari *node sensor* sampai *node fog*.
3. Dapat mengukur kinerja RTT (*round trip-time*) pengiriman nRF24l01 dalam mengirimkan data dari *node sensor* sampai *node fog*.
4. Dapat mengukur kinerja *Throughput* pengiriman nRF24l01 dalam mengirimkan data dari *node sensor* sampai *node fog*.
5. Dapat mengukur kinerja *Packet Loss* pengiriman nRF24l01 dalam mengirimkan data dari *node sensor* sampai *node fog*.
6. Dapat mengukur variasi waktu pengiriman data dari *node sensor* sampai ke *cloud*.

1.4 Manfaat

Harapan dari penelitian *fog computing* untuk rumah cerdas dengan menggunakan nRF24l01 ini, menghasilkan manfaat sebagai berikut:

1. Dapat menjadi rujukan untuk penelitian mengenai penerapan fog computing, rumah cerdas, dan juga nRF24l01.
2. Dapat menghasilkan suatu sistem yang dapat membantu penghuni atau pemilik rumah untuk mempermudah melakukan *monitoring* kondisi rumah melalui perangkat *smart phone*, laptop, atau perangkat komputer lain melalui aplikasi website walaupun dalam kondisi jaringan internet tidak stabil.

1.5 Batasan Masalah

Untuk memfokuskan penelitian ini supaya tidak meluas, maka diberikan batasan masalah. Adapun batasan masalah yang diberikan antara lain:

1. Jumlah perangkat yang digunakan adalah 4, 1 untuk *node sink*, 1 untuk *node fog* dan 2 untuk *node sensor.*

2. Sistem belum menerapkan efisiensi konsumsi daya.
3. Sistem belum menerapkan sumber daya cadangan.
4. Aktuator pada sistem berupa indikator LED yang terpasang pada *node fog.*
5. Sistem menggunakan mikrokontroller Arduino Uno untuk *node sensor* dan *node sink*, sedangkan untuk *node fog* menggunakan *Single Board Computer* raspberry pi, dan modul komunikasi adalah nrf24l01.
6. Analisis data pada *node fog* masih menggunakan *if else* dengan nilai threshold yang sudah ditentukan.
7. Pengiriman data antar node masih secara *flooding.*
8. *Sensing* lingkungan rumah hanya suhu dan intensitas cahaya.
9. *Sensor* suhu dan cahaya pada penelitian ini, tidak diuji keakuratan *sensing*.

1.6 Sistematika Pembahasan

Berikut adalah sistematika penulisan dalam penelitian ini:

BAB 1 : PENDAHULUAN

Bab 1 pendahuluan membahas latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika pembahasan.

BAB 2 : LANDASAN KEPUSTAKAAN

Bab 2 landasan kepustakaan membahas terkait penelitian dan teori yang telah ada sebelumnya untuk menunjang kebutuhan pada penelitian.

BAB 3 : METODOLOGI

Bab 3 metodologi membahas tentang metode atau perancangan alur kerja yang digunakan dalam penelitian.

BAB 4 : REKAYASA KEBUTUHAN

Bab 4 rekayasa kebutuhan membahas mengenai kebutuhan sistem yang diperlukan dalam penelitian yang meliputi kebutuhan fungsional dan kebutuhan non-fungsional.

BAB 5 : PERANCANGAN DAN IMPLEMENTASI

Bab 5 Perancangan membahas tentang proses merancang sistem sesuai dengan kebutuhan yang telah dijelaskan, mulai dari skematik alat setiap node, bagaimana cara komunikasi antar node, dan cara *node fog* dapat menampilkan data, menyimpan data, analisis data dan melakukan kontrol terhadap sistem.

Bab 5 implementasi membahas tentang menerapkan perancangan sistem yang telah dibuat. Pada bagian ini akan ditunjukkan Gambar, dan kode program dari dokumentasi hasil implementasi sistem.

BAB 6 : PENGUJIAN

Bab 6 Pengujian membahas tentang pengujian kinerja sistem yang sudah dibuat. Pada bagian ini akan dijelaskan mengenai grafik, dan tabel hasil pengujian sekaligus penarikan kesimpulan dari hasil pengujian.

BAB 7 : PENUTUP

Bab 7 Penutup membahas tentang kesimpulan dan juga saran dari hasil perancangan, implementasi, dan pengujian yang telah dilakukan.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Isi Landasan Kepustakaan

Dalam pengerjaan penelitian penulis melakukan tinjauan pustaka. Berikut penelitian yang terkait dengan jaringan sensor nirkabel, *fog computing*, rumah cerdas, nRF24l01 bisa dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terkait

No	Nama Penulis (Tahun), Judul Penelitian	Persamaan	Perbedaan	
			Penelitian Terdahulu	Rencana Penelitian
1.	Fauzan Masykur dan Fiqiana Prasetiyowati. Aplikasi Rumah Pintar (*Smart Home*) Pengendali Peralatan Elektronik Rumah Tangga Berbasis Web	Menggunakan *mini PC* Raspberry Pi sebagai web server dan sistem kontrol pada rumah cerdas	Sistem hanya melakukan kontrol terhadap Lampu, AC, dan TV melalui aplikasi web berbasis PHP dan data kontrol tidak disimpan pada database	Menggunakan mini PC Raspberry Pi sebagai *node fog*. *Node fog* akan menyimpan data *sensing* dan hasil kontrol, menampilkan hasil *sensing*, dan melakukan analisis dari hasil *sensing*
2.	oleh Xinlong Wen dan Yunliang Wang, *Design of Smart Home Environment Monitoring System Based on Raspberry Pi*	Sistem menerapkan jaringan sensor nirkabel dan melakukan *monitoring* suhu dan intensitas cahaya pada lingkungan rumah cerdas yang ditampilkan melalui aplikasi web lokal atau *localhost*	Sistem menggunakan teknologi EnOcean mulai dari sensor sampai komunikasi antar *node*	Sistem menggunakan modul sensor DHT11 sebagai *sensing* suhu, modul sensor LDR untuk *sensing* intensitas cahaya, dan modul komunikasi nRF24l01 sebagai komunikasi antar *node*

3.	Francisco Javier Ferrández Pastor dkk, *Deployment of IoT Edge and Fog Computing Technologies to Develop Smart Building Services*	Sistem menggunakan fog computing pada rumah cerdas dan Raspberry Pi digunakan sebagai *node fog*	*Node fog* digunakan untuk Machine Learning dengan menggunakan K-NN untuk analisis data dan untuk *monitoring* diserahkan ke cloud tingspeak dengan komunikasi MQTT	Kecerdasan buatan pada *node fog* masih menggunakan IF ELSE (fake AI) dan untuk *monitoring* dilakukan pada *node fog* juga
4	Peng Zhang Lianke dkk, *Wireless network design and implementation in smart home*	Sistem menggunakan jaringan sensor nirkabel untuk *monitoring* suhu dan intensitas cahaya pada rumah berbasis web dan modul komunikasi menggunakan nRF24l01	Aplikasi web berada pada cloud atau hosting dan semua node menggunakan berbasis teknologi ARM mikrokontroller	Aplikasi web berada pada *node fog* atau masih berada lokal area, tiga *node* menggunakan *mikrokontroller* arduino, dan satu *node* menggunakan *mini pc* Raspberry Pi
5	Anand Paul dkk, *Fog Computing-Based IoT for Health Monitoring System*	Menerapkan *fog computing* dan jaringan *sensor* nirkabel untuk *monitoring*	Menggunakan iFogSim dan cloudSim untuk simulasi *monitoring* kesehatan	Sistem menggunakan perangkat nyata untuk *monitoring* rumah cerdas

2.2 Dasar Teori

Subbab dasar teori membahas teori yang digunakan pada penyusunan penelitian yang diperlukan.

2.2.1 Rumah Cerdas

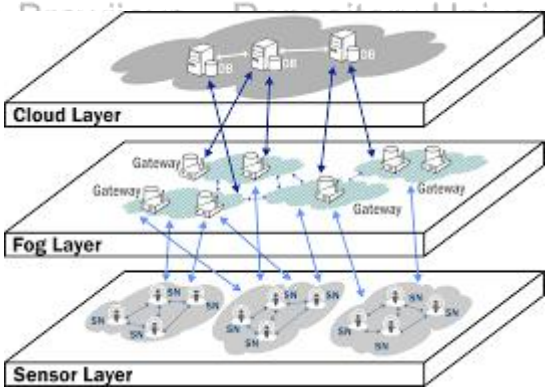
Rumah Cerdas adalah istilah tempat tinggal yang memiliki peralatan - peralatan rumah yang berkomunikasi satu sama lain untuk menyediakan rumah

yang nyaman, aman, dan efisien energi. Manfaat adanya rumah cerdas salah satunya adalah mudah digunakan, semenjak adanya perangkat portabel seperti *smart phone* atau tablet atau laptop untuk melakukan kontrol terhadap perabotan rumah menjadi mudah seperti menyalakan lampu atau AC, manfaat lain adanya rumah cerdas adalah dapat meningkatkan ramah lingkungan dan energi pada rumah, dengan menerapkan sistem rumah cerdas pemilik rumah dapat memantau, mencatat, dan mengkontrol kondisi rumah sesuai kebutuhan dari pemilik rumah (Ardi Tian, 2018). Gambar 2.1 menunjukkan ilustrasi penerapan rumah cerdas.

Gambar 2.1 Teknologi Rumah Cerdas
Sumber: homeautotechs (2018)

2.2.2 Fog Computing

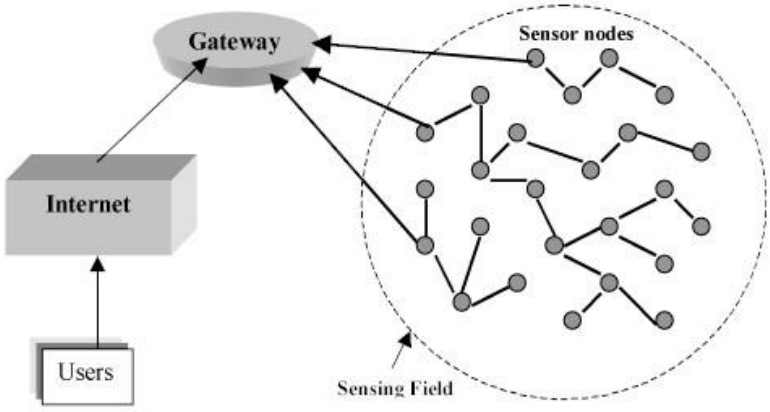
Fog Computing adalah arsitektur *system-level horizontal* yang mendistribusikan sumber daya dan layanan komputasi, penyimpanan, kontrol, dan jaringan di mana saja di sepanjang *continuum* dari *Cloud to Things*. Dengan memperluas cloud agar lebih dekat dengan hal-hal yang menghasilkan dan bertindak pada data IoT, *Fog* memungkinkan komputasi dilakukan dalam jarak yang dekat dengan sensor, yang nantinya menghasilkan *bandwidth* jaringan yang lebih efisien dan solusi IoT yang lebih fungsional dan lebih efisien. *Fog Computing* juga menawarkan bisnis yang lebih besar melalui pengolahan data yang dalam dan lebih cepat, peningkatan keamanan, dan biaya pengoperasian yang lebih rendah (OpenFog, 2019). Sesuai pada Gambar 2.2 menunjukkan pada *layer internet of things,* fog computing menjembatani sensor layer dengan cloud layer, di mana sensor layer bertugas untuk melakukan *sensing* lingkungan dan cloud layer untuk dilakukan analisis,dan penyimpanan data.



Gambar 2.2 Layer *Internet of Things*
***Sumber:* semanticscholar (2017)**

2.2.3 Jaringan Sensor Nirkabel (JSN)

Jaringan sensor nirkabel(JSN) atau biasanya disebut *wireless sensor Network* merupakan jaringan komputer terdistribusi (*Distributed Computing Network*) yang memanfaatkan sejumlah *node sensor* berukuran kecil, dikembangkan dan di konfigurasi dalam sekala besar untuk membantu *sensing* terhadap lingkungan sekitar, memanfaatkan parameter pengukuran berupa temperatur, tekanan, gerakan, atau entitas lain yang diketahui oleh manusia (Alzaid H, 2011).



Gambar 2.3 Arsitektur Jaringan Sensor Nirkabel

Sumber: telcominded (2017)

Sebagai salah satu teknologi di dalam jaringan komputer, Jaringan sensor nirkabel memiliki enam buah ciri yang membedakannya dengan jenis jaringan komputer lain. Keenam buah ciri utama tersebut meliputi adanya minimal dua buah *node sensor*, memiliki *self organizing network*(SON), adanya *self network maintenance*(SNM), pengiriman paket data secara *broadcast*, penggunaan multhi Hop routing, serta komunikasi yang dilakukan dalam jaringan yang relative dekat. (Suakanto S, 2015). Pada Gambar 2.3 menunjukkan bahwa jaringan sensor nirkabel terdiri dari banyak *node yang* saling terhubung satu sama lain, dan untuk menampilkan hasil *sensing* dari *node sensor* biasanya diperlukan akses internet yang dilewatkan melalui *gateway*.

2.2.4 Arduino Uno

Arduino Uno adalah sebuah *board* mikrokontroller berdasarkan ATmega328. Arduino Uno memiliki 14 pin digital input/output 6 pin diantaranya dapat digunakan untuk PWM dan memiliki 6 pin analog output. Pada *board* Arduino Uno terdapat sebuah USB konektor, jack power, ICSP Header, dan sebuah tombol reset (Arduino, 2019). Beberapa fitur yang ada pada Arduino Uno dapat dilihat pada Tabel 2.2.

Tabel 2.1 ringkasan fitur Arduino Uno

Mikrokontroller	ATmega328P-PU
Tegangan kerja	5 V
Tegangan input	7-12 V
Tegangan input	6(minimum) - 20 VDC(maksimum)
Pin Digital I/O	14 pin(6 untuk pin PWM)
Pin input analog	6 pin
Arus DC setiap I/O	40 mA
Memori flash	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Kecepatan Clock	16MHz

2.2.5 Raspberry Pi 3 B+

Raspberry Pi 3 B+ adalah *Single Board Computer (SBC*) yang populer karena merupakan komputer penuh yang dikemas ke dalam satu *board*. Untuk menjalankan Raspberry pi 3 B+ perlu memasang sistem operasi Raspbian atau windows IoT yang khusus untuk Raspberry, tampilan dan nuansa sistem operasi Raspbian mudah dipahami oleh setiap pengguna komputer dekstop. Pada Gambar 2.4 merupakan bentuk fisik dari Raspberry Pi 3 B+. Sistem operasi yang terus ditingkatkan mengalami perombakan grafik, dan termasuk perambah web, permainan, dan perangkat lunak lain(raspberrypi, 2018). Berikut spesifikasi dari Raspbery Pi 3 B+:

1. Input power: 5 V/2.5 A DC via micro USB konektor
2. Chipset: Broadcom BCM2837
3. CPU: 1.4 GHz Cortex-A53
4. Ethernet : 10/100 (maksimal throughput 100Mbps)
5. USB: empat USB 2.0 dengan 480Mbps data transfer
6. Storage: MicroSD card atau melalui USB-attached storage
7. Wireless: 802.11n Wireless LAN (Peak transmit/receive throughput of 150Mbps), Bluetooth 4.1

8. Memori : 1GB LPDDR2 SDRAM
9. Pin yang Akses: 40 general purpose input-output pins
10. Video: Full size HDMI
11. Audio: 4 pole stereo output
12. Kamera interface (CSI)

Gambar 2.4 Raspberry pi 3 B+

Sumber: raspberrypi (2018)

2.2.6 nRF24l01

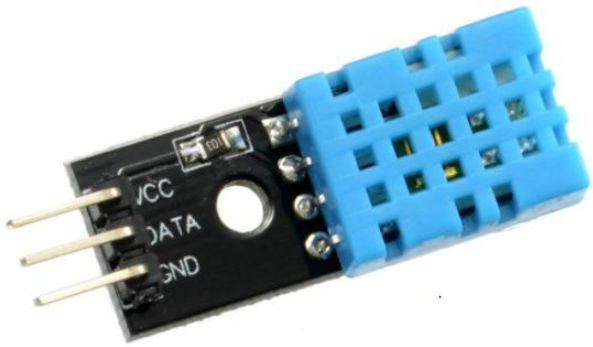
nRF24l01 adalah sebuah single chip 2.4Ghz transceiver dengan sebuah perangkat tertanam pada protokol baseband(Enchanced ShockBurst), sehinga sangat cocok untuk sumber daya rendah pada pengaplikasian wireless. Modul nRF24L01 dirancang untuk bekerja pada pita frequensi 2.400-2.4835 GHz, Untuk merancang sebuah sistem radio dengan nRF24L01, dapat menggunakan sebuah *mikrokontroller* dan beberapa external komponen pasif. nRF24L01 dapat bekerja dan di konfigurasi melalui sebuah Serial Peripheral Interface(SPI). Pemetaan register yang dapat di akses melalui SPI berisi semua konfigurasi register di dalam modul nRF24L01 dan dapat diakses semua operasi oleh mode chip. Protokol *embedded baseband engine* (*Enchanced ShockBur*t) berdasarkan paket komunikasi dan banyak didukung berbagai variasi model untuk penggunaan manual ke protokol penggunaan secara otonom. Pada perangkat internal FIFO memastikan kelancaran data *flow* antara frekuensi radio dan sistem mikrokontroller. Protokol *Enhanced Shock-Burst* mengurangi biaya beban kerja sistem dengan menangani semua operasi pada lapisan link yang berkecepatan tinggi. Radio frekuensi pada nRF24L01 menggunakan GFSK modul, sehingga pengguna pada mengkonfigurasi parameter seperti kanal frekuensi, keluaran daya dan data rate. nRF24L01 mendukung transmisi data rate sebesar 250kbps, 1 Mbps, dan 2Mbps. Penggunaan transmisi data rate yang tinggi dikombinasikan dengan dua model penghemat daya sehingga nRF24L01 sangat cocok untuk perancangan *ultra low power* (nordic semiconductor, 2006).

2.2.7 DHT11

Sensor DHT11 dapat membaca nilai suhu dan kelembaban, keluaran sinyal berupa sinya digital yang dikalibrasi. Sensor DHT11 mencakup pengukuran kelembaban tipe resistif komponen dan komponen pengukuran suhu NTC, dan menghubungkan ke kinerja tinggi. DHT 11 menawarkan kualitas yang baik, respon cepat, tidak muda terkena interference dari lingkungan, dan harga yang murah (Mouse Electronics, 2018).

Berikut fitur - fitur dari sensor DHT11 :

1. Jarak pengukuran 20-90%RH 0-50 ℃
2. Akurasi pengukuran kelembaban kurang lebih 5%RH
3. Akurasi pengukuran suhu kurang lebih 2 ℃
4. Resolution 1
5. Ada tiga pin pada DHT11 yaitu pin VCC, data, dan GND



Gambar 2.5 Sensor DHT11

Sumber: RudraNarayanG (2019)

2.2.8 Light Dependent Resistor (LDR)

Sensor LDR merupakan sensor untuk menditeksi tingkat intensitas cahaya. Pada LDR terdapat dua *cadmium sulphide* (cds) cell photoconductive dengan respon spektral mirip dengan mata manusia. Keluaran nilai LDR berupa resistan, nilai resistan akan berkurang jika tingkat intensitas cahaya tinggi. LDR dapat digunakan dalam berbagai jenis rangkaian dan aplikasi (SUNROM, 2008). Gambar 2.6 merupakan bentuk fisik dari sensor LDR.

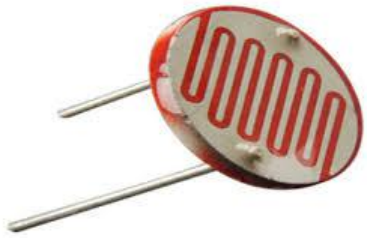
Berikut aplikasi LDR pada analog sistem:

1. Kamera kontrol
2. Auto Slide Fokus
3. Densitometer
4. Mesin *photocopy*

5. Alat uji Colometric

Dan berikut aplikasi LDR pada digital sistem:

1. Automic *Headlight Dimmer*
2. Kontrol *night light*
3. Lampu jalan raya
4. Posisi *sensor*
5. Sistem pembakan minyak



Gambar 2.6 Sensor LDR

Sumber: shemabook (2019)

2.2.9 SQLite

SQlite merupakan library perangkat lunak yang mengimplementasikan *self-contained*, *serverless*, *zero-configuration*, dan menerapkan SQL database engine. *Zero-configuration* pada SQLite memiliki makna programmer tidak perlu melakukan konfigurasi pada sistem tidak seperti database lain yang memerlukan konfigurasi pada sistem yang dibangun. SQLite adalah salah satu database dengan pertumbuhan paling cepat dalam hal popularitas. Proses pada database SQLite tidak berjalan mandiri, programmer dapat menyambungkan sesuai kebutuhan pada aplikasi. SQLite sendiri tidak memerlukan proses atau sistem *server* terpisah untuk beroperasi (*serverless*) dan database SQLite memiliki ukuran yang kecil dan ringan dengan ukuran kurang lebih 400 kb untuk terkonfigurasi atau kurang lebih 250 kb dengan fitur beberapa fitur yang dihilangkan (tutorialpoint, 2019). Pada Gambar 2.7 merupakan logo dari SQLite.

Gambar 2.7 Logo SQLite

Sumber: Mahesa Iqbal Ridwansyah (2019)

2.2.10 Flask

Flask merupakan sebuah mikro *web framework* yang yang ditulis menggunakan bahasa pemrograman python dan dikategorikan sebagai *microframework* karena flask tidak menyediakan library atau tool untuk abstraksi database layer, validasi form, atau komponen lainyang membutuhkan pihak ketiga untuk membuat suatu fungsi. Ada beberapa ekstensi yang dapat langsung digunakan oleh *microframework* flask yaitu dapat membuat pemetaan *object-relational*, penanganan upload, authetikasi , dan sebagainya (flask, 2020). Pada Gambar 2.8 merupakan tampilan logo dari Flask.

Berikut merupakan fitur yang disediakan oleh flask:

1. Menggunakan Jinja2 sebagai templating
2. Menggunakan WSG1 1.0
3. Berdasarkan *Unicode*
4. Dokumentasi yang luas
5. Kompatibel dengan Google App Engine

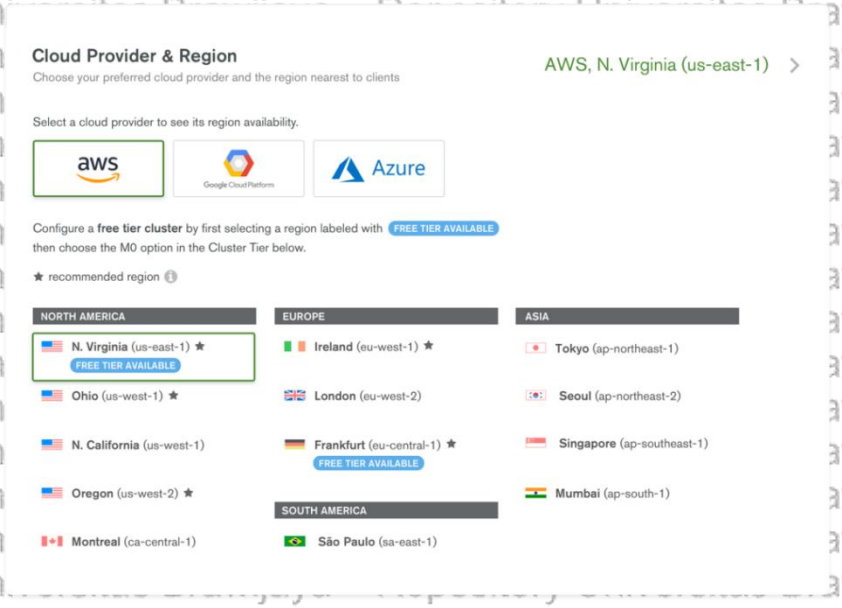
Gambar 2.8 Tampilan logo *mickroframework* flask

Sumber: Flask (2020)

2.2.11 MongoDB Atlas

MongoDB Atlas adalah sebuah *database as a service* (DAAS) untuk mongoDB dan menerapkan NoSQL database *engine* dengan format data berupa JSON. MongoDB Atlas menggunakan *storage engine* berupa *WiredTiger* yang membuat penyimpanan yang efisien karena mengimplementasikan teknologi kompresi seperti Snapp, gzip dan prefix kompresi, selain itu dapat meningkatkan skalabilitas baca dan tulis secara bersama hal ini dapat meningkatkan kinerja throughput pada database dan dapat mengoptimalkan penggunaan memori dengan menggunakan cache internal dan cache pada sistem file. Perlindungan keamanan data pada MongoDB Atlas menggunakan TLS/SSL enkripsi, authetikasi, otoritas melalui SCRAM, VPC peering pada AWS (*across availability zones*) untuk

isolasi jaringan, dan IP whitelists, dengan menerapkan teknik keamanan tersebut mongoDB Atlas memiliki keamanan yang tinggi (mongoDB, 2019). Pada Gambar 2.9 merupakan tampilan untuk memilih lokasi dari cloud database mongoDB yang akan di pakai.



Gambar 2.9 Provider *Cloud* mongoDB Atlas

Sumber: mongoDB (2019)

BAB 3 METODOLOGI

3.1 Strategi dan Rancangan Penelitian

3.1.1 Lokasi Penelitian

Pada penelitian ini lokasi penelitian dilakukan pada area lingkungan yang memiliki banyak bangunan dan mempunyai area tanpa penghalang dan ada penghalang, hal ini dikarenakan untuk menyimulasikan rumah cerdas.

3.1.2 Teknik Pengumpulan Data

Pada penelitian ini teknik pengumpulan data dilakukan dengan cara melakukan percobaan 100 kali pengambilan data di lapangan dengan menyebarkan setiap node ke beberapa titik dengan jarak yang sudah ditentukan dengan area lingkungan tanpa penghalang dan area lingkungan dengan adanya penghalang.

3.1.3 Peralatan Pendukung Yang Digunakan

Pada penelitian ini peralatan yang digunakan untuk membangun sistem yang utuh membutuhkan beberapa komponen seperti Laptop, Raspberry pi 3, Arduino Uno, sensor DHT11, sensor LDR, nRF24l01, dan kabel jumper. Laptop pada penelitian ini berfungsi untuk melakukan program terhadap Raspberry pi 3 dan juga Arduino Uno. Raspberry pi 3 pada penelitian ini berfungsi sebagai *node fog*. Arduino Uno pada penelitian ini berfungsi sebagai *node sensor* jika menggabungkan dengan sensor DHT11 dan LDR, dan sebagai *node sink* jika hanya menggabungkan dengan modul komunikasi nRF24l01. nRF24l01 berfungsi sebagai komunikasi antar node dan kabel jumper untuk menggabungkan keseluruhan komponen.

3.2 Metode atau teknik lain

3.2.1 Metodologi Penelitian

Pada sub-bab ini menjelaskan langka - langkah yang akan dilakukan dalam pengerjaan suatu penelitian. Pada penelitian ini bertipe Implementatif-pengembangan, dengan membangun sebuah sistem *monitoring* suhu dan intensitas cahaya berbasiskan *fog computing* pada lingkungan rumah cerdas dan menggunakan modul komunikasi nRF24l01 untuk komunikasi antar perangkat *node*. Tahapan penelitian ini dilakukan secara runtun sesuai dengan diagram alir pada Gambar 3.1. Sesuai pada Gambar 3.1 penelitian ini dimulai dari studi literatur, yang kemudian dilanjutkan dengan analisis kebutuhan, setelah itu dilanjutkan dengan perancangan sistem, setelah tahap perancangan selesai dilakukan implementasi yang sudah dibuat pada perancangan sistem, kemudian setelah implementasi selesai dilakukan pengujian sistem untuk melihat kinerja dari sistem yang telah dibuat, dan yang terakhir kesimpulan untuk menunjukkan hasil penarikan kesimpulan pada sistem yang telah dilakukan.

Gambar 3.1 Diagram alir metodologi penelitian

3.2.2 Studi Literature

Keperluan studi literatur untuk digunakan sebagai landasan dasar dan penguat teori yang dapat diambil dari buku, jurnal, paper, dan website. Literatur yang dipakai pada penelitian ini adalah :

1. Teori mengenai *Fog Computing*, Rumah Cerdas, dan Jaringan Sensor Nirkabel.
2. Arduino Uno, Raspberry pi 3, DHT11, LDR, nRF24L01, SQLite, mongoDB Atlas dan Flask.

3.2.3 Analisis Kebutuhan

Supaya sistem dapat berjalan sesuai dengan yang diharapkan, maka diperlukan analisis kebutuhan yang akan digunakan oleh sistem. Berikut daftar kebutuhan sistem pada penelitian ini:

Kebutuhan fungsional:

1. *Node sensor* dapat melakukan *sensing*
2. Setiap *node* dapat saling berkomunikasi
3. *Node fog* dapat melakukan analisis data hasil *sensing*
4. *Node fog* dapat menyimpan data hasil *sensing* ke database lokal SQLite
5. *Node fog* dapat mengirim data hasil *sensing* ke cloud mongoDB Atlas pada jaringan internet yang tidak stabil

6. *Node fog* dapat menampilkan data hasil *sensing* melalui aplikasi web secara real time
7. *Node fog* dapat melakukan kontrol melalui aplikasi web

Kebutuhan non-fungsional :

1. Kebutuhan Perangkat Keras
2. Kebutuhan Perangkat Lunak

3.2.4 Perancangan Sistem

Perancangan sistem pada penelitian ini berlangsung setelah analisis kebutuhan untuk penelitian sudah terpenuhi. Harapan pelaksanaan perancangan sistem pada penelitian ini adalah untuk dapat mengimplementasikan sistem dengan baik. Terdapat beberapa perancangan yang akan dibuat, berikut penjelasan poin-poin perancangan pada penelitian ini:

1. Terdapat empat *node*, dua *node* untuk *node sensor*, satu *node* untuk *node sink* atau *gateway*, dan satu *node* untuk *node fog*.
2. *Node sensor* terbagi menjadi dua yaitu pertama *node sensor* untuk melakukan *sensing* suhu dan kedua *node sensor* untuk melakukan *sensing cahaya*. Data hasil *sensing* dikirim ke *node sink* melalui modul komunikasi nRF24l01.
3. *Node sink* memiliki peran untuk meneruskan data hasil sensing dari *node sensor* ke *node fog*. Proses meneruskan data oleh *node sink* menggunakan modul komunikasi nRF24l01.
4. *Node fog* bertugas untuk melakukan analisis, menampilkan, dan menyimpan data *sensing* secara lokal, selain itu *node fog* bertugas untuk melakukan kontrol manual terhadap sistem dan melakukan penyimpanan data *sensing* ke *cloud* untuk *backup* data atau untuk di analisis lebih lanjut oleh pakar.

3.2.5 Implementasi Sistem

Penerapan implementasi sistem mengacu pada perancangan yang sudah dibuat sebelumnya yaitu perancangan *node sensor* suhu, *node sensor* cahaya, *node sink*, dan perancangan *node fog*. Pada tahap implementasi sistem seluru gagasan rancangan desain sistem baik berupa flowchart atau pun skematik alat akan diterapkan sesuai dengan tujuan perancangan sistem. Berikut tahapan implementasi sistem pada penelitian ini:

1. Implementasi arsitektur alat *node sensor* suhu, *node sensor* cahaya, *node sink*, dan *node fog*
2. Implementasi komunikasi antar *node* dengan modul komunikasi nRF24l01
3. Implementasi aplikasi web, penyimpanan data, analisis data dan kontrol sistem pada *node fog*.

3.2.6 Pengujian dan Analisis

Pada tahap pengujian sistem, akan dilakukan analisis QoS(*Quality of Service*) pada pengiriman *node sensor* ke *node sink* dari *node sink ke node fog*. QoS yang dipakai pada penelitian ini menggunakan *Delay, packet loss, throughput, RTT (round trip-time) dan* Variasi waktu pengiriman dari *node fog* ke *cloud*. Untuk mengukur nilai RTT (*round trip-time)* dengan mengirimkan paket data dari *node sensor* sampai ke *node fog* kemudian paket data tersebut dikembalikan lagi ke *node sensor*.Persamaan matematis QoS *Delay* bisa dilihat pada persamaan 3.1, persamaan matematis QoS *Packet loss* bisa dilihat pada persamaan 3.2, persamaan matematis QoS *throughput* bisa dilihat pada persamaan 3.3, dan persamaan matematis Variasi waktu pengiriman dari *node fog* ke *cloud* dengan membagi standar deviasi dengan mean sesuai dengan persamaan 3.4.

Delay = *Total Delay*/ Total paket yang di terima (3.1)

$$\textit{Packet loss} = \frac{\text{(data yang dikirim - paket data yang diterima)}}{\text{Paket data yang dikirim}} \times 100\% \quad (3.2)$$

Throughput = Besar paket data yang dikirim/ *delay* (3.3)

Coefficient of Variation = standar deviasi / mean (3.4)

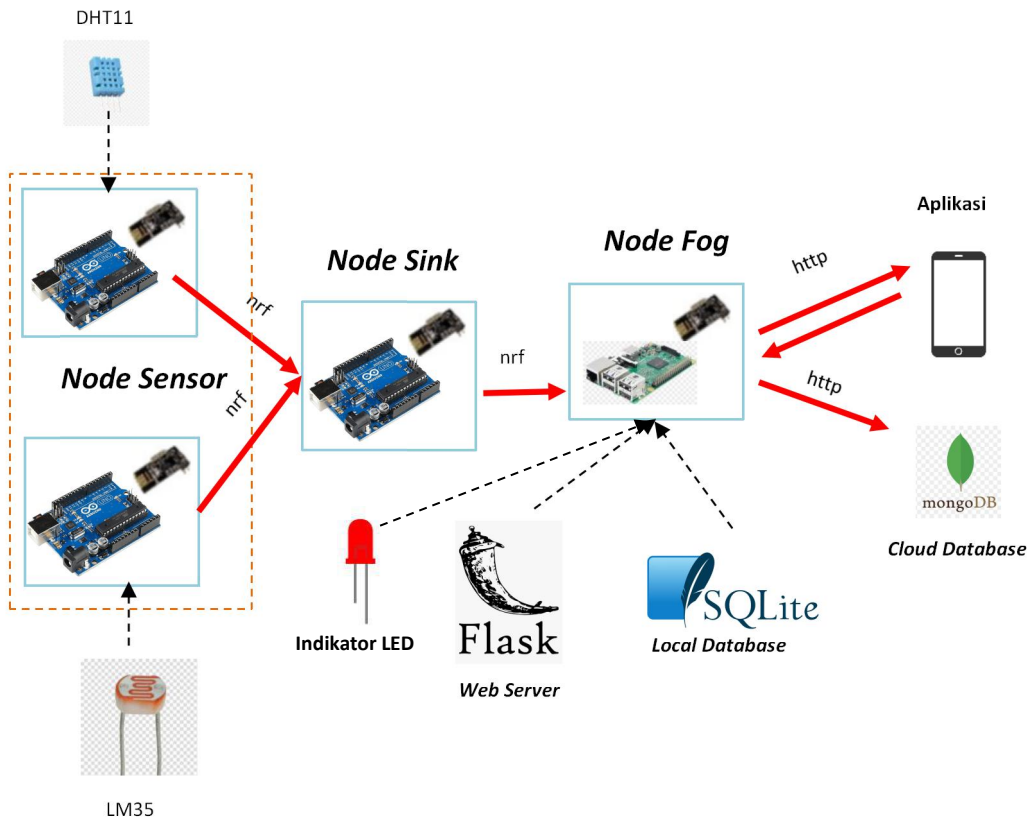
3.2.7 Kesimpulan

Pengambilan kesimpulan dan saran dapat dilakukan saat seluruh tahap metodologi penelitian sudah dilaksanakan. Hasil kesimpulan diambil berdasarkan hasil pengujian akhir implementasi serta mengacu pada rumusan masalah. Adanya kesimpulan dan saran diharapkan dapat menjadi acuan untuk pengembangan dan penelitian - penelitian yang lain.

BAB 4 REKAYASA KEBUTUHAN

4.1 Gambaran Umum Sistem

Pada penelitian ini akan dibuat sistem *monitoring* lingkungan rumah cerdas berbasis *fog computing* dan nrf24l01. Sistem ini terdiri dari empat *node* yaitu dua *node sensor*, satu *node sink*, dan satu *node fog* yang saling terhubung dengan modul komunikasi nrf24l01. *Node sensor* melakukan *sensing* suhu (DHT11) dan cahaya (LDR) hasil *sensing* dari *node sensor* akan diteruskan ke *node fog* melalui *node sink.* data hasil *sensing* yang diterima *node fog* akan dilakukan seleksi kondisi dan di simpan ke database SQLite. *Node fog* juga menjalankan web *server* (flask) untuk menampilkan data hasil *sensing* yang diambil di database SQLite melalui webSocket supaya *real time* dan tombol untuk menyalakan kipas (LED) dan lampu (LED), selain itu *node fog* akan mengirimkan data hasil *sensing* yang berada pada database SQLite ke mongoDB untuk dilakukan *backup* atau dilakukan analisis *big data* lingkungan *cloud*. Perancangan sistem dapat dilihat pada Gambar 4.1



Gambar 4.1 Skema Perancangan Sistem

4.2 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan yang harus ada pada sistem supaya sistem dapat berjalan sesuai dengan fungsinya. Kebutuhan fungsional pada penelitian ini terdiri atas kebutuhan sistem.

4.2.1 Kebutuhan Sistem

Kebutuhan sistem yang dibutuhkan dalam penelitian ini bisa dilihat pada Tabel 4.2.

Tabel 4.1 Kebutuhan Sistem

No	Kode	Kebutuhan Sistem	Penjelasan
1	KS-01	*Node sensor* dapat melakukan *sensing*	Merupakan kebutuhan sistem pada *node sensor* untuk melakukan *sensing* suhu dan intensitas cahaya lingkungan rumah. *Node sensor* pada penelitian ini terdapat dua yaitu *node sensor suhu* untuk melakukan *sensing* suhu pada lingkungan rumah dan *node sensor* cahaya untuk melakukan *sensing* intensitas cahaya pada lingkungan rumah. Modul sensor yang dipakai *node sensor* suhu adalah DHT11 dan mikrokontroller yang digunakan yaitu Arduino Uno, sedangkan pada *node sensor* cahaya modul sensor yang digunakan yaitu LDR dan mikrokontroller yang digunakan yaitu Arduino Uno.
2	KS-02	Setiap *node* dapat saling berkomunikasi melalui modul komunikasi nRF24l01	Merupakan kebutuhan sistem untuk pertukaran data antar *node*. Pada kebutuhan ini modul komunikasi nrf24l01 di pasang pada setiap *node*. Tahap awal pertukaran data dimulai pada *node sensor*, setelah *node sensor* melakukan *sensing* data tersebut akan di kirim ke *node sink* untuk kemudian di teruskan ke *node fog* untuk pengolahan data *sensing* lebih lanjut. Proses pengiriman data antar *node* akan menghasilkan nilai kembali “1” jika data pengiriman berhasil dan menghasilkan nilai kembali “0” jika data pengiriman kegagalan.

3	KS-03	*Node fog* dapat melakukan analisis data hasil *sensing.*	Merupakan kebutuhan sistem untuk melakukan analisis dari data hasil *sensing* oleh *node sensor*. Data hasil *sensing* diterima oleh *node fog*, *node fog* akan melakukan seleksi data dengan kategori suhu atau cahaya jika data hasil *sensing* yang diterima merupakan kategori suhu maka *node fog* akan melakukan penyeleksian kembali data dengan kategori suhu tersebut jika data kategori suhu tersebut lebih dari "30" maka LED berwarna merah akan menyala jika tidak LED berwarna merah akan mati. Sedangkan jika data yang diterima oleh *node fog* merupakan data dengan kategori cahaya maka data akan di bandingan dengan nilai "600", jika melebihi nilai "600" maka LED berwarna hijau akan menyala jika tidak LED berwarna hijau akan mati.
4	KS-04	*Node fog* dapat menyimpan data hasil *sensing* ke database lokal SQLite	Merupakan kebutuhan sistem pada *node fog* untuk menyimpan data hasil *sensing* ke database SQLite. Data *sensing* yang diterima oleh *node fog* melalui *node sink* terdiri atas nilai *sensing* dan kategori *sensing*. Sebelum data di masukan ke dalam database data *sensing* akan ditambahkan dengan data tanggal dan jam untuk mengetahui kapan data *sensing* tersebut di masukan ke dalam database SQLite.
5	KS-05	*Node fog* dapat mengirim data hasil *sensing* dari SQLite ke cloud mongoDB Atlas	Merupakan kebutuhan sistem pada *node fog* untuk meningkatkan keandalan sistem melalui mekanisme *backup* data *sensing*. Sistem pengiriman data ke *cloud* akan melakukan inisialisasi nilai batas atas dan batas bawah untuk menentukan range pengambilan data yang akan dikirim dari database SQLite ke *cloud* mongoDB Atlas baru kemudian sistem pengiriman data ke *cloud* akan

			melakukan koneksi dengan database mongoDB Atlas dan pengambilan data *sensing* dari database SQLite yang ada dikirim ke *cloud* mongoDB Atlas, jika terjadi kegagalan koneksi atau kegagalan pengambilan data *sensing* dari database SQLite, maka sistem pengiriman data ke *cloud* akan melakukan koneksi ulang ke mongoDB Atlas dan mengambil ulang data *sensing* dari database SQLite, jika koneksi dan pengambilan data *sensing* dari database SQLite berhasil maka data *sensing* yang telah diambil dari database akan dikirim ke cloud mongoDB Atlas jika berhasil maka batas atas dan batas bawah akan diperbarui jika mengalami kegagalan dalam pengiriman data ke cloud mongoDB Atlas maka akan dilakukan koneksi ulang ke *cloud* mongoDB Atlas dan pengambilan data ke database SQLite.
6	KS-06	*Node fog* dapat menampilkan data hasil *sensing* melalui aplikasi web secara *real time*	Merupakan kebutuhan sistem pada *node fog* untuk menampilkan data hasil *sensing* yang sudah tersimpan pada database SQLite secara real time di aplikasi web. Pada sistem pengambilan data pada database SQLite oleh aplikasi web dengan menggunakan mekanisme *webSocket*, dengan menerapkan *webSocket* pada aplikasi web jika ada penambahan data *sensing* pada database SQLite di aplikasi web akan secara *real time* akan mengalami perubahan sesuai dengan database SQLite.
7	KS-07	*Node fog* dapat melakukan kontrol melalui aplikasi web	Merupakan kebutuhan sistem pada *node fog* untuk melakukan kontrol terhadap aktuator pada sistem melalui aplikasi website. Pada sistem kontrol melalui aplikasi web, pengguna akan disediakan tampilan tombol untuk melakukan kontrol terhadap aktuator

			kipas atau lampu yang pada penelitian ini masih menggunakan indikator LED, jika pengguna menekan tombol ”nyala kipas” sistem akan menyalakan LED berwarna biru dan pengguna akan diberikan notifikasi kalau kipas sedang menyala dan tombol “nyala kipas “ akan berubah menjadi tombol “matikan kipas” jika pengguna menekan tombol “matikan kipas” pengguna akan diberikan notifikasi kalau kipas sudah mati dan tombol “matikan kipas” akan berubah menjadi tombol “nyala kipas”. Sedangkan pada kontrol terhadap lampu, jika pengguna menekan tombol ”nyala lampu” sistem akan menyalakan LED berwarna kuning dan pengguna akan diberikan notifikasi kalau lampu sedang menyala dan tombol “nyala lampu“ akan berubah menjadi tombol “matikan lampu” jika pengguna menekan tombol “matikan lampu” pengguna akan diberikan notifikasi kalau lampu sudah mati dan tombol “matikan lampu” akan berubah menjadi tombol “nyala lampu”.

4.3 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah kebutuhan supaya sistem dapat berjalan lebih baik. Kebutuhan non-fungsional pada penelitian ini terdiri atas kebutuhan perangkat keras dan kebutuhan perangkat lunak.

4.3.1 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang dibutuhkan dalam penelitian ini bisa dilihat pada Tabel 4.2.

Tabel 4.2 Kebutuhan Perangkat Keras

No	Perangkat	Keterangan
1	Arduino Uno	Pada penelitian ini Arduino Uno sebagai perangkat mikrokontroller pada *node sensor* untuk melakukan *sensing* terhadap lingkungan dan pada *node sink* untuk meneruskan data *sensing* dari *node sensor* ke *node fog*.

2	Raspberry Pi	Pada penelitian ini Raspberry Pi digunakan sebagai perangkat *node fog* untuk menerima hasil data *sensing*, mengolah data *sensing*, menampilkan data *sensing,* dan meneruskan data *sensing* ke *cloud*.
3	nRF24l01	Pada penelitian ini nRF24l01 digunakan sebagai perangkat komunikasi data antar *node*.
4	DHT11	Pada penelitian ini modul *sensor* DHT11 digunakan sebagai perangkat untuk melakukan *sensing* suhu.
5	LDR	Pada penelitian ini modul *sensor* LDR digunakan sebagai perangkat untuk melakukan *sensing* intensitas cahaya.
6	LED	Pada penelitian ini perangkat LED digunakan sebagai indikator kontrol aktuator dan indikator analisis data hasil *sensing.*

4.3.2 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang dibutuhkan dalam penelitian ini bisa dilihat pada Tabel 4.3.

Tabel 4.3 Kebutuhan Perangkat Lunak

No	Perangkat	Keterangan
1	C/C++	Pada penelitian ini bahasa pemrograman C/C++ digunakan untuk memprogram perangkat Arduino Uno dan memprogram modul komunikasi nRF24l01 pada *node fog*.
2	Python	Pada penelitian ini bahasa pemrograman python digunakan sebagai *back-end* aplikasi web dan proses pengiriman data *sensing* ke *cloud* mongoDB Atlas.
3	Javascript	Pada penelitian ini bahasa pemrograman javascript digunakan sebagai *front-end* aplikasi web
4	Arduino IDE	Pada penelitian ini Arduino IDE digunakan sebagai menulis baris kode untuk memprogram perangkat Arduino Uno.
5	Visual Studio Code	Pada penelitian ini Visual Studio Code digunakan untuk menulis baris kode untuk memprogram perangkat *node fog*.
6	MobaXterm	Pada penelitian ini MobaXtrem digunakan untuk melakukan *remote* terhadap raspberry pi dan *update* atau *download* file pada raspberry pi.

7	pyMongo	Pada penelitian ini *library* pyMongo digunakan pada *node fog* untuk dapat mengakses database mongoDB Atlas.
8	Sqlite3	Pada penelitian ini *library* Sqlite3 digunakan pada *node fog* untuk penyimpanan data secara lokal.
9	flask	Pada penelitian ini *library* flask digunakan pada *node fog* untuk membuat web server.
10	enventlet	Pada penelitian ini *library* envenlet digunakan pada *node fog* untuk mengirimkan data *sensing* pada database SQLite ke web server secara *realtime*
11	RF24Network	Pada penelitian ini *library* RF24Network digunakan pada semua node supaya setiap node dapat mengakses dan melakukan pengiriman data menggunakan modul komunikasi nRF24l01.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

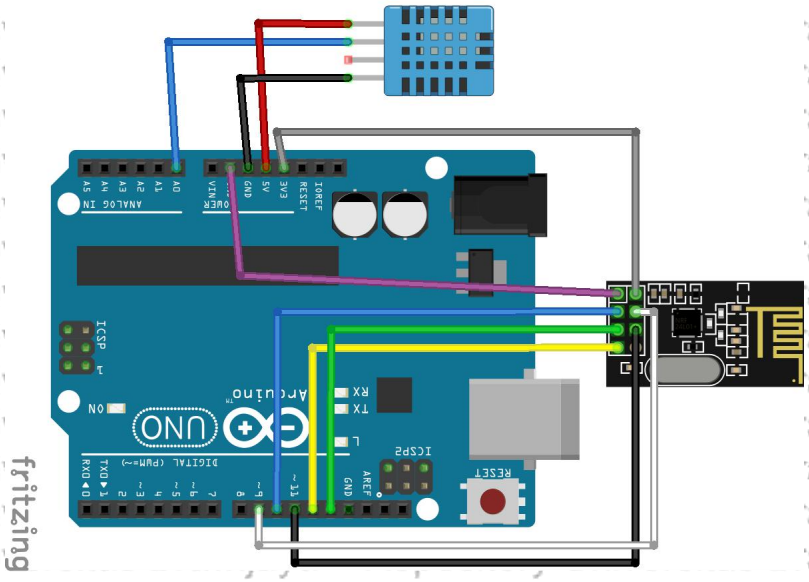
5.1 Perancangan

Pada bab perancangan, membahas mengenai rancangan sistem dari perancangan *node sensor suhu*, perancangan *node sensor cahaya*, perancangan *node sink*, dan perancangan *node fog* yang terbagi menjadi dua bagian yaitu perancangan perangkat keras dan perancangan perangkat lunak.

5.1.1 Perancangan Perangkat Keras

5.1.1.1 Perancangan Skema *Node Sensor Suhu*

Pada perancangan skema *node sensor* suhu dapat dilihat pada Gambar 5.1, dalam perancangan arsitektur *node sensor* suhu terbagi menjadi dua bagian yaitu perancangan Arduino Uno dengan modul sensor DHT11 dan perancangan Arduino Uno dengan modul komunikasi nRF24l0. Bagian perancangan Arduino Uno dengan modul sensor DHT11, sensor DHT11 sebagai komponen *input* yang melakukan *sensing* suhu pada lingkungan rumah. Modul sensor DHT11 dihubungkan dengan *board* Arduino Uno menggunakan kabel jumper sesuai pada Tabel 5.1. Pin VCC pada modul sensor DHT11 dihubungkan dengan pin 5v pada *board* Arduino Uno supaya modul sensor DHT11 dapat bekerja melakukan *sensing* sesuai harapan, dan Pin Data pada modul sensor DHT11 dihubungkan dengan pin AO pada *board* Arduino Uno hal ini disebabkan *output* dari DHT11 merupakan data analog. Bagian perancangan Arduino Uno dengan modul komunikasi nRF24l01, modul nRF24l01 sebagai komponen untuk melakukan komunikasi dengan *node sink*. Modul nRF24l01 dihubungkan dengan *board* Arduino Uno menggunakan kabel jumper sesuai dengan Tabel 5.2.



Gambar 5.1 Skema *Node Sensor* Suhu

Tabel 5.1 Hubungan antara pin nRF24l01 dan Arduino Uno

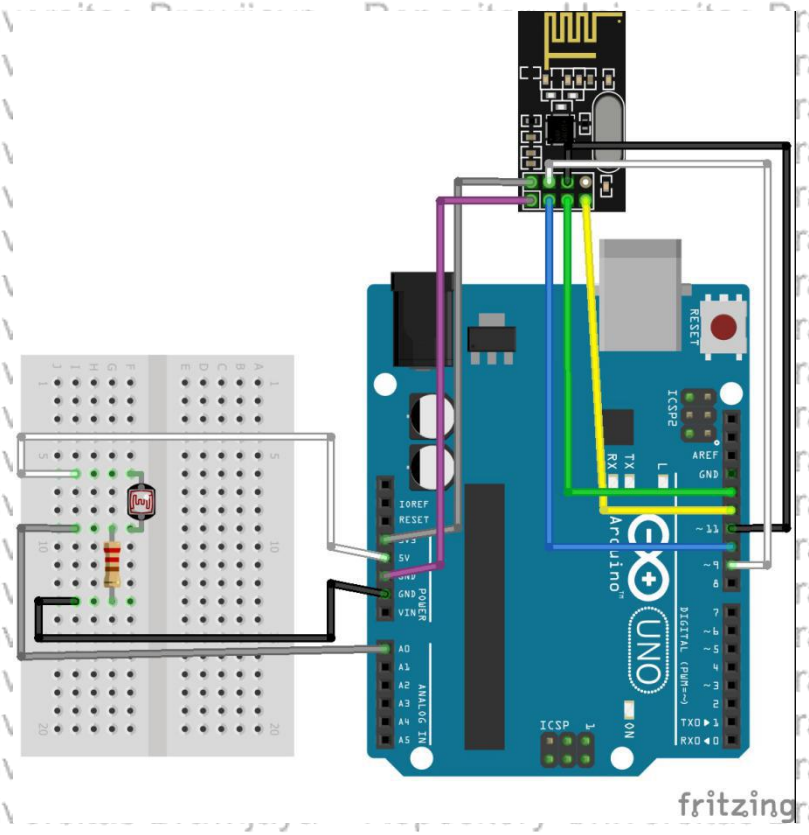
Pin nRF24l01	Pin Arduino Uno
VCC	3.3V
GND	GND
CS	9
CE	10
SCK	13
MOSI	11
MISO	12

Tabel 5.2 Hubungan antara pin DHT11 dan Arduino Uno

Pin DHT11	Pin Arduino Uno
VCC	5V
GND	GND
Data	A0
NC	-

5.1.1.2 Perancangan Skema *Node Sensor* Cahaya

Pada perancangan Skema *node sensor* cahaya dapat dilihat pada Gambar 5.2, dalam perancangan arsitektur *node sensor* cahaya terbagi menjadi dua bagian yaitu perancangan Arduino Uno dengan modul sensor LDR dan perancangan Arduino Uno dengan modul komunikasi nRF24l0. Bagian perancangan Arduino Uno dengan modul sensor LDR, sensor LDR sebagai komponen *input* yang melakukan *sensing* cahaya pada lingkungan rumah. Modul sensor LDR dihubungkan dengan *board* Arduino Uno dan resistor menggunakan kabel jumper sesuai pada Tabel 5.3. Salah satu kaki pada modul sensor LDR dihubungkan dengan pin 5v pada *board* Arduino Uno supaya modul sensor LDR dapat bekerja melakukan *sensing* sesuai harapan, dan kaki satunya pada modul sensor LDR dihubungkan dengan pin AO pada *board* Arduino Uno dan GND pada *board* Arduino Uno dihubungkan dengan resistor dan bagian kaki resistor yang lain dihubungkan dengan A0 satu jalur dengan sensor LDR hal ini dilakukan untuk dapat mengubah keluaran sensor LDR dari hambatan menjadi tegangan yang berupa data analog. Bagian perancangan Arduino Uno dengan modul komunikasi nRF24l01, modul nRF24l01 sebagai komponen untuk melakukan komunikasi dengan *node sink*. Modul nRF24l01 dihubungkan dengan *board* Arduino Uno menggunakan kabel jumper sesuai dengan Tabel 5.4.



Gambar 5.2 Skema *Node Sensor* Cahaya

Tabel 5.3 Hubungan antara pin nRF24l01 dan Arduino Uno

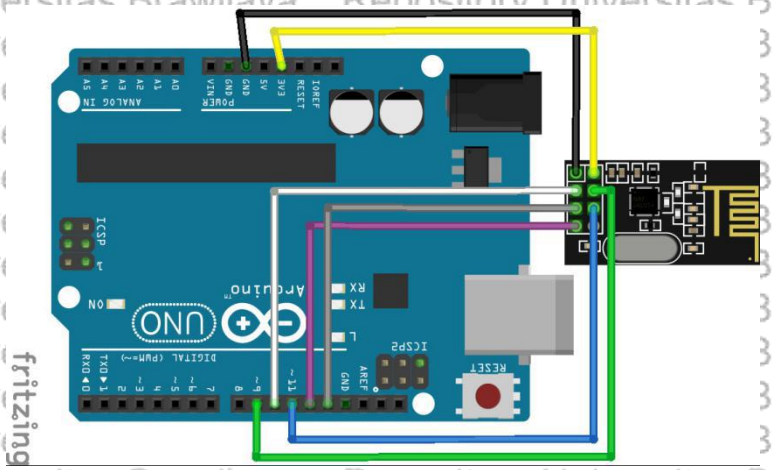
Pin nRF24l01	Pin Arduino Uno
VCC	3.3V
GND	GND
CS	9
CE	10
SCK	13
MOSI	11
MISO	12

Tabel 5.4 Hubungan antara pin LDR, Arduino Uno, dan Resistor 10k

Pin LDR	Pin Arduino Uno	Resistor 10k
P0	5V	-
P1	A0	P0
-	GND	P1

5.1.1.3 Perancangan Skema *Node Sink*

Pada perancangan skema *node sink* dapat dilihat pada Gambar 5.3. Komponen *node sink* yang digunakan terdiri dari arduino uno dan nRF24l01. Hubungan antara pin arduino uno dengan nRF24l01 dapat dilihat pada Tabel 5.4.



Gambar 5.3 Skema *Node Sink*

Tabel 5.3 Hubungan antara pin nRF24l01 dan Arduino Uno

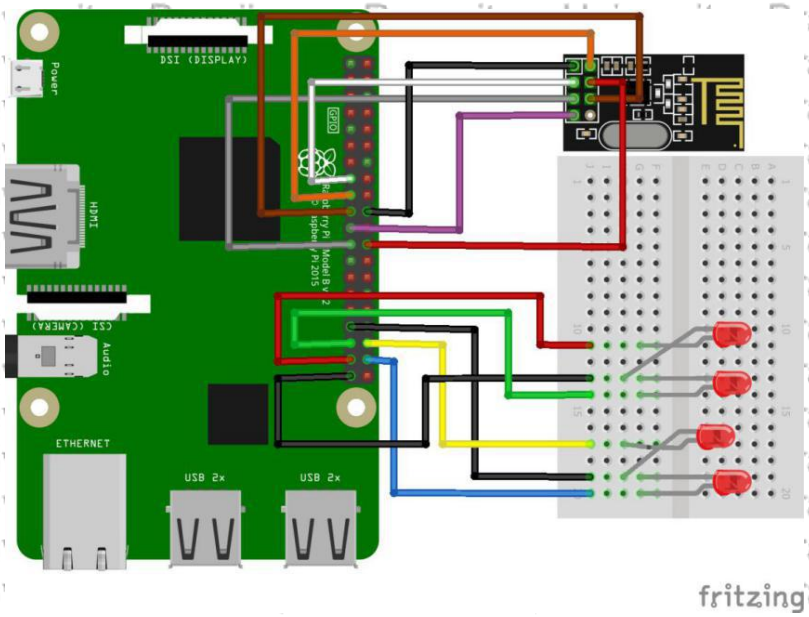
Pin nRF24l01	Pin Arduino Uno
VCC	3.3V
GND	GND
CS	9
CE	10
SCK	13
MOSI	11
MISO	12

Sesuai dengan Tabel 5.4 menunjukkan modul nRF24l01 bekerja pada tegangan 3.3V sehingga pin VCC pada nRF24l01 dihubungkan dengan pin 3.3V pada *board* Arduino Uno dan pin GND pada nRF24l01 dihubungkan dengan pin GND pada *board* Arduino Uno. Pin CS dan CE pada nRF24l01 dihubungkan dengan pin digital 9 untuk CS dan pin digital 10 untuk CE pada *board* Arduino Uno. Pin SCK (Serial Clock) dihubungkan dengan pin digital 13 pada *board* Arduino Uno yang bertindak sebagai pin SCK (Serial Clock). pin MOSI dan MISO pada nRF24l01 dihubungkan pada pin digital 11 untuk MOSI dan pin digital 12 untuk MISO pada *board* Arduino Uno.

5.1.1.4 Perancangan Skema *Node Fog*

Pada perancangan skema *node fog* dapat dilihat pada Gambar 5.4. Komponen *node fog* yang digunakan terdiri dari Raspberry Pi, LED, Bread*board*, dan nRF24l01. Hubungan antara pin raspberry pi dengan LED dapat dilihat pada

Tabel 5.5, dan hubungan antara pin raspberry pi dengan nRF24l01 dapat dilihat pada Tabel 5.6. Pada perancangan arsitektur *node fog* terbagi menjadi dua bagian yaitu bagian peracangan raspberry pi dengan nRF24l01 dan raspberry pi dengan LED



Gambar 5.4 Skema *Node Fog*

Tabel 5.4 Hubungan pin antara nRF24l01 dan Raspberry pi

Pin nRF24l01	Pin Raspberry pi
VCC	3.3V
GND	GND
CS	GPIO8
CE	GPIO22
SCK	GPIO11
MOSI	GPIO10
MISO	GPIO9

Pada bagian perancangan raspberry pi dengan nRF24l011 sesuai dengan Tabel 5.5 menunjukkan modul nRF24l01 bekerja pada tegangan 3.3V sehingga pin VCC pada nRF24l01 dihubungkan dengan pin 3.3V pada *board* Raspberry pi dan pin GND pada nRF24l01 dihubungkan dengan pin GND pada *board* Raspberry pi. Pin CS dan CE pada nRF24l01 dihubungkan dengan GPIO8 untuk CS dan GPIO22 untuk CE pada *board* Raspberry pi. Pin SCK(*Serial Clock*) dihubungkan dengan GPIO11 pada *board* Raspberry pi yang bertindak sebagai pin SCK(*Serial Clock*). pin MOSI dan MISO pada nRF24l01 dihubungkan pada pin GPIO10 untuk MOSI dan pin GPIO9 untuk MISO pada *board* Raspberry pi.

Tabel 5.5 Hubungan pin antara LED dan Raspberry pi

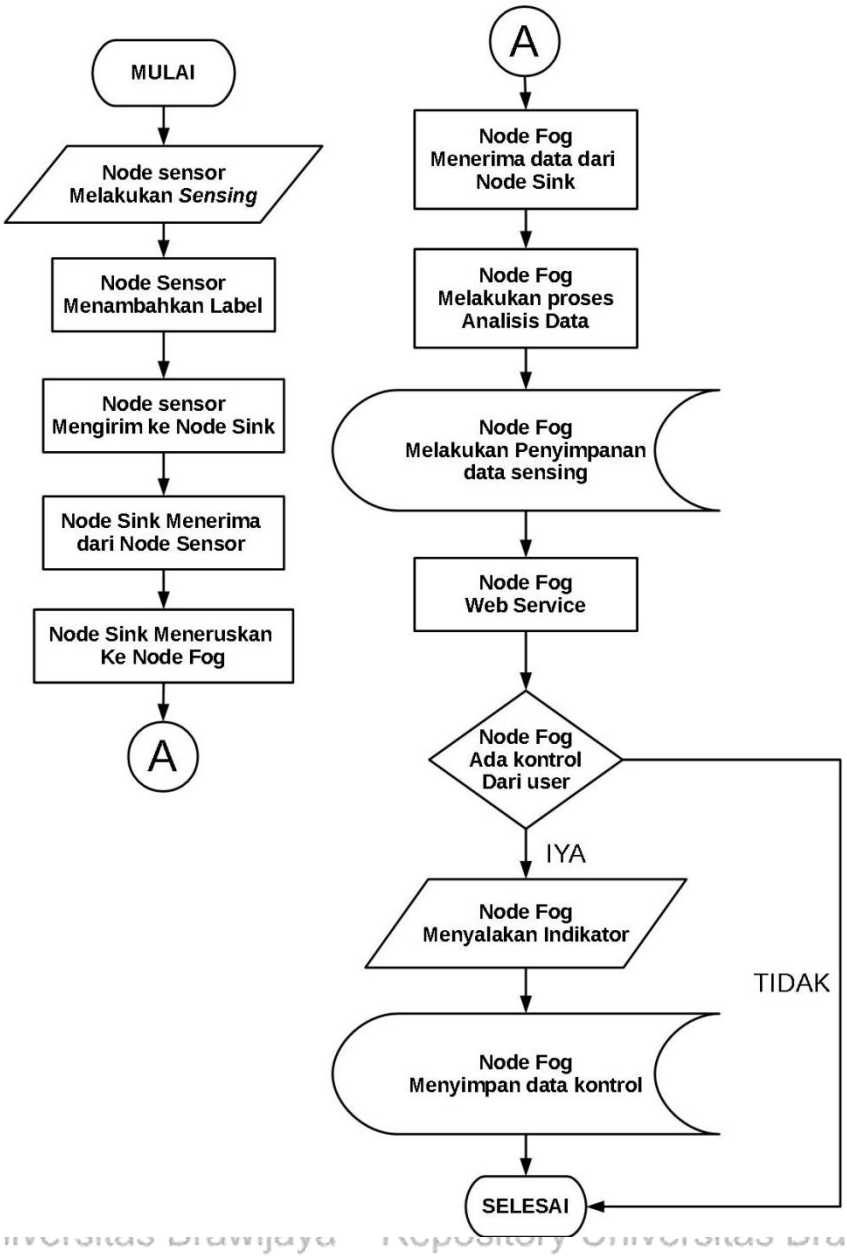
Pin LED	Pin Raspberry pi
LED merah	GPIO26
LED kuning	GPIO16
LED hijau	GPIO19
LED biru	GPIO20
GND	GND

Pada bagian perancangan raspberry pi dengan LED sesuai dengan Tabel 5.6 hubungan pin antara LED warna merah dihubungkan dengan GPIO026, LED warna kuning dihubungkan dengan GPIO16, LED warna hijau dihubungkan GPIO19, LED warna biru dihubungkan dengan GPIO20 dan GND dari semua LED dihubungkan dengan GND yang ada pada *board* Raspberry pi.

5.1.2 Perancangan Perangkat Lunak

5.1.2.1 Perancangan keseluruhan sistem

Perancangan keseluruhan sistem dapat dilihat pada *flowchart* Gambar 5.5. *Node sensor* yang terdiri dari *node sensor* cahaya dan suhu melakukan *sensing* lingkungan, setelah itu menambahkan label sesuai dengan masih masing sensor, setelah itu data yang sudah diberikan label di kirim ke *node sink*. Kemudian *node sink* akan menunggu data *sensing* dari *node sensor*, setelah data *sensing* diterima oleh *node sink* data tersebut akan langsung di kirim ke *Node Fog*. Pada *node fog* akan menunggu data *sensing* dari *node sink*, setelah data diterima oleh *node fog* kemudian data *sensing* tersebut akan dianalisis untuk menentukan apakah hasil data *sensing* melawati nilai *threshold* atau tidak, setelah data *sensing* sudah dianalisis data *sensing* akan disimpan pada database lokal SQLite. Setelah itu data yang tersimpan pada database SQLite akan ditampilkan ke aplikasi web dan data akan dikirim ke *cloud*. Pada aplikasi web akan ditampilkan data *sensing* dan tombol kontrol untuk menyalakan atau mematikan kipas atau lampu yang pada penelitian ini berupa indikator LED, dan aksi *input* dari tombol pada aplikasi web akan disimpan pada database lokal SQLite.



Gambar 5.5 *Flowchart* keseluruhan sistem

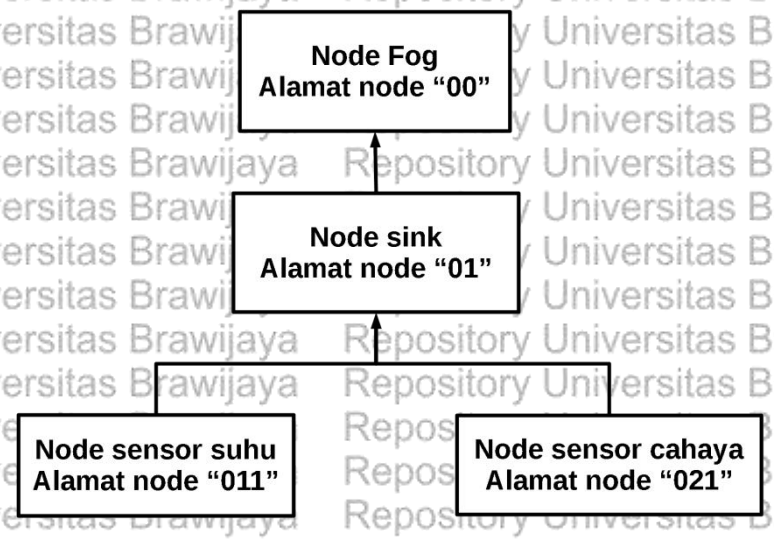
5.1.2.2 Perancangan Pengalamatan Node

Sesuai dengan Gambar 5.6 menunjukkan pengalamatan setiap node dengan *topology tree* sebagai arsitektur jaringannya hal ini disebabkan library RF24Network menggunakan pendekatan tree untuk menentukan alamat node (2019, tmrh20). Setiap angka dalam pengalamatan node akan mewakili posisi dari node itu sendiri sebagai contoh dapat dilihat pada Tabel 5.6 yang menunjukkan pengalamatan node dengan library RF24Network. Oleh sebab itu pada penelitian ini memberikan alamat node sensor suhu dengan nilai "011" sedangkan node sensor cahaya dengan nilai "021" supaya dapat berkomunikasi dengan node sink yang beralamat "01" yang berada di atas satu tingkat dengan

node sensor. Begitu juga dengan node sink diberi alamat "01" supaya dapat berkomunikasi dengan node fog yang beralamat "00".

Tabel 5.6 Contoh Nilai Pengalamatan

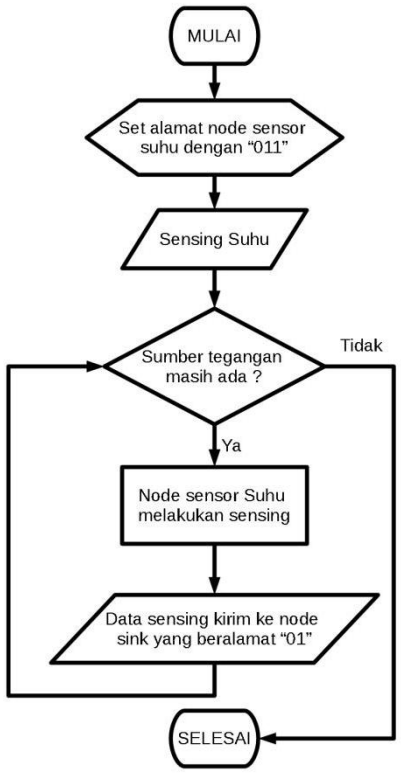
Nilai alamat									Penjelasan
			00						Master Node
		01			04				Tingkat 1 anak dari node master
	011		021			014			Tingkat 2 anak dari tingkat 1
111			121	221			114		Tingkat 3 anak dari tingkat 2
				1221		1114	2114	3114	Tingkat 4 anak dari tingkat 3



Gambar 5.6 Perancangan Alamat Setiap Node

5.1.2.3 Perancangan pengiriman data *sensing node sensor* suhu

Pada perancangan pengiriman data *sensing node sensor* suhu dapat dilihat pada Gambar 5.7.



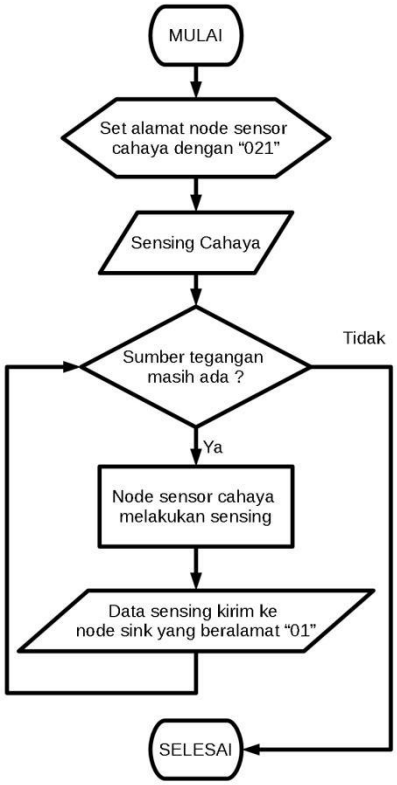
Gambar 5.7 *Flowchart* pengiriman data *sensing node sensor* suhu

Sesuai dengan Gambar 5.7 *Node sensor* suhu terdapat beberapa tahapan. Tahap pertama *node sensor* suhu diberikan alamat "011" sebelum melakukan *sensing* supaya mendefinisikan *node sensor* suhu dapat dikenal oleh node yang lain dan untuk definisikan *node sensor* suhu merupakan node pada bagian level dua dari node master. Tahap kedua *node sensor* akan melakukan *sensing* dengan modul sensor DHT11. Tahap ketiga hasil sensing dari DHT11 kemudian diubah menjadi tipe data string dan ditambah string yang menunjukkan jenis sensor supaya memudahkan pengenalan hasil *sensing* oleh *node fog* untuk di olah lebih lanjut. Tahap keempat *node sensor* suhu mengirimkan data *sensing* ke *node sink* yang memiliki alamat "01".

5.1.2.4 Perancangan pengiriman data *sensing node sensor* cahaya

Pada perancangan pengiriman data *sensing node sensor* cahaya dapat dilihat pada Gambar 5.8. Sesuai dengan Gambar 5.8 *node sensor* cahaya terdapat beberapa tahapan. Tahap pertama *node sensor* cahaya diberikan alamat "021" sebelum melakukan *sensing* supaya mendefinisikan *node sensor* cahaya dapat dikenal oleh node yang lain dan untuk definisikan *node sensor* cahaya merupakan node pada bagian level dua dari node master. Tahap kedua *node sensor* akan melakukan *sensing* dengan modul sensor LDR. Tahap ketiga hasil *sensing* dari LDR kemudian diubah menjadi tipe data string dan ditambah string yang menunjukkan jenis sensor supaya memudahkan pengenalan hasil *sensing*

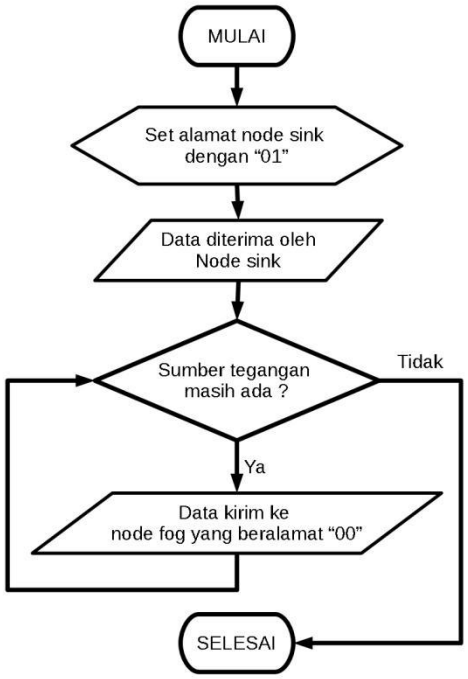
oleh *node fog* untuk di oleh lebih lanjut. Tahap keempat *node sensor* suhu mengirimkan data *sensing* ke *node sink* yang memiliki alamat "01".



Gambar 5.8 *Flowchart* pengiriman data *sensing node sensor* cahaya

5.1.2.5 Perancangan pengiriman data *sensing node sink*

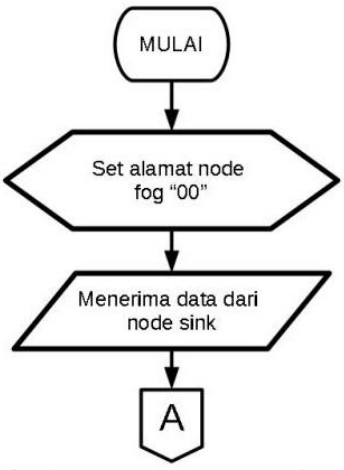
Pada perancangan pengiriman data *sensing node sink* dapat dilihat pada Gambar 5.9. Sesuai dengan Gambar 5.9 *node sensor* cahaya terdapat beberapa tahapan. Tahap pertama *node sink* diberikan alamat "01" sebelum menerima data *sensing* dari *node sensor* suhu atau *node sensor* cahaya supaya *node sink* dapat dikenal oleh node yang lain dan untuk mendefinisikan *node sink* merupakan node pada bagian level satu dari node master. Tahap kedua *node sink* menerima data *sensing* dari *node sensor* suhu atau *node sensor* cahaya. Tahap keempat *node sink* meneruskan data *sensing* ke *node fog* yang memiliki alamat "00".



Gambar 5.9 *Flowchart* pengiriman data *sensing* ke *node fog*

5.1.2.6 Perancangan penerimaan data dari *node sink* ke *node fog*

Pada perancangan penerimaan data dari *node sink* dapat dilihat pada Gambar 5.10.

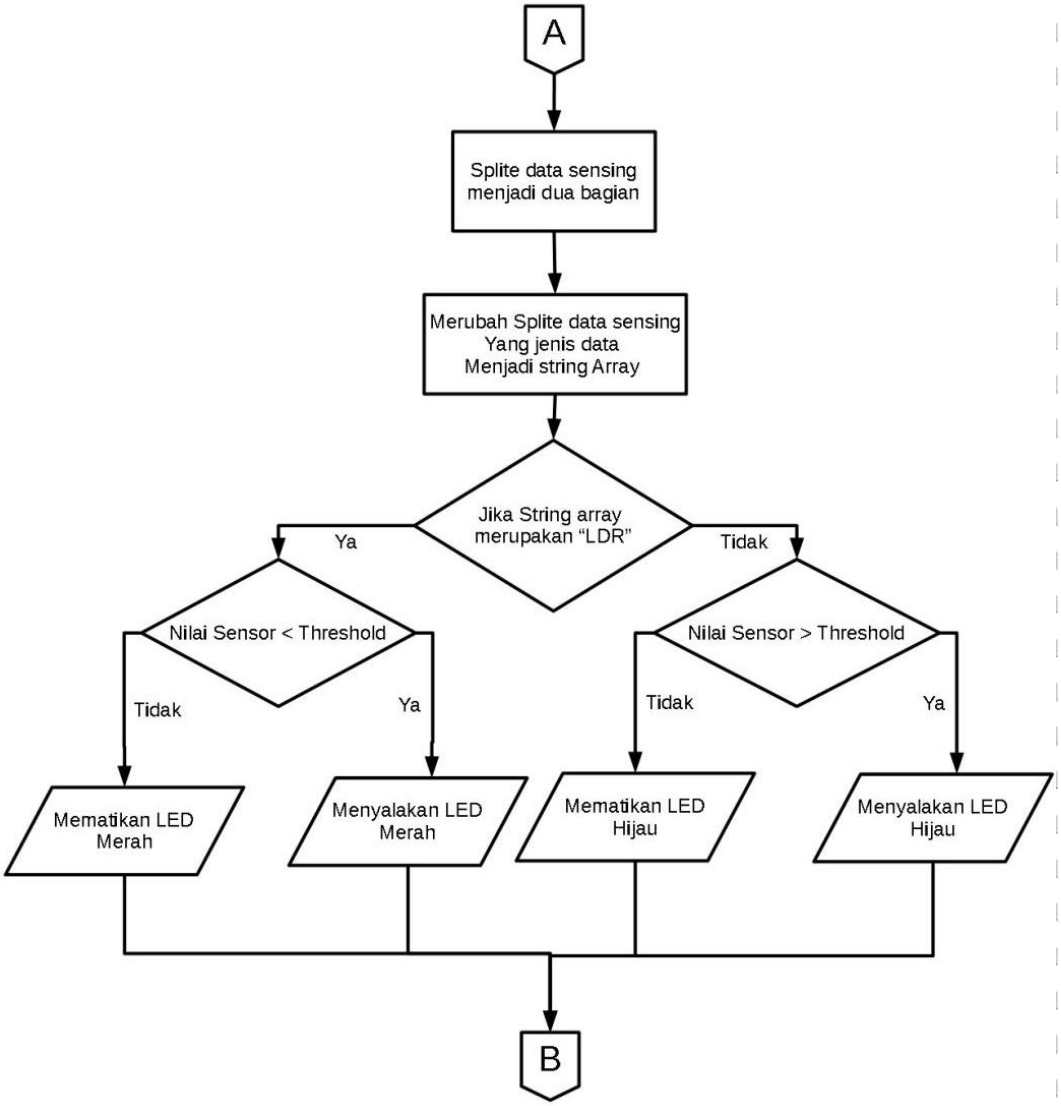


Gambar 5.10 *Flowchart* penerimaan data dari *node sink*

Sesuai dengan Gambar 5.9 *Node fog* memiliki beberapa tahapan. Tahap pertama *node fog* diberikan alamat "00" sebelum menerima data dari *node sink* hal ini untuk menunjukkan *node fog* merupakan master node dan mendefinisikan alamat *node fog* supaya dapat berkomunikasi dengan *node sink*. Tahap kedua *node fog* menerima data *sensing* dari *node sink*, data yang diterima oleh *node fog* akan diolah lebih lanjut seperti ditampilkan melalui aplikasi web, analisis data, dan disimpan.

5.1.2.7 Perancangan analisis data

Pada perancangan analisis data dapat dilihat pada Gambar 5.11 yang menunjukkan flowchart analisis data pada penelitian ini.



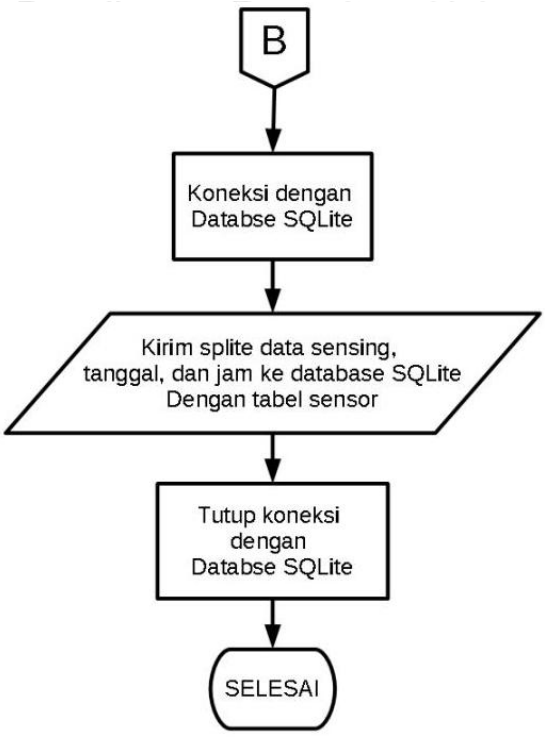
Gambar 5.11 Perancangan analisis data *sensing*

Pada Gambar flowchart tersebut menjelaskan proses analisis data akan dilakukan oleh sistem setelah proses penerimaan data *sensing* dari *node sink*, kemudian data *sensing* akan di *parsing* menjadi dua bagian variabel yaitu variabel jenis sensor dan nilai sensor, hal ini dilakukan untuk kemudahan melakukan analisis data. Sistem akan membandingkan variabel jenis sensor terlebih dahulu, jika variabel sensor adalah suhu maka sistem akan membandingkan nilai variabel nilai sensor, jika variabel nilai sensor kurang dari nilai *threshold* sistem akan menyalakan LED berwarna hijau. Sedangkan jika variabel sensor adalah cahaya maka sistem akan membandingkan nilai variabel nilai sensor dengan nilai *threshold*, jika melebihi nilai threshold sistem akan menyalakan LED berwarna merah, jika tidak sistem akan mematikan LED

berwarna merah. Pada penelitian ini nilai *threshold* variabel sensor suhu bernilai 30 dan nilai *threshold* variabel sensor cahaya bernilai 600.

5.1.2.8 Perancangan penyimpanan data *sensing* ke SQLite

Sesuai pada Gambar 5.12 proses penyimpanan data *sensing* ke database SQLite merupakan kelanjutan dari proses analisis data. Setelah sistem melakukan analisis data sistem akan melakukan proses penyimpanan data ke lokal database SQLite hal ini digunakan untuk ditampilkan ke pengguna melalui aplikasi web. Pada perancangan penyimpanan data ke SQLite terbagi menjadi beberapa tahap. Tahap pertama sistem melakukan koneksi dengan database SQLite setelah proses analisis data selesai. Tahap kedua sistem memasukkan data variabel nilai sensor, variabel jenis sensor, tanggal, dan jam ke database SQLite. Tahap ketiga sistem akan memutuskan koneksi dengan database SQLite yang menandakan proses yang berhubungan dengan database SQLite sudah selesai.



Gambar 5.12 Perancangan penyimpanan data *sensing* ke SQLite

5.1.2.9 Perancangan Database

Pada Tabel 5.7 menunjukkan struktur dari tabel aktuator yang terdiri dari id dengan tipe data Integer yang berfungsi sebagai primary key, statusSuhu dengan tipe data *Text* yang berfungsi sebagai menyimpan data kondisi indikator aktuator suhu yang bernilai "hidup" atau "mati", statusCahaya dengan tipe data *Text* yang berfungsi sebagai menyimpan data kondisi indikator aktuator cahaya yang bernilai "hidup" atau "mati", tgl dengan tipe data *text* yang berfungsi sebagai menyimpan tanggal data masuk ke database, jam dengan tipe data *text* yang berfungsi sebagai menyimpan waktu data ketika masuk ke database.

Pada Tabel 5.8 menunjukkan struktur dari tabel sensor yang terdiri dari id dengan tipe data *Integer* yang berfungsi sebagai primary key, nilai dengan tipe data *Integer* yang berfungsi sebagai menyimpan nilai data sensor, tgl dengan tipe data *text* yang berfungsi sebagai menyimpan tanggal data masuk ke database, jam dengan tipe data *text* yang berfungsi sebagai menyimpan waktu data ketika masuk ke database, dan sensor dengan tipe data *text* yang berfungsi sebagai menyimpan jenis data dari node sensor yang bernilai "DHT" atau "LDR".

Tabel 5.7 Struktur Tabel Aktuator

Key	Type	Deskripsi
id	*Integer*	*Primary key* dari tabel aktuator
statusSuhu	*Text*	Kondisi indikator aktuator dari suhu
statusCahaya	*Text*	Kondisi indikator aktuator dari cahaya
tgl	*Text*	Tanggal data masuk ke database
jam	*Text*	Jam data masuk ke database

Tabel 5.8 Struktur Tabel Suhu

Key	Type	Deskripsi
id	*Integer*	*Primary key* dari tabel *Sensor*
nilai	*Integer*	Nilai data dari *Node Sensor*
tgl	*Text*	Tanggal data masuk ke database
jam	*Text*	Jam data masuk ke database
sensor	*Text*	Jenis *Node Sensor*

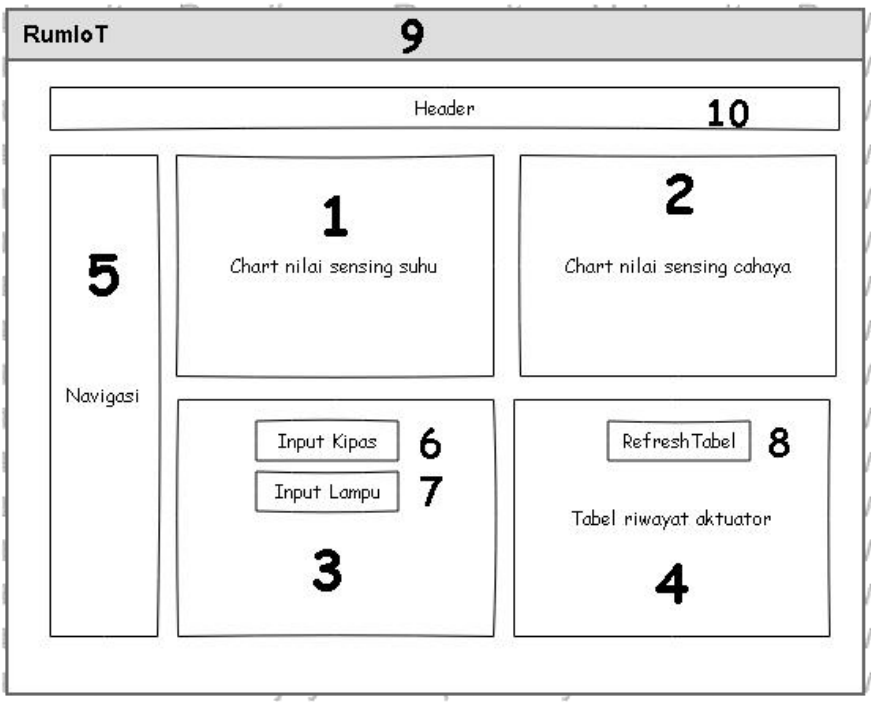
5.1.2.10 Perancangan antarmuka aplikasi web

Pada perancangan aplikasi web terdapat dua perancangan yaitu perancangan antarmuka web. *Layout* perancangan antarmuka web dapat dilihat pada Gambar 5.13.

Penjelasan setiap point dari Gambar 5.13 layout antarmuka aplikasi web dapat dijelaskan sebagai berikut:

1. *Layout* untuk menampilkan hasil data *sensing* dari *node sensor* suhu yang terbaru.
2. *Layout* untuk menampilkan hasil data *sensing* dari *node sensor* cahaya yang terbaru.
3. *Layout* untuk menampilkan tombol kontrol terhadap aktuator kipas dan lampu.

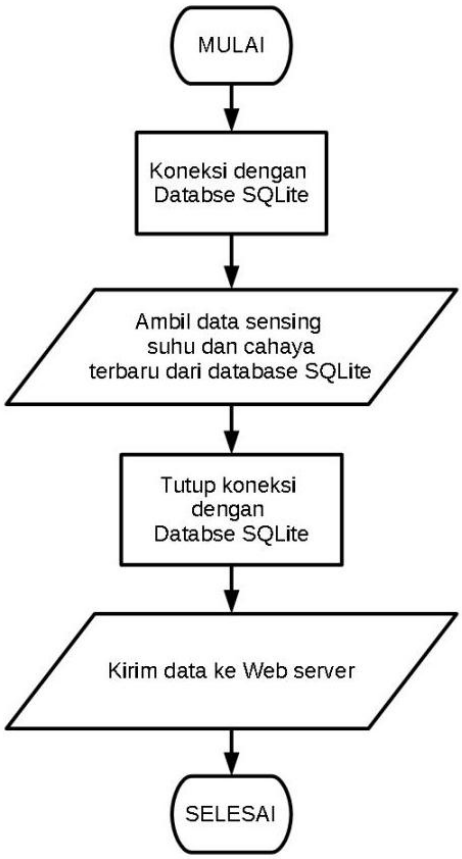
4. *Layout* untuk menampilkan data kondisi aktuator yang baru.
5. *Layout* untuk menampilkan navigasi dari aplikasi web sistem *monitoring*.
6. Tombol untuk menghidupkan atau mematikan kipas.
7. Tombol untuk menghidupkan atau mematikan lampu.
8. Tombol untuk mengupdate tabel data aktuator.
9. *Layout* untuk menampilkan *title* aplikasi web yang pada penelitian ini bernama "RumIoT".
10. *Layout* untuk menampilkan *header* pada aplikasi web yang pada penelitian ini terdiri atas nama halaman, pencarian, dan tombol akun.



Gambar 5.13 Perancangan antarmuka web

5.1.2.11 Perancangan pengambilan data *sensing* dari SQLite ke aplikasi web

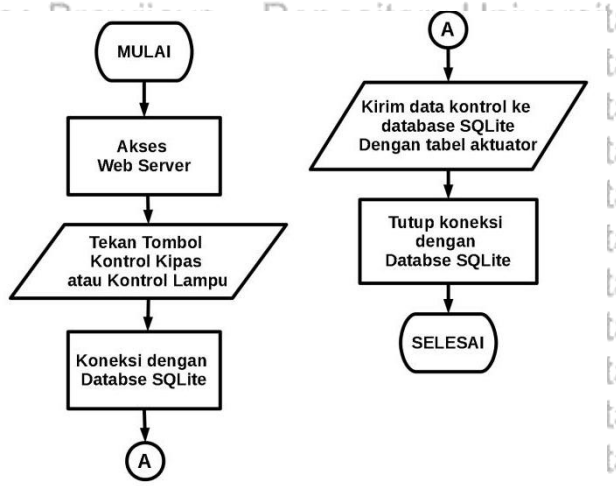
Sesuai pada *flowchart* Gambar 5.14 pengambilan data *sensing* dari database SQLite terdapat beberapa tahapan. Tahap pertama sistem melakukan koneksi dengan database SQLite untuk memastikan sistem dapat melakukan pengambilan data *sensing*. Tahap kedua sistem melakukan pengambilan data *sensing* terbaru dari database SQLite. Tahap ketiga sistem memutuskan koneksi ke database SQLite hal ini untuk menandakan proses pengambilan data *sensing* sudah selesai. Tahap keempat data *sensing* yang sudah diambil akan dikirim ke aplikasi web melalui webSocket supaya pengiriman data *sensing* ke aplikasi web dapat *real time*.



Gambar 5.14 ***Flowchart*** **pengambilan data dari SQLite ke web server**

5.1.2.12 Perancangan penyimpanan data aktuator ke SQLite

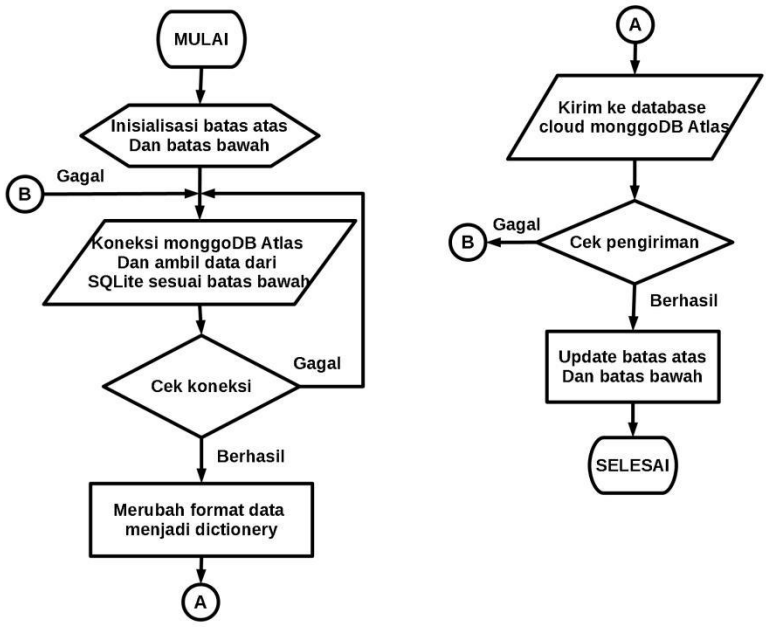
Sesuai pada Gambar 5.15 proses penyimpanan data aktuator ke database SQLite ada beberapa tahap. Tahap pertama user mengakses aplikasi web sistem *monitoring* melalui jaringan yang sama dengan sistem. Tahap kedua user menekan tombol kontrol kipas atau kontrol lampu, aksi yang dilakukan user tersebut akan disimpan ke database SQLite. Tahap ketiga sistem melakukan koneksi dengan database SQLite untuk memulai persiapan pengiriman data aktuator ke database SQLite. Tahap keempat aksi menekan tombol yang dilakukan oleh user disimpan ke dalam database SQLite. Tahap kelima setelah sistem memasukkan data aktuator ke database SQLite, sistem akan memutuskan koneksi dengan database SQLite yang menandakan proses penyimpanan data selesai.



Gambar 5.15 *Flowchart* pengiriman data aktuator ke SQLite

5.1.2.13 Perancangan penyimpanan data ke mongoDB Atlas

Sesuai pada Gambar 5.16 proses penyimpanan data ke cloud mongoDB Atlas terdapat beberapa tahapan.



Gambar 5.16 *Flowchart* pengiriman data *sensing* ke mongoDB Atlas

Tahap pertama yaitu melakukan inisialisasi batas atas dan batas bawah untuk menentukan pengambilan data yang akan dikirim ke mongoDB Atlas. Tahap kedua melakukan koneksi ke cloud mongoDB Atlas dan mengambil data pada SQLite sesuai dengan batas atas dan batas bawah, jika terjadi kegagalan sistem akan melakukan koneksi ulang ke cloud mongoDB atlas dan mengambil data pada SQLite, jika berhasil data dari SQLite diubah format data menjadi *dictionary* supaya data mempunyai atribut *key* dan *value* sehingga data mudah dipahami. Tahap ketiga mengirim data ke cloud mongoDB Atlas jika gagal sistem

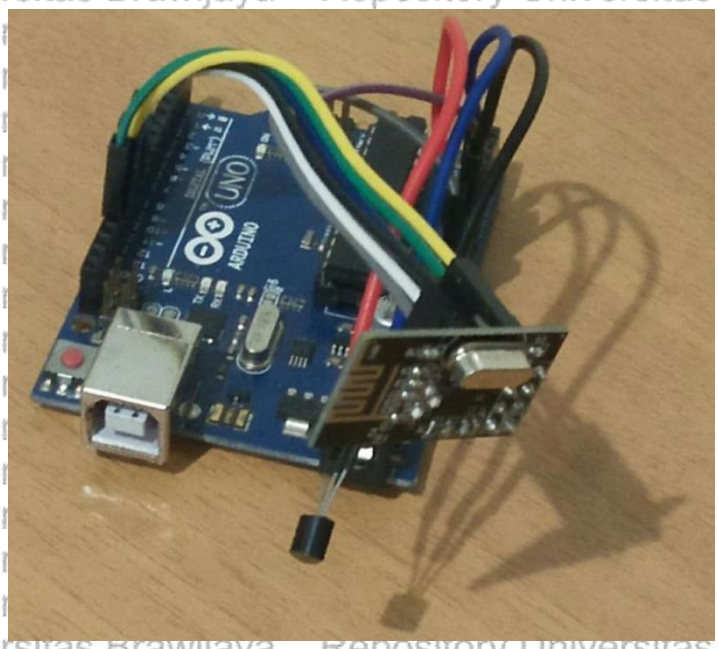
akan kembali ke tahap dua, jika berhasil batas atas dan batas bawah diperbarui supaya pengambilan data ke SQLite dapat mengambil data yang baru.

5.2 Implementasi

Pada bab implementasi, berisi mengenai cara menerapkan sistem yang telah dirancang sebelumnya. Sub bab implementasi terdiri atas implementasi *node sensor* suhu, implementasi *node sensor* cahaya, implementasi *node sink*, dan implementasi *node fog* yang terbagi menjadi implementasi perangkat keras dan implementasi perangkat lunak.

5.2.1 Implementasi Perangkat Keras

5.2.1.1 Implementasi Skema *Node Sensor* Suhu



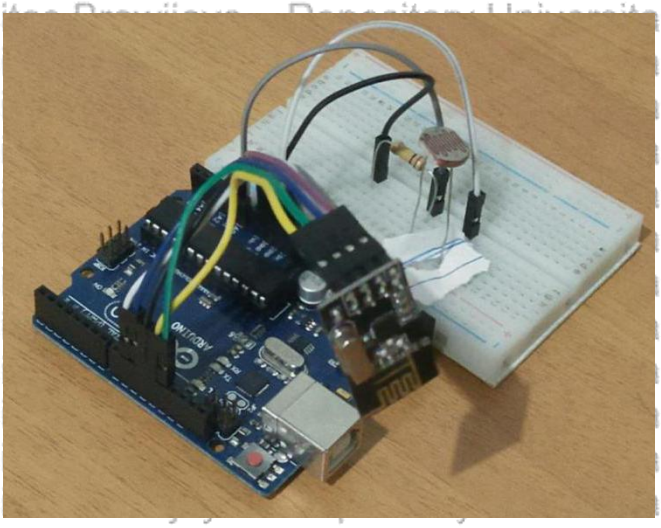
Gambar 5.17 Implementasi Node suhu

Pada implementasi skema *node sensor* suhu dilakukan sesuai dengan perancangan pada Gambar 5.1. Perancangan tersebut meliputi Arduino Uno yang dihubungkan dengan sensor DHT11 yang memiliki fungsi untuk membaca suhu lingkungan, dan modul komunikasi nRF24l01 sebagai media komunikasi dengan *node sink*. Baik sensor DHT11 dan nRF34l01 terhubung dengan *board* Arduino Uno dengan menggunakan kabel jumper. Foto implementasi skema *node sensor* suhu dapat dilihat pada Gambar 5.17.

5.2.1.2 Implementasi Skema *Node Sensor* Cahaya

Pada implementasi *node sensor* cahaya dilakukan sesuai dengan perancangan pada Gambar 5.2. Perancangan tersebut meliputi Arduino Uno yang dihubungkan dengan sensor LDR yang memiliki fungsi untuk membaca intensitas cahaya, resistor 10k yang berfungsi untuk meningkatkan akurasi pembacaan sensor LDR, dan modul komunikasi nRF24l01 sebagai media

komunikasi dengan *node sink*. Baik sensor DHT11 dan nRF34l01 terhubung dengan *board* Arduino Uno dengan menggunakan kabel jumper. Foto implementasi skema *node sensor* cahaya dapat dilihat pada Gambar 5.18.



Gambar 5.18 Implementasi Node cahaya

5.2.1.3 Implementasi Skema *Node Sink*

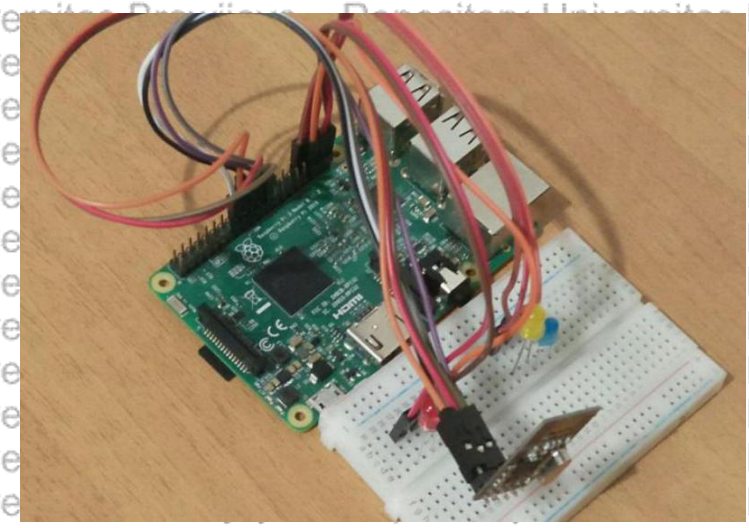
Gambar 5.19 Implementasi Node cahaya

Pada implementasi *node sensor* sink dilakukan sesuai dengan perancangan pada Gambar 5.3. Perancangan tersebut meliputi Arduino Uno yang dihubungkan dengan modul komunikasi nRF24l01 sebagai media komunikasi dengan *node fog*. nRF24l01 dihubungkan ke *board* Arduino Uno dengan menggunakan kabel jumper. Foto skema *node sink* dapat dilihat pada Gambar 5.19.

5.2.1.4 Implementasi skema *node fog*

Pada implementasi *node fog* dilakukan sesuai dengan perancangan pada Gambar 5.4. Perancangan tersebut meliputi Arduino Uno yang dihubungkan

dengan modul komunikasi nRF24l01 sebagai media komunikasi, dan empat LED. Baik modul komunikasi nRF24l01 dan LED terhubung dengan *board* Arduino Uno melalui kabel jumper. Foto skema *node sink* dapat dilihat pada Gambar 5.20.



Gambar 5.20 Implementasi *Node fog*

5.2.2 Implementasi Perangkat Lunak

5.2.2.1 Implementasi pengiriman data *sensing node sensor* suhu

Tabel 5.9 merupakan *Pseudocode* pada implementasi *node sensor* suhu. Pada baris 1 pada implementasi *Pseudocode* dilakukan pemanggilan *library* yang akan digunakan oleh *node sensor* suhu supaya modul komunikasi nRF24l01 dapat bekerja pada Arduino Uno. Pada baris 2 *node sensor* suhu akan melakukan mendefinisikan alamat node dan mendefinisikan pin nRF24l01 yang tersambung dengan *board* Arduino Uno supaya modul komunikasi nRF24l01 dapat bertukar data dengan Arduino Uno. Pada baris 3 *node sensor* suhu akan melakukan *sensing* suhu melalui modul sensor DHT11 yang terhubung dengan *board* Arduino Uno pada pin A0. Pada baris 4 - 5 hasil data *sensing* suhu yang dilakukan oleh sensor DHT11 akan dirubah menjadi char array yang kemudian akan ditambahkan dengan kata " DHT" supaya proses penyortiran data pada *node fog* akan lebih mudah. Pada baris 6 - 7 *node sensor* suhu akan mendefinisikan alamat *node* yang akan menerima data dengan alamat node "011", kemudian data sensor suhu yang sudah diolah sebelumnya dikirimkan ke alamat *node* yang sudah ditentukan.

Tabel 5.9 *Pseudocode Node Sensor* Suhu

No	*Pseudocode Node Sensor* Suhu
1	INCLUDE library
2	DEFINE nRF24l01
3	READ data suhu
4	PACK data suhu ke array char

5	INSERT " DHT" ke data suhu
6	SET alamat "011" nRF24l01 yang dituju
7	SEND data suhu melalui nRF24l01

5.2.2.2 Implementasi pengiriman data *sensing node sensor* cahaya

Tabel 5.10 merupakan *Pseudocode* pada implementasi *node sensor* cahaya. Pada baris 1 pada implementasi *Pseudocode* dilakukan pemanggilan *library* yang akan digunakan oleh *node sensor* cahaya supaya modul komunikasi nRF24l01 dapat bekerja pada Arduino Uno. Pada baris 2 *node sensor* cahaya akan melakukan mendefinisikan alamat node dan mendefinisikan pin nRF24l01 yang tersambung dengan *board* Arduino Uno supaya modul komunikasi nRF24l01 dapat bertukar data dengan Arduino Uno. Pada baris 3 *node sensor* cahaya akan melakukan *sensing* intensitas cahaya melalui modul sensor LDR yang terhubung dengan *board* Arduino Uno pada pin A0. Pada baris 4 - 5 hasil data *sensing* intensitas cahaya yang dilakukan oleh sensor LDR akan dirubah menjadi *char array* yang kemudian akan ditambahkan dengan kata " LDR" supaya proses penyortiran data pada *node fog* akan lebih mudah. Pada baris 6 - 7 *node sensor* cahaya akan mendefinisikan alamat *node* yang akan menerima data dengan alamat node "011", kemudian data sensor cahaya yang sudah diolah sebelumnya dikirimkan ke alamat *node* yang sudah ditentukan.

Tabel 5.10 *Pseudocode Node Sensor* Cahaya

No	*Pseudocode Node Sensor* Cahaya
1	INCLUDE library
2	DEFINE nRF24l01
3	READ data cahaya
4	PACK data cahaya ke tipe data array char
5	INSERT " LDR" into data suhu
6	SET alamat "011" nRF24l01 yang dituju
7	SEND data cahaya melalui nRF24l01

5.2.2.3 Implementasi Pengiriman Data *Sensing Node Sink*

Tabel 5.11 merupakan *Pseudocode* pada implementasi *node sink*. Pada baris 1 pada implementasi *Pseudocode* dilakukan pemanggilan *library* yang akan digunakan oleh *node sink* supaya modul komunikasi nRF24l01 dapat bekerja pada Arduino Uno. Pada baris 2 *node sink* akan melakukan mendefinisikan alamat node dan mendefinisikan pin nRF24l01 yang tersambung dengan *board* Arduino Uno supaya modul komunikasi nRF24l01 dapat bertukar data dengan Arduino Uno. Pada baris 3 *node sink* akan menerima data hasil *sensing* dari *node sensor* suhu atau *node sensor* cahaya dengan cara mendefinisikan terlebih dahulu alamat dari *node sensor* sebelum melakukan penerimaan data. Pada baris

4 hasil penerimaan data *sensing* akan ditampung ke dalam tipe data *char array* hal ini dilakukan karena tipe data yang dikirim oleh *node sensor* berupa tipe data *char array*. Pada baris 5 - 6 *node sink* akan mendefinisikan alamat *node* yang akan menerima data dengan alamat node "00", kemudian data *sensing* yang sudah diterima oleh *node sink* sebelumnya akan dikirimkan ke alamat *node fog* yang sudah ditentukan.

Tabel 5.11 *Pseudocode* Pengiriman Data *Sensing Node Sink*

No	*Pseudocode* Pengiriman Data *Sensing Node Sink*
1	INCLUDE library
2	DEFINE nRF24l01
3	RECEIVE data *sensing* dari *node sensor*
4	PACK data *sensing* ke array char
5	SET alamat "00" nRF24l01 yang dituju
6	SEND data *sensing* melalui nRF24l01

5.2.2.4 Implementasi Penerimaan Data Dari *Node Sink* Ke *Node Fog*

Tabel 5.12 merupakan *Pseudocode* pada implementasi penerimaan data *sensing* oleh *node fog*. Pada baris 1 sistem akan melakukan pemanggilan library yang akan digunakan oleh *node fog* supaya sistem dapat bekerja dengan baik pada perangkat Raspberry pi. Pada baris 2 *node fog* akan melakukan mendefinisikan alamat *node* dan mendefinisikan pin nRF24l01 yang tersambung dengan *board* Raspberry pi supaya modul komunikasi nRF24l01 dapat bertukar data dengan Raspberry pi. Pada baris 3 *node fog* akan menerima data hasil *sensing* dari *node sink* dengan cara mendefinisikan terlebih dahulu alamat dari *node sink* sebelum melakukan penerimaan data. Pada baris 4 hasil penerimaan data *sensing* akan ditampung ke dalam tipe data *char array* hal ini dilakukan karena tipe data yang dikirim oleh *node sink* berupa tipe data *char array*.

Tabel 5.12 *Pseudocode* Penerimaan Data Dari *Node Sink* Ke *Node Fog*

No	*Pseudocode* Penerimaan Data Dari *Node Sink* Ke *Node Fog*
1	INCLUDE library
2	DEFINE nRF24l01
3	RECEIVE data *sensing* from *node sink*
4	PACK data *sensing* into array char

5.2.2.5 Implementasi Analisis Data Pada *Node Fog*

Tabel 5.13 merupakan *Pseudocode* untuk implementasi analisis data pada *node fog*. Pada baris 1 data *sensing* yang sebelumnya sudah diterima oleh *node fog* akan di pisah menjadi dua bagian yaitu bagian nilai sensor dan jenis sensor,

proses memisahkan data *sensing* dengan menggunakan *delimiter* spasi yang memisahkan bagian nilai sensor dan jenis sensor. Pada baris 2 - 6 hasil pemisahan data *sensing* akan dilakukan pengecekan pada bagian jenis sensor, jika jenis sensor adalah "LDR" maka sistem akan melakukan pengecekan terhadap nilai *sensing* sensor cahaya, jika nilai *sensing* sensor cahaya kurang dari nilai threshold yang sudah ditentukan maka sistem akan menyalakan LED merah, namun jika nilai *sensing* sensor cahaya lebih dari nilai threshold yang sudah ditentukan maka sistem akan mematikan LED merah. Pada baris 7 - 11 merupakan *Pseudocode* untuk dilakukan pengecekan pada bagian jenis sensor suhu, kemudian sistem akan melakukan pengecekan terhadap nilai *sensing* sensor suhu, jika nilai *sensing* sensor suhu lebih dari nilai threshold yang sudah ditentukan maka sistem akan menyalakan LED hijau, namun jika nilai *sensing* sensor suhu kurang dari nilai threshold yang sudah ditentukan maka sistem akan mematikan LED hijau.

Tabel 5.13 *Pseudocode* Analisis Data Pada *Node Fog*

No	*Pseudocode* Analisis Data Pada *Node Fog*
1	PACK data *sensing* become two variable
2	IF type data *sensing* is "LDR"
3	IF value data *sensing* less than threshold
4	SET red led high
5	ELSE
6	SET red led low
7	ELSE
8	IF value data *sensing* more than threshold
9	SET green led high
10	ELSE
11	SET green led low

5.2.2.6 Implementasi Penyimpanan Data *Sensing* Ke SQLite Pada *Node Fog*

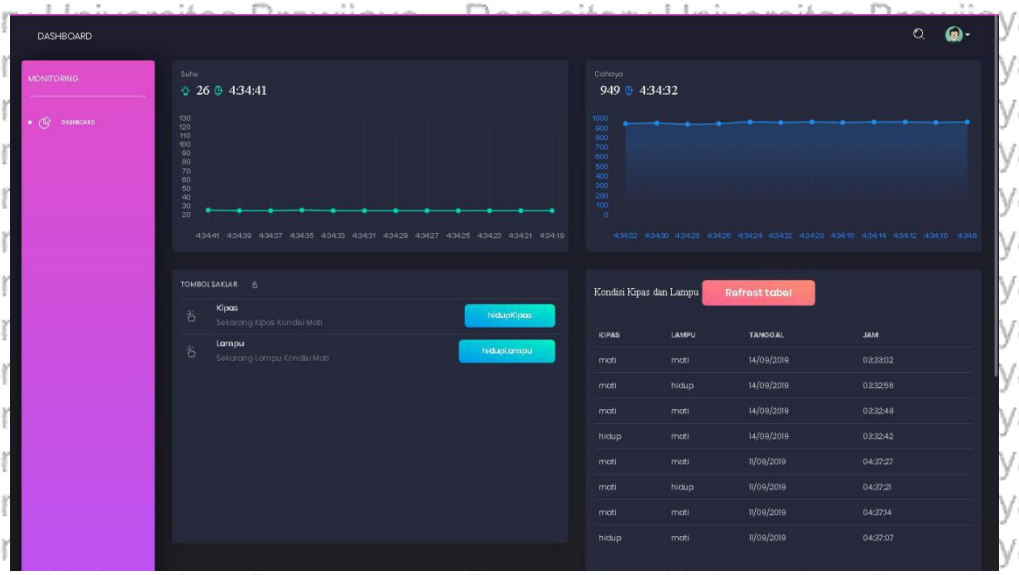
Tabel 5.14 merupakan *Pseudocode* untuk implementasi penyimpanan data *sensing* ke database SQLite pada *node fog*. Pada baris 1 sistem mendefinisikan *date* untuk menentukan tanggal penyimpanan data *sensing* ke database SQLite. Pada baris 2 sistem mendefinisikan *time* untuk menentukan waktu penyimpanan data *sensing* ke database SQLite. Pada baris 3 sistem mendefinisikan database SQLite untuk siap menggunakan database SQLite. Pada baris 4 - 6 sistem membuka *session* file database SQLite yang akan digunakan supaya data file database SQLite dapat di akses, setelah proses membuka database SQLite berhasil, data, tanggal, dan waktu *sensing* akan di masukan ke dalam database SQLite, kemudian sistem akan menutup *session* file database SQLite supaya sistem tidak dapat mengakses database SQLite.

Tabel 5.14 *Pseudocode* Penyimpanan Data *Sensing* Ke SQLite Pada *Node Fog*

No	***Pseudocode* Penyimpanan Data *Sensing* Ke** SQLite **Pada *Node Fog***
1	DEFINE date
2	DEFINE time
3	DEFINE SQLite
4	OPEN database SQLite
5	INSERT data *sensing*, date, and time into database SQLite
6	CLOSE database SQLite

5.2.2.7 Implementasi antarmuka aplikasi web pada *node fog*

Implementasi antarmuka aplikasi web untuk sistem pemantauan kondisi lingkungan rumah pada penelitian ini dapat dilihat pada Gambar 5.21. Pada tampilan halaman utama aplikasi web terdapat informasi dari data *sensing* suhu, dan cahaya, bar pencarian, navigasi, tombol kontrol aktuator dan informasi kontrol dari aktuator. Aplikasi web pada penelitian ini pada bagian *front-end* menggunakan HTML, CSS, dan javascript dengan bantuan *library* chartjs untuk membuat grafik *chart*, dan *framework* bootstrap untuk mempermuda *layout* tampilan aplikasi web, sedangkan pada bagian *back-end* aplikasi web menggunakan python dengan bantuan *framework* flask sebagai *web server* dan *library* eventlet untuk menerapkan mekanisme *websocket* dalam pengiriman data *sensing* ke aplikasi web.



Gambar 5.21 Tampilan Aplikasi Web

5.2.2.8 Implementasi Pengambilan Data *Sensing* Dari SQLite Ke Aplikasi Web Pada *Node Fog*

Pada Tabel 5.15 merupakan *Pseudocode* pengambilan data *sensing* dari database SQLite untuk dikirimkan ke aplikasi web dengan menggunakan

webscoket pada *node fog*. Pada baris 1 sistem akan memanggil library yang akan digunakan oleh *node fog* dalam melakukan pengambilan database SQLite dan mengirimkan data *sensing* ke aplikasi web menggunakan websocket. Pada baris 2 dan 3 sistem mendefinisikan websocket dan database SQLite, supaya sistem dapat menerapkan mekanisme pengiriman data menggunakan websocket dan sistem dapat melakukan akses terhadap database SQLite. Pada baris 4 - 6 sistem membuka *session* file database SQLite yang akan digunakan supaya data file database SQLite dapat di akses, setelah proses membuka database SQLite berhasil, sistem akan mengambil data *sensing* pada database SQLite, kemudian sistem akan menutup *session* file database SQLite supaya sistem tidak dapat mengakses database SQLite. Pada baris 7 hasil pengambilan data *sensing* dari database SQLite akan dikirim ke aplikasi web melalui websocket.

Tabel 5.15 *Pseudocode* Pengambilan Data *Sensing* Dari Sqlite Ke Aplikasi Web Pada *Node Fog*

No	*Pseudocode* Pengambilan Data *Sensing* Dari SQLite Ke Aplikasi Web Pada *Node Fog*
1	INCLUDE library
2	DEFINE WebSocket
3	DEFINE SQLite
4	OPEN database SQLite
5	FEATCH data *sensing*
6	CLOSE database SQLite
7	SEND into Web Server

5.2.2.9 Implementasi Penyimpanan Data Aktuator Ke SQLite Pada *Node Fog*

Tabel 5.16 merupakan *Pseudocode* penyimpanan data aktuator ke dalam database SQLite dari proses pengguna melakukan aksi terhadap kontrol pada aplikasi web. Pada baris 1 sistem menggunakan library untuk mengakses GPIO pada raspberry pi, mengakses database SQLite, dan membuat web server dengan menggunakan flask. Pada baris 2 dan 3 sistem mendefinisikan flask dan database SQLite, supaya sistem dapat menerapkan web server pada *node fog* dan sistem dapat melakukan akses terhadap database SQLite. Pada baris 4 sistem aplikasi web akan melakukan pemeriksaan terhadap permintaan method POST yang diterima, sebab tombol kontrol pada aplikasi web berupa method POST. Pada baris 5 - 24 sistem melakukan pengecekan masukan data dari method POST, setelah melakukan pemeriksaan sistem akan menyalakan atau mematikan LED yang sesuai dengan perintah dari masukan data, kemudian sistem akan membuka *session* file database SQLite yang digunakan supaya data file database SQLite dapat di akses, setelah proses membuka database SQLite berhasil, setelah itu sistem memasukkan data aktuator pada database SQLite,

kemudian sistem akan menutup *session* file database SQLite supaya sistem tidak dapat mengakses database SQLite.

Tabel 5.16 *Pseudocode* Penyimpanan Data Aktuator Ke SQLite Pada *Node Fog*

No	*Pseudocode* Penyimpanan Data Aktuator Ke SQLite Pada *Node Fog*
1	INCLUDE library
2	DEFINE flask
3	DEFINE SQLite
4	IF request method equal POST
5	IF request form equal “hidupKipas”
6	SET led blue high
7	OPEN database sqlite
8	INSERT data action into database sqlite
9	CLOSE database sqlite
10	IF request form equal “matiKipas”
11	SET led blue low
12	OPEN database sqlite
13	INSERT data action into database sqlite
14	CLOSE database sqlite
15	IF request form equal “hidupLampu”
16	SET led red high
17	OPEN database sqlite
18	INSERT data action into database sqlite
19	CLOSE database sqlite
20	IF request form equal “matiLampu”
21	OPEN database sqlite
22	SET led red low
23	INSERT data action into database sqlite
24	CLOSE database sqlite

5.2.2.10 Implementasi Penyimpanan Data Ke MongoDB Atlas Pada *Node Fog*

Tabel 5.17 merupakan *Pseudocode* dari backup data *sensing* di database SQLite ke cloud mongoDB Atlas. Pada baris 1 sistem memanggil library untuk koneksi dengan database cloud mongoDB Atlas supaya sistem dapat melakukan penanganan request dan response pada cloud database mongoDB Atlas dan

memanggil library untuk koneksi dengan database SQLite supaya sistem dapat mengakses database SQLite. Pada baris 2 sistem akan mendefinisikan batas atas dan batas bawah yang berisi id terakhir dari data *sensing* pada database SQLite untuk nantinya digunakan sebagai menentukan pengambilan data *sensing* pada database SQLite yang akan di backup ke cloud database mongoDB Atlas. Pada baris 3 - 8 sistem akan melakukan uji coba untuk melakukan koneksi dengan cloud database mongoDB Atlas dan melakukan uji coba untuk mengambil data *sensing* pada database SQLite sesuai dengan batas bawah yang ditentukan, jika terjadi kegagalan maka sistem akan menampilkan "gagal koneksi" dan akan melakukan koneksi ulang dengan cloud database mongoDB Atlas dan database SQLite. Pada baris 9 - 10 jika sistem dapat tersambung dengan cloud database mongoDB Atlas dan dapat mengambil data *sensing* pada database SQLite, sistem akan mencoba untuk melakukan memasukkan data *sensing* ke dalam cloud database mongoDB Atlas. Pada baris 11 - 12 sebelum data *sensing* dikirim ke cloud database mongoDB Atlas data *sensing* diubah menjadi data dictionary, kemudian sistem akan memperbarui batas atas dengan id terakhir dari data *sensing* terbaru yang sudah diambil. Pada baris 13 - 15 sistem akan melakukan pemeriksaan jika batas atas dan batas bawah tidak sama maka sistem akan mengirim data *sensing* ke cloud database mongoDB Atlas dan memperbarui batas bawah dengan nilai id terakhir dari data *sensing* yang terakhir diambil. Pada baris 16 - 17 jika terjadi kegagalan saat melakukan pengiriman data *sensing* ke *cloud* database mongoDB Atlas maka sistem akan menampilkan "gagal insert data" dan sistem akan melakukan koneksi ulang dengan cloud database mongoDB dan database SQLite. Pada baris 18 - 19 jika sistem berhasil memasukkan data *sensing* ke dalam cloud database mongoDB Atlas sistem akan memutus koneksi dengan database SQLite.

Tabel 5.17 *Pseudocode* penyimpanan data ke mongoDB Atlas pada *node fog*

No	*Pseudocode* penyimpanan data ke mongoDB Atlas pada *node fog*
1	INCLUDE library
2	DEFINE batasAtas, dan batasBawah data *sensing*
3	TRY
4	CONNECT mongoDB Atlas
5	OPEN database sqlite
6	FEATCH data *sensing* where id greater than batasBawah from sqlite
7	EXCEPT
8	PRINT "gagal koneksi"
9	FINALLY
10	TRY
11	PACK data *sensing* into dictionary

```
12          UPDATE batasAtas
13      IF batasAtas not equal batasBawah
14          INSERT data sensing into mongoDB Atlas
15          UPDATE batasBawah
16      EXCEPT
17      PRINT “gagal insert data”
18      FINALLY
19      CLOSE database sqlite
```

BAB 6 PENGUJIAN

Pada Bab ini akan menjelaskan mengenai hasil dan pembahasan dari pengujian yang telah dilakukan. Hasil dan pembahasan pada penelitian ini bertujuan untuk mengetahui kinerja dari *node fog* berdasarkan pengujian fungsional sistem yang telah ditentukan sebelumnya dan kinerja sistem dengan *parameter* pengujian *delay,* RTT*(round trip-time)*, *Throughput, Packet Loss,* dan variasi waktu pengiriman ke *cloud* menurut jarak pengiriman data baik ada penghalang atau tidak ada penghalang.

6.1 Hasil dan Analisis Fungsional Sistem

Pengujian fungsional sistem pada penelitian ini bertujuan untuk mengetahui kebutuhan fungsional yang telah ditentukan sebelumnya sesuai dengan implementasi yang telah dilakukan. Penerapan pengujian fungsional sistem pada penelitian ini merujuk pada pengujian perancangan dan implementasi yang telah dibuat sebelumnya.

6.1.1 Hasil dan Analisis Kode KS-01

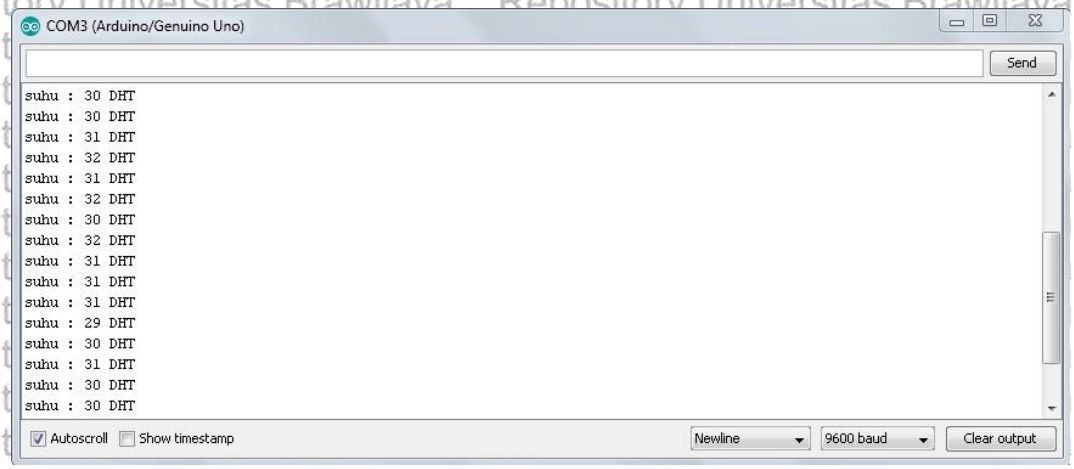
Pada pengujian kode KS-01 adalah untuk menguji apakah *node sensor* dapat melakukan *sensing* lingkungan. Penjelasan lengkap tentang pengujian kode KS-01 dapat dilihat pada Tabel 6.1.

Tabel 6.1 Deskripsi Pengujian Kode-KS-01

Kode	KS-01
Nama Pengujian	*Node sensor* mampu untuk melakukan *sensing* lingkungan
Tujuan	Mengetahui hasil pengujian dari kemampuan *node sensor* dalam melakukan *sensing* terhadap lingkungan
Skenario Pengujian	1. Modul *sensor* terpasang pada masing - masing *node sensor* 2. Arduino Uno terpasang pada sumber daya 3. Arduino Uno mengambil data *sensing* dari modul sensor 4. Data *sensing* ditampilkan pada *serial monitor* di Arduino IDE
Hasil yang diharapkan	*Node sensor* dapat melakukan akuisisi data dari modul sensor yang sudah terpasang.
Hasil Pengujian	*Node sensor* dapat melakukan akuisisi data dari modul sensor yang sudah terpasang.

6.1.1.1 Hasil Pengujian Kode KS-01 Pada *Node Sensor* Suhu

Gambar 6.1 merupakan hasil dari pengujian kemampuan *node sensor* suhu untuk dapat melakukan *sensing* terhadap lingkungan dan hasil *sensing* dapat ditampilkan ke *serial monitor* pada aplikasi Arduino IDE.

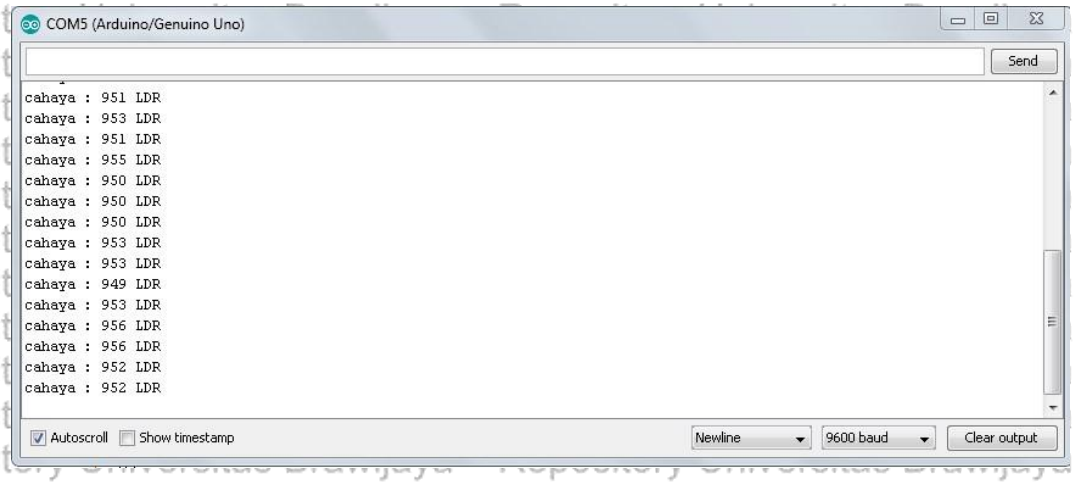


Gambar 6.1 Hasil Pengujian *Sensing Node Sensor* Suhu

Sesuai pada Gambar 6.1, format data yang akan dikirim oleh *node sensor* ke *node sink* yaitu nilai dari hasil *sensing* yang kemudian diberi label DHT dengan pemisah *whitespace* atau spasi.

6.1.1.2 Hasil Pengujian Kode KS-01 Pada *Node Sensor* Cahaya

Gambar 6.2 merupakan hasil dari pengujian kemampuan *node sensor* cahaya untuk dapat melakukan *sensing* terhadap lingkungan dan hasil *sensing* dapat ditampilkan ke *serial monitor* pada aplikasi Arduino IDE.



Gambar 6.2 Hasil Pengujian *Sensing Node Sensor* Cahaya

Sesuai pada Gambar 6.1, format data yang akan dikirim oleh *node sensor* ke node sink yaitu nilai dari hasil sensing yang kemudian diberi label LDR dengan pemisah *whitespace* atau spasi.

6.1.2 Hasil dan Analisis Kode KS-02

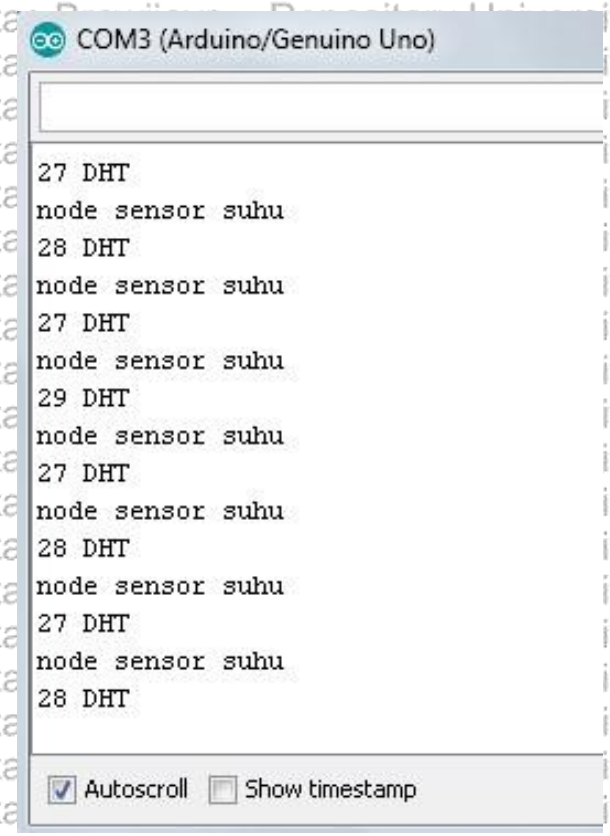
Pada pengujian kode KS-02 adalah untuk menguji apakah data dari *node sensor* dapat dikirim ke *node fog* melalui modul komunikasi nRF24l01. Penjelasan lengkap tentang pengujian kode KS-02 dapat dilihat pada Tabel 6.2.

Tabel 6.2 Deskripsi Pengujian Kode-KS-02

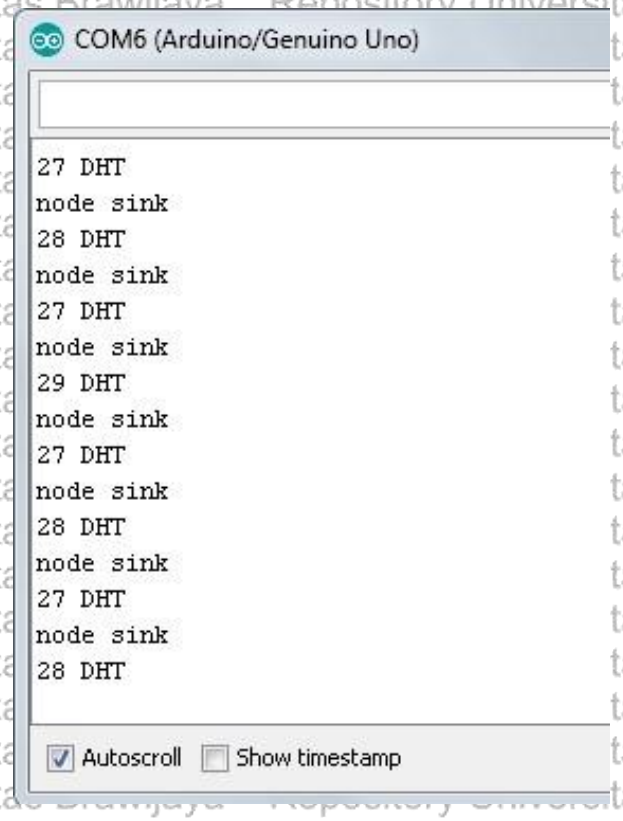
Kode	KS-02
Nama Pengujian	*Node sensor* mampu mengirimkan data ke *node fog* melalui nRF24l01
Tujuan	Mengetahui hasil pengujian dari kemampuan pengiriman data *sensing* mulai dari *node sensor* sampai ke *node fog* melalui modul komunikasi nRF24l01
Skenario Pengujian	1. Modul komunikasi nRF24l01 terpasang pada masing - masing node 2. Arduino Uno dan raspberry pi terpasang pada sumber daya 3. *Node sensor* mengirimkan data ke *node sink*, dan *node sink* meneruskan data ke *node fog* 4. Data yang dikirim dan diterima ditampilkan pada *serial monitor*
Hasil yang diharapkan	*Node sensor* dapat mengirimkan data ke *node fog* melalui nRF24l01
Hasil Pengujian	*Node sensor* dapat mengirimkan data ke *node fog* melalui nRF24l01

6.1.2.1 Hasil Pengujian Pengiriman *Data Sensing* Dari *Node Sensor* Suhu Sampai Ke *Node Fog*

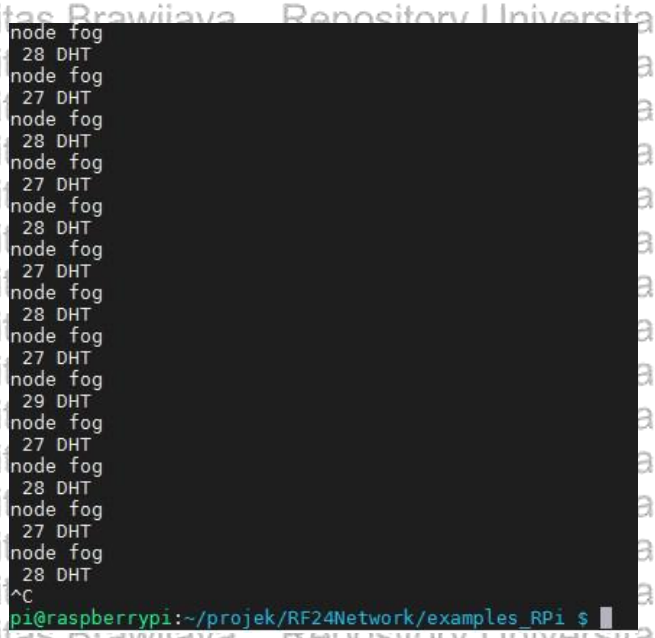
Pada Gambar 6.3 menunjukkan bahwa hasil pengujian pengiriman data *sensing* dari *node sensor* suhu ke *node sink* berhasil dengan menampilkan pada *serial monitor* berupa data yang dikirim oleh *node sensor* suhu dan menampilkan "*node sensor* suhu" yang menujukan data tersebut berada pada *node sensor* suhu. Gambar 6.4 menunjukkan bahwa hasil pengujian *node sink* mampu menerima data dari *node sensor* suhu dan mengirimkan data ke *node fog* dengan menampilkan pada *serial monitor* berupa data yang terima oleh *node sink* dari *node sensor* suhu dan menampilkan "*node sink*" yang menujukan data tersebut berada pada *node sink*. Gambar 6.5 menunjukkan bahwa hasil pengujian penerimaan *data sensing* suhu dari *node sink* ke *node fog* berhasil dengan menampilkan pada *terminal* berupa data yang dikirim oleh *node sink* dan menampilkan "*node fog*" yang menujukan data tersebut berada pada *node fog*.



Gambar 6.3 *Node Sensor* Suhu Mengirim Data Ke *Node Sink*



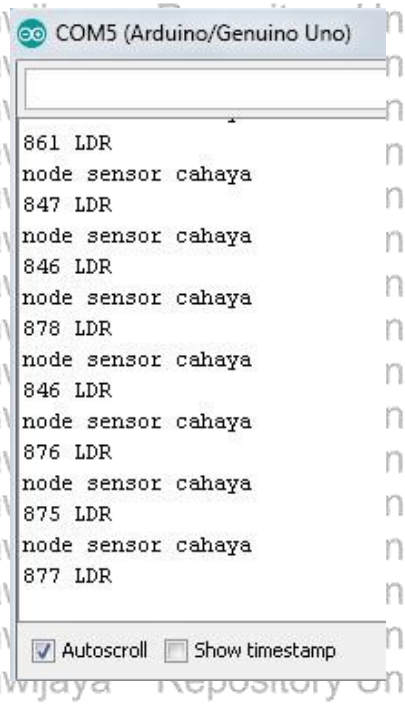
Gambar 6.4 *Node Sink* Mengirim dan Menerima Data Hasil *Sensing Node Sensor* Suhu



Gambar 6.5 *Node Fog* Menerima asil Data *Sensing Node Sensor* Suhu

6.1.2.2 Hasil Pengujian Pengiriman *Data Sensing* Dari *Node Sensor* Cahaya Sampai Ke *Node Fog*

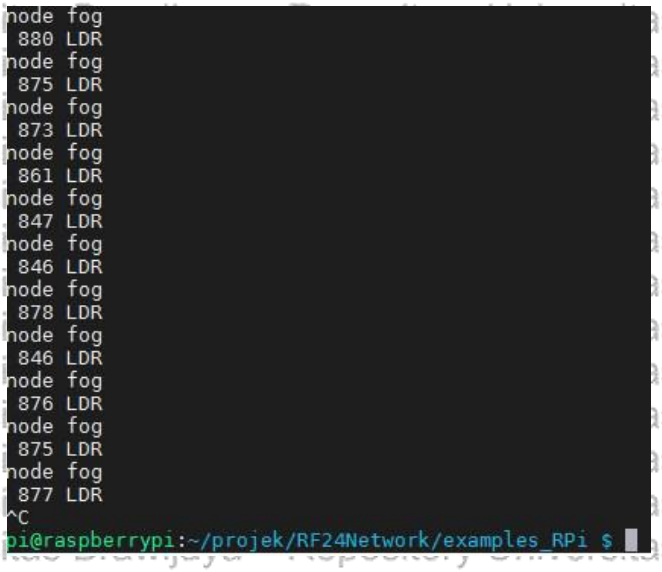
Gambar 6.6 menunjukkan bahwa hasil pengujian pengiriman data *sensing* dari *node sensor* cahaya ke *node sink* berhasil dengan menampilkan pada *serial monitor* berupa data yang dikirim oleh *node sensor* cahaya dan menampilkan "*node sensor* cahaya" yang menujukan data tersebut berada pada *node sensor* cahaya.



Gambar 6.6 ***Node Sensor* Cahaya Mengirim Data Ke *Node Sink***

Gambar 6.7 *Node Sink* Mengirim dan Menerima Hasil *Sensing Node Sensor* Cahaya

Gambar 6.7 menunjukkan bahwa hasil pengujian *node sink* mampu menerima data dari *node sensor* cahaya dan mengirimkan data ke *node fog* dengan menampilkan pada *serial monitor* berupa data yang terima oleh *node sink* dari *node sensor* suhu dan menampilkan "*node sink*" yang menujukan data tersebut berada pada *node sink*.



Gambar 6.8 *Node Fog* Menerima Hasil Data *Sensing Node Sensor* Cahaya

Gambar 6.8 menunjukkan bahwa hasil pengujian penerimaan *data sensing* cahaya dari *node sink* ke *node fog* berhasil dengan menampilkan pada *terminal* berupa data yang dikirim oleh *node sink* dan menampilkan "*node fog*" yang menujukan data tersebut berada pada *node fog*.

6.1.3 Hasil dan Analisis Kode KS-03

Pada pengujian kode KS-03 adalah untuk menguji apakah *node fog* dapat melakukan analisis data *sensing*. Penjelasan lengkap tentang pengujian kode KS-03 dapat dilihat pada Tabel 6.3.

Tabel 6.3 Deskripsi Pengujian Kode-KS-03

Kode	KS-03
Nama Pengujian	*Node fog* mampu melakukan analisis data hasil *sensing*
Tujuan	Mengetahui hasil dari pengujian kemampuan *node fog* dalam melakukan analisis data *sensing*
Skenario Pengujian	1. *Node fog* tersambung dengan sumber daya 2. *Node fog* menerima data *sensing* 3. *Node fog* analisis data hasil *sensing* 4. *Node fog* mengaktifkan LED 5. Menampilkan hasil analisis data *sensing*
Hasil yang diharapkan	*Node sensor* dapat melakukan akuisisi data dari modul sensor yang sudah terpasang.
Hasil Pengujian	*Node sensor* dapat melakukan akuisisi data dari modul sensor yang sudah terpasang.

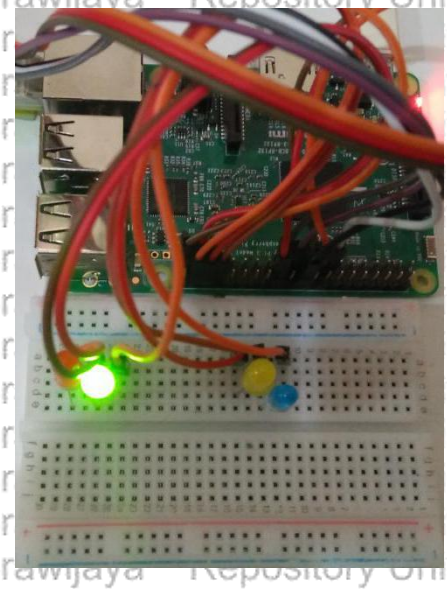
6.1.3.1 Hasil Pengujian Analisis Data Untuk Suhu

Tabel 6.4 merupakan hasil analisis dari data *sensing* yang dikirim oleh *node sensor* suhu. Pada Tabel 6.4 menunjukkan bahwa jika nilai dari data *sensing* suhu kurang dari nilai "30" (nilai *threshold*) sistem tidak akan menyalakan LED hijau akan tetapi jika nilai dari data *sensing* suhu melebihi dari nilai "30" (nilai *threshold*) maka sistem akan menyalakan LED hijau. Mekanisme pengukuran waktu *delay* analisis data dimulai dari *node fog* menerima data dari *node sink* sampai data tersebut selesai dianalisis oleh *node fog*, dari pengujian tersebut *delay* waktu menyalakan LED dari data mulai masuk pada *node fog* berkisar 508 dan 509 *millisecond*. Pada Gambar 6.9 menujukan bahwa sistem dapat menyalakan LED hijau jika nilai dari *sensing* suhu melebihi nilai "30" (*threshold*).

Tabel 6.4 Hasil Analisis Data Suhu Oleh *Node Fog*

No	Nilai	Hasil Analisis	Waktu Data Sampai (*Millisecond*)	Waktu Menjalankan Perintah (*Millisecond*)	Waktu Menjalankan Perintah - Waktu Data Sampai
1	29	led greed low	1978	2486	508

2	28	led greed low	6035	6544	509
3	29	led greed low	0090	0599	509
4	28	led greed low	4148	4656	508
5	29	led greed low	8203	8711	508
6	28	led greed low	2257	2765	508
7	31	led greed high	5927	6436	509
8	36	led greed high	49982	50491	509
9	35	led greed high	4038	4546	508
10	35	led greed high	58093	58602	509



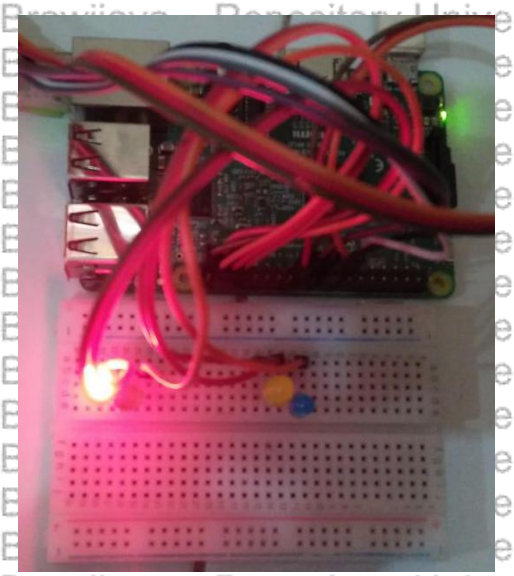
Gambar 6.9 Indikator Hasil Analisis Data Suhu

6.1.3.2 Hasil Pengujian Analisis Data Untuk Cahaya

Tabel 6.4 merupakan hasil analisis dari data *sensing* yang dikirim oleh *node sensor* cahaya. Pada Tabel 6.4 menunjukkan bahwa jika nilai dari data *sensing* cahaya kurang dari nilai "600" (nilai *threshold*) sistem tidak akan menyalakan LED merah akan tetapi jika nilai dari data *sensing* suhu melebihi dari nilai "600" (nilai *threshold*) maka sistem akan menyalakan LED merah. Mekanisme pengukuran waktu *delay* analisis data dimulai dari *node fog* menerima data dari *node sink* sampai data tersebut selesai dianalisis oleh *node fog, dari pengujian tersebut delay* waktu menyalakan LED dari data mulai masuk pada *node fog* berkisar 508 dan 509 millisecond. Pada Gambar 6.10 menujukan bahwa sistem dapat menyalakan LED merah jika nilai dari *sensing* suhu melebihi nilai "600" (*threshold*).

Tabel 6.5 Hasil Analisis Data Cahaya Oleh *Node Fog*

No	Nilai	Hasil Analisis	Waktu Data Sampai (*Millisecond*)	Waktu Menjalankan Perintah (*Millisecond*)	Waktu Menjalankan Perintah - Waktu Data Sampai
1	709	led red high	6576	7085	509
2	737	led red high	2426	2935	509
3	325	led red low	6483	6992	509
4	501	led red low	0543	1051	508
5	747	led red high	4603	5111	508
6	652	led red high	8659	9168	509
7	629	led red high	2222	2730	509
8	474	led red low	6279	6788	508
9	500	led red low	0337	0846	508
10	636	led red high	4396	4904	509



Gambar 6.10 Indikator Hasil Analisis Data Cahaya

6.1.4 Hasil dan Analisis Kode KS-04

Pada pengujian kode KS-04 adalah untuk menguji apakah *node fog* mampu menyimpan data hasil *sensing* ke database lokal SQLite. Penjelasan lengkap tentang pengujian kode KS-04 dapat dilihat pada Tabel 6.6.

Tabel 6.6 Deskripsi Pengujian Kode-KS-04

Kode	KS-04
Nama Pengujian	*Node fog* mampu menyimpan data hasil *sensing* ke database lokal SQLite.
Tujuan	Mengetahui hasil dari pengujian kemampuan *node fog* dalam melakukan penyimpanan data *sensing* ke dalam database SQLite
Skenario Pengujian	1. *Node fog* tersambung dengan sumber daya 2. *Node fog* menerima data *sensing* 3. *Node fog* menyimpan data *sensing* ke dalam database SQLite 4. Menampilkan data yang telah dimasukkan ke dalam database SQLite
Hasil yang diharapkan	*Node fog* dapat menyimpan data hasil *sensing* ke database lokal SQLite.
Hasil Pengujian	*Node fog* dapat menyimpan data hasil *sensing* ke database lokal SQLite.

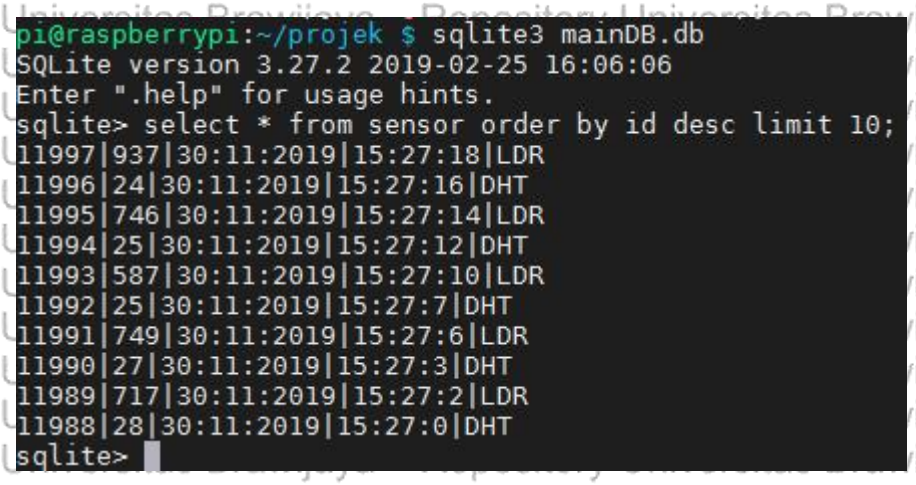
6.1.4.1 Data *Sensing*

Pada Tabel 6.7 merupakan hasil 10 kali percobaan data *sensing* yang dikirimkan oleh *node sensor* dan berhasil diterima oleh *node fog*. Mekanisme pengukuran waktu penyimpanan data hasil *sensing* ke database SQLite dimulai dari *node fog* menerima data dari *node sink* sampai data hasil *sensing* tersebut disimpan pada database SQLite. Dari percobaan tersebut dapat ditarik kesimpulan bahwa dari data masuk ke *node fog* sampai disimpan ke database SQLite memerlukan waktu 525 *millisecond* untuk yang tercepat dan 530 *millisecond* untuk yang terlama. Sedangkan pada Gambar 6.11 menunjukkan bahwa sistem mampu untuk memasukkan data *sensing* yang sudah diterima oleh *node fog* sesuai dengan pada Tabel 6.7 ke dalam database SQLite, oleh sebab itu berarti *node fog* mampu untuk menerima data *sensing* dari *node sink* dan menyimpannya ke dalam database SQLite.

Tabel 6.7 Data *Sensing* Yang Diterima Oleh *Node Fog*

No	Waktu	Nilai	Jenis *Sensor*	Waktu Terima (*Millisecond*)	Waktu Simpan SQLite (*Millisecond*)	Waktu Simpan SQLite - Waktu Terima
1	15:27:18	937	Cahaya	68827	69354	527
2	15:27:16	24	Suhu	70355	70880	525

3	15:27:14	746	Cahaya	72381	72911	530
4	15:27:12	25	Suhu	74412	74939	527
5	15:27:10	587	Cahaya	76440	76969	529
6	15:27:7	25	Suhu	78471	78998	527
7	15:27:6	749	Cahaya	80499	81024	525
8	15:27:3	27	Suhu	82025	82550	525
9	15:27:2	717	Cahaya	84552	85083	531
10	15:27:0	28	Suhu	86084	86610	526



Gambar 6.11 Hasil Memasukan Data *Sensing* Ke SQLite

6.1.4.2 Data Aktuator

Pada Tabel 6.8 merupakan hasil 10 kali percobaan data kontol manual yang dihasilkan dari melakukan aksi menekan tombol kontrol pada aplikasi web. Mekanisme pengukuran waktu penyimpanan data aktuator ke database SQLite dimulai dari aksi kontrol manual pada apliksi web sampai aksi kontrol manual tersebut disimpan pada database SQLite. Dari percobaan tersebut dapat ditarik kesimpulan bahwa dari data mulai masuk ke sistem sampai disimpan ke database SQLite memerlukan waktu 0.521094 *second* untuk yang tercepat dan 0.527909 *second* untuk yang terlama. pada Gambar 6.12 menunjukkan bahwa sistem mampu untuk memasukkan data kontrol yang sudah diterima oleh *node fog* ke dalam database SQLite yang sesuai pada Tabel 6.8, hal ini menunjukkan *node fog* mampu untuk menerima masukan kontrol oleh user ke dalam sistem dan menyimpannya ke dalam database SQLite.

Tabel 6.8 Data Kontrol Yang Diterima Oleh *Node Fog*

No	Waktu	Indikator Kipas	Indikator Lampu	Waktu Post	Waktu Simpan SQLite	Waktu Simpan SQLite -

				(Second)	(Second)	Waktu Post
1	18:28:47	Mati	Mati	39.32179	39.849699	0.527909
2	18:29:09	Mati	Hidup	48.281747	48.80367	0.521923
3	18:29:14	Mati	Mati	50.14028	50.675915	0.535635
4	18:29:18	Hidup	Mati	51.693475	52.215402	0.521927
5	18:29:22	Mati	Mati	52.997061	53.519726	0.522665
6	18:29:27	Mati	Hidup	54.7304	55.251494	0.521094
7	18:29:30	Mati	Mati	56.124427	56.648749	0.524322
8	18:29:35	Hidup	Mati	57.708206	58.233809	0.525603
9	18:29:39	Hidup	Hidup	59.291344	59.824936	0.533592
10	18:29:46	Hidup	Mati	0.603194	1.125034	0.52184

```
sqlite> select * from aktuator order by id desc limit 10;
206|mati|mati|31/10/2019|18:29:46
205|mati|hidup|31/10/2019|18:29:39
204|mati|mati|31/10/2019|18:29:35
203|hidup|mati|31/10/2019|18:29:30
202|mati|mati|31/10/2019|18:29:27
201|mati|hidup|31/10/2019|18:29:22
200|mati|mati|31/10/2019|18:29:18
199|hidup|mati|31/10/2019|18:29:14
198|hidup|hidup|31/10/2019|18:29:09
197|hidup|mati|31/10/2019|18:28:47
sqlite>
```

Gambar 6.12 Hasil Memasukan Data Kontrol Ke SQLite

6.1.5 Hasil dan Analisis Kode KS-05

Pada pengujian kode KS-05 adalah untuk menguji apakah *Node fog* dapat mengirim data hasil *sensing* dari SQLite ke *cloud* mongoDB Atlas. Penjelasan lengkap tentang pengujian kode KS-05 dapat dilihat pada Tabel 6.9.

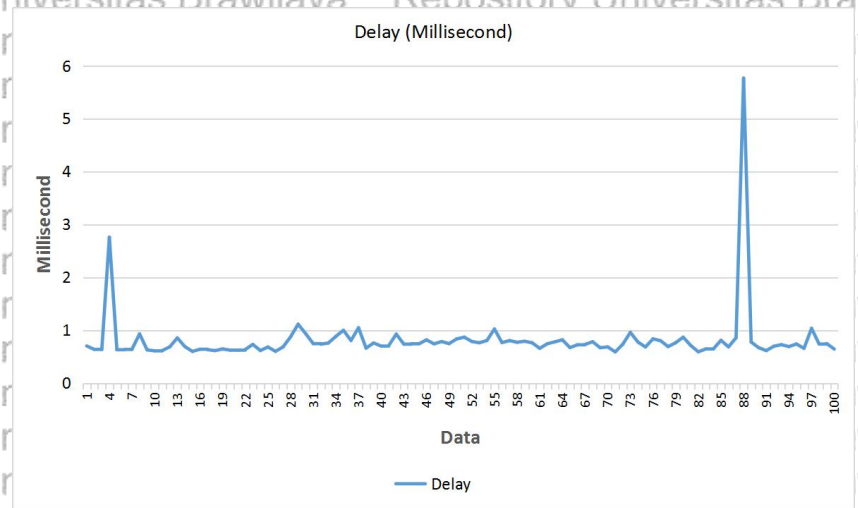
Tabel 6.9 Deskripsi Pengujian Kode-KS-05

Kode	KS-05
Nama Pengujian	*Node fog* mampu mengirim data hasil *sensing* dari SQLite ke *cloud* mongoDB Atlas
Tujuan	Mengetahui hasil dari pengujian kemampuan *node fog* dalam melakukan *backup* data dari database SQLite ke *cloud* mongoDB Atlas
Skenario Pengujian	1. *Node fog* tersambung dengan sumber daya

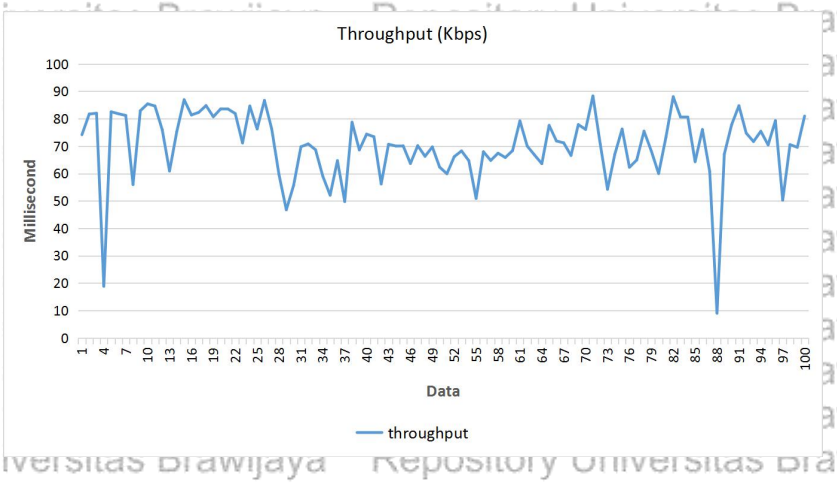
	2. *Node fog* tersambung internet 3. *Node fog* mengambil data *sensing* dari database SQLite 4. *Node fog* mengunggah data ke cloud mongoDB Atlas 5. Menampilkan data yang telah diunggah
Hasil yang diharapkan	*Node fog* dapat mengirim data hasil *sensing* dari SQLite ke cloud mongoDB Atlas
Hasil Pengujian	*Node fog* dapat mengirim data hasil *sensing* dari SQLite ke *cloud* mongoDB Atlas

Pada Gambar 6.13 menunjukkan delay pengiriman data dari *node fog* ke dalam *cloud* database mongoDB atlas dengan jumlah pengiriman sebanyak 100 kali pengiriman data, sedangkan pada Gambar 6.14 menunjukkan *throughput* pengiriman data dari *node fog* ke *cloud* database mongoDB Atlas dengan jumlah pengiriman sebanyak 100 kali pengiriman data juga. Mekanisme pengukuran waktu pengiriman data ke cloud database dimulai dari *node fog* mengambil data dari database SQLite sampai data tersebut sampai ke *cloud* database MongoDB.

Sesuai dengan grafik pada Gambar 6.13 dan Gambar 6.14 menunjukkan bahwa jika delay pengiriman data ke mongoDB atlas naik maka *throughput* pengiriman data ke cloud database mongoDB atlas mengalami penurunan nilai. Oleh sebab itu dapat disimpulkan bahwa besar kecilnya nilai *throughput* saat pengiriman data ke *cloud* database mongoDB Atlas dipengaruhi oleh besar kecilnya *delay* pengiriman data ke *cloud* database mongoDB Atlas. Sedangkan pada Gambar 6.14 merupakan potongan tampilan dari dasboard *cloud* database mongoDB Atlas di mana pada Gambar 6.14 menunjukkan data *sensing* yang disimpan oleh *cloud* database mongoDB Atlas.



Gambar 6.13 *Delay* pengiriman ke *cloud* database mongoDB atlas



Gambar 6.14 *throughput* pengiriman ke *cloud* database mongoDB atlas

QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("5dea120b78586f38675a454e")
id: 12131
nilai: 24
tgl: "6:12:2019"
jam: "15:32:10"
sensor: "DHT"
```

```
_id: ObjectId("5dea120e78586f38675a4551")
id: 12132
nilai: 933
tgl: "6:12:2019"
jam: "15:32:11"
sensor: "LDR"
```

```
_id: ObjectId("5dea121578586f38675a4557")
id: 12135
nilai: 23
tgl: "6:12:2019"
jam: "15:32:17"
sensor: "DHT"
```

```
_id: ObjectId("5dea121878586f38675a455a")
id: 12137
nilai: 25
tgl: "6:12:2019"
```

Gambar 6.15 Hasil Pengiriman Data *Sensing* Ke *Cloud* mongoDB Atlas

6.1.6 Hasil dan Analisis Kode KS-06

Pada pengujian kode KS-06 adalah untuk menguji apakah *Node fog* dapat menampilkan data hasil *sensing* melalui aplikasi web. Penjelasan lengkap tentang pengujian kode KS-06 dapat dilihat pada Tabel 6.10.

Tabel 6.10 Deskripsi Pengujian Kode-KS-06

Kode	KS-06
Nama Pengujian	*Node fog* mampu menampilkan data hasil *sensing* melalui aplikasi web secara *real time*
Tujuan	Mengetahui hasil dari pengujian kemampuan aplikasi web dalam melakukan menampilkan data hasil *sensing* secara *real time*.
Skenario Pengujian	1. Aplikasi *websocket* mengambil data *sensing* dari database SQLite 2. Aplikasi *websocket* mengirimkan data *sensing* ke aplikasi web 3. Aplikasi web menampilkan data *sensing*
Hasil yang diharapkan	*Node fog* dapat menampilkan data hasil *sensing* melalui aplikasi web secara *real time*
Hasil Pengujian	*Node fog* dapat menampilkan data hasil *sensing* melalui aplikasi web secara *real time*

```
sqlite> select * from sensor where sensor = "DHT" order by id desc limit 12;
4613|27|31:10:2019|19:54:35|DHT
4612|28|31:10:2019|19:54:33|DHT
4611|27|31:10:2019|19:54:31|DHT
4610|27|31:10:2019|19:54:29|DHT
4609|28|31:10:2019|19:54:27|DHT
4608|27|31:10:2019|19:54:25|DHT
4607|27|31:10:2019|19:54:23|DHT
4606|27|31:10:2019|19:54:21|DHT
4605|26|31:10:2019|19:54:19|DHT
4603|27|31:10:2019|19:54:17|DHT
4602|27|31:10:2019|19:53:47|DHT
4601|28|31:10:2019|19:53:45|DHT
sqlite>
```

Gambar 6.16 Data *Sensing* Suhu Yang Akan Dikirim Ke Aplikasi Web

```
sqlite> select * from sensor where sensor = "LDR" order by id desc limit 12;
4604|917|31:10:2019|19:54:18|LDR
4591|911|31:10:2019|19:53:26|LDR
4589|913|31:10:2019|19:53:24|LDR
4587|903|31:10:2019|19:53:22|LDR
4585|917|31:10:2019|19:53:20|LDR
4583|902|31:10:2019|19:53:18|LDR
4581|918|31:10:2019|19:53:15|LDR
4579|905|31:10:2019|19:53:13|LDR
4577|895|31:10:2019|19:47:42|LDR
4575|909|31:10:2019|19:47:40|LDR
4573|901|31:10:2019|19:47:38|LDR
4571|888|31:10:2019|19:47:36|LDR
sqlite>
```

Gambar 6.17 Data *Sensing* Cahaya Yang Akan Dikirim Ke Aplikasi Web

Pada Gambar 6.16 dan 6.17 merupakan data *sensing* pada database SQLite yang akan dikirimkan ke aplikasi web melalui mekanisme *websocket* dan pengambilan data *sensing* dilakukan secara terpisah antara data *sensing* suhu dan juga data *sensing* cahaya.

```
[(27, '19:54:35'), (28, '19:54:33'), (27, '19:54:31'), (27, '19:54:29'), (28, '19:54:27'), (27, '19:54:25'), (27, '19:54:23'), (27, '19:54:21'), (26, '19:54:19'), (27, '19:54:17'), (27, '19:53:47'), (28, '19:53:45'), [(917, '19:54:18'), (911, '19:53:26'), (913, '19:53:24'), (903, '19:53:22'), (917, '19:53:20'), (902, '19:53:18'), (918, '19:53:15'), (905, '19:53:13'), (895, '19:47:42'), (909, '19:47:40'), (901, '19:47:38'), (888, '19:47:36')]]
^C192.168.43.184 - - [31/Oct/2019 18:54:42] "GET / HTTP/1.1" 200 0 17.179758
wsgi exiting
(1385) wsgi exited, is_accepting=True
pi@raspberrypi:~/projek/api-websoket $
```

Gambar 6.18 Hasil Pengambilan Data *Sensing*

Gambar 6.19 Hasil Penerimaan Data *Sensing*

Pada Gambar 6.18 menunjukkan sistem mampu untuk mengambil data *sensing* pada database SQLite dan mengirimkan data tersebut melalui *websocket*. Sedangkan pada Gambar 6.19 menunjukkan sistem mampu untuk menampilkan data *sensing* yang dikirimkan melalui *websocket* ke dalam aplikasi web.

6.1.7 Hasil dan Analisis Kode KS-07

Pada pengujian kode KS-07 adalah untuk menguji apakah *Node fog* dapat melakukan kontrol melalui aplikasi web. Penjelasan lengkap tentang pengujian kode KS-07 dapat dilihat pada Tabel 6.11.

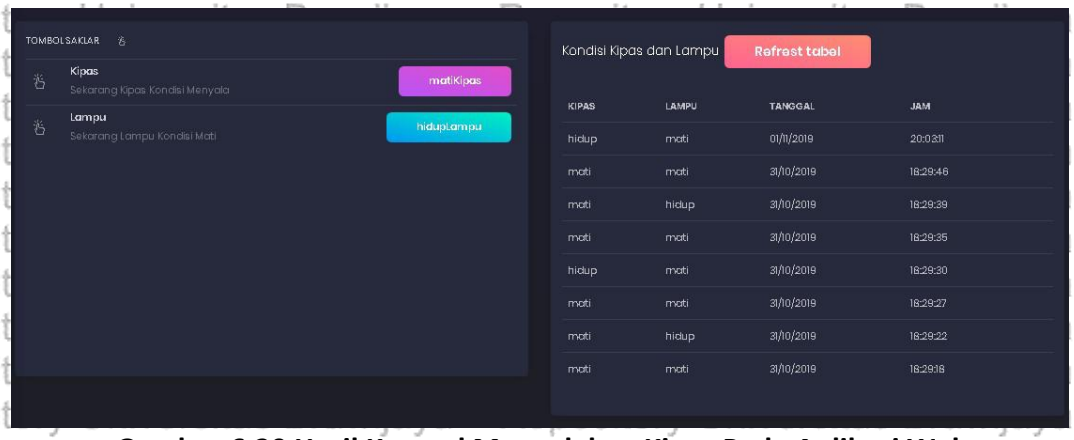
Tabel 6.11 Deskripsi Pengujian Kode-KS-06

Kode	KS-07

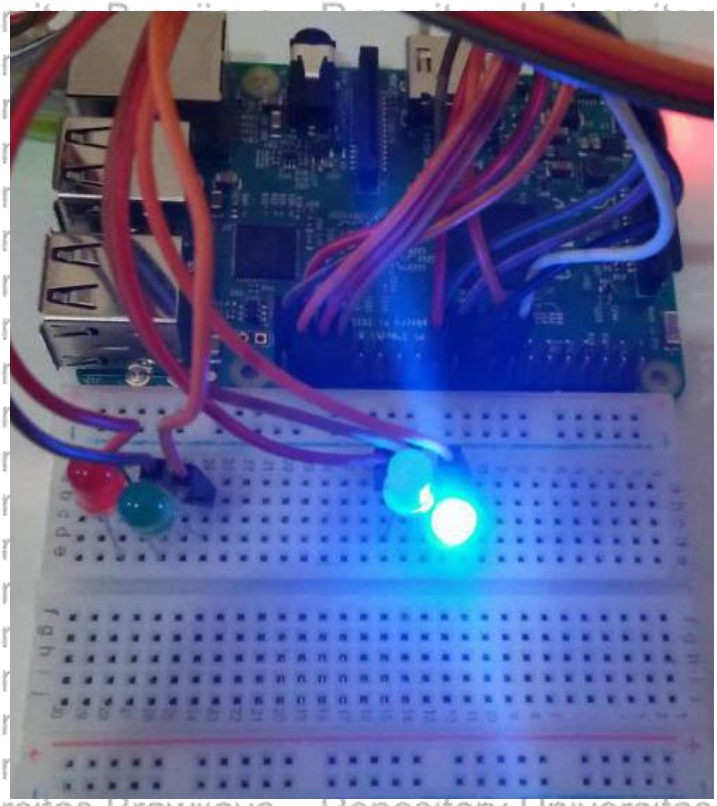
Nama Pengujian	*Node fog* dapat melakukan kontrol melalui aplikasi web
Tujuan	Mengetahui hasil dari pengujian kemampuan *node fog* dalam melakukan kontrol manual melalui aplikasi web.
Skenario Pengujian	1. Mengaktifkan Web server 2. Mengakses aplikasi web 3. Melakukan aksi pada tombol kontrol pada aplikasi web 4. Menampilkan hasil aksi tombol kontrol
Hasil yang diharapkan	*Node fog* dapat melakukan kontrol melalui aplikasi web.
Hasil Pengujian	*Node fog* dapat melakukan kontrol melalui aplikasi web.

6.1.7.1 Kontrol Manual Menyalakan Kipas

Pada Gambar 6.20 dan Gambar 6.21 menunjukkan bahwa sistem mampu untuk melakukan kontrol manual melalui aplikasi web dengan aksi yang dilakukan kipas menyala dan lampu mati. Gambar 6.20 menunjukkan kalau user melakukan aksi untuk menyalakan kipas, sedangkan pada Gambar 6.21 menunjukkan hasil kontrol manual oleh user dengan menyalakan LED berwarna biru yang bermakna indikator untuk menyalakan kipas sedang aktif.



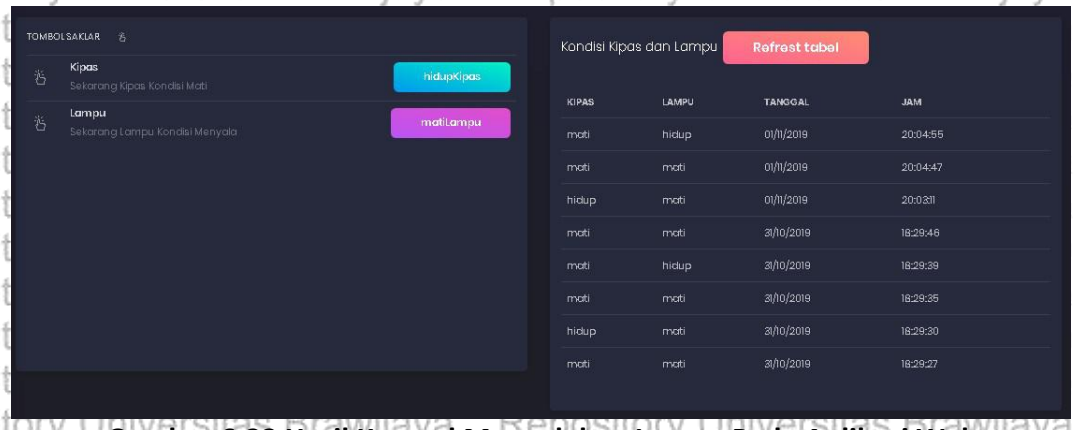
Gambar 6.20 Hasil Kontrol Menyalakan Kipas Pada Aplikasi Web



Gambar 6.21 Indikator Hasil Kontrol Kipas

6.1.7.2 Kontrol Manual Menyalakan Lampu

Pada Gambar 6.22 dan Gambar 6.23 menunjukkan bahwa sistem mampu untuk melakukan kontrol manual melalui aplikasi web dengan aksi yang dilakukan kipas mati dan lampu menyala. Gambar 6.22 menunjukkan kalau user melakukan aksi untuk menyalakan lampu, sedangkan pada Gambar 6.23 menunjukkan hasil kontrol manual oleh user dengan menyalakan LED berwarna kuning yang bermakna indikator untuk menyalakan lampu sedang aktif.

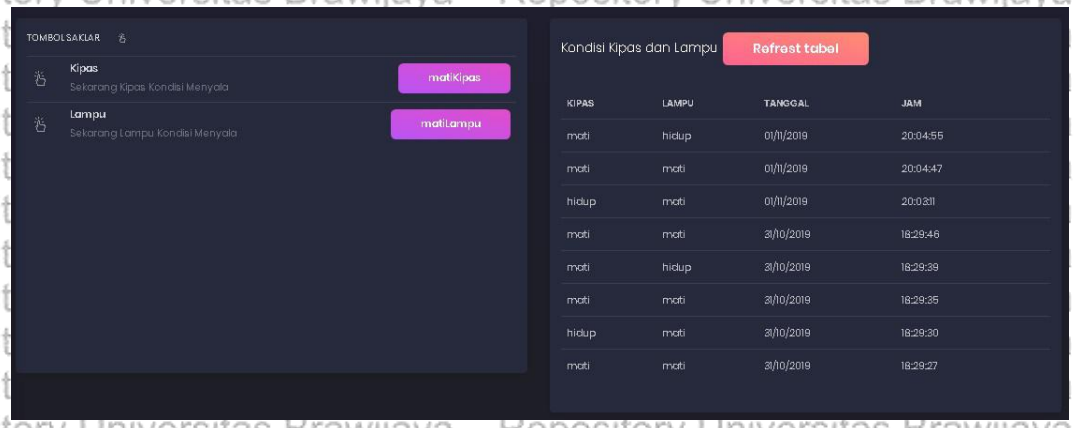


Gambar 6.22 Hasil Kontrol Menyalakan Lampu Pada Aplikasi Web

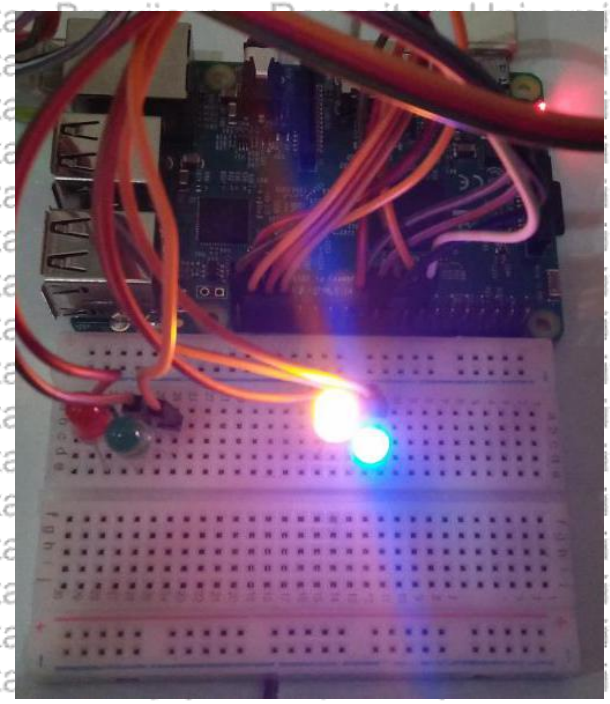
Gambar 6.23 Indikator Hasil Kontrol Lampu

6.1.7.3 Kontrol Manual Menyalakan Kipas Dan Lampu

Pada Gambar 6.24 dan Gambar 6.25 menunjukkan bahwa sistem mampu untuk melakukan kontrol manual melalui aplikasi web dengan aksi yang dilakukan kipas menyala dan lampu menyala. Gambar 6.24 menunjukkan kalau user melakukan aksi untuk menyalakan kipas dan lampu, sedangkan pada Gambar 6.24 menunjukkan hasil kontrol manual oleh user dengan menyalakan LED berwarna biru yang bermakna indikator untuk menyalakan kipas dan lampu sedang aktif.



Gambar 6.24 Hasil Kontrol Menyalakan Kipas Dan Lampu Pada Aplikasi Web



Gambar 6.25 Indikator Hasil Kontrol Kipas Dan Lampu

6.1.7.4 Analisis komputasi kontrol aktuator

Pada Tabel 6.12 merupakan hasil pengukuran kinerja *delay* komputasi dari kontrol aktuator melalui aplikasi web sampai sistem menyalakan LED sebanyak 10 kali percobaan. Mekanisme pengukuran waktu komputasi kontrol aktuator dimulai dari aksi kontrol manual pada aplikasi web sampai menyala atau matinya LED pada Raspberry pi. Sesuai pada Tabel 6.12 *delay* komputasi mulai dari masuknya perintah sampai sistem menyalakan LED memiliki *delay* terkecil bernilai 0.00205 *second* dan *delay* terbesar bernilai 0.007065 *second*.

Tabel 6.12 Waktu komputasi kontrol aktuator

No	Waktu Data Masuk (*Second*)	Waktu Perintah LED (*Second*)	Waktu Perintah LED - Waktu Data Masuk
1	39.32179	39.328855	0.007065
2	48.281747	48.283835	0.002088
3	50.14028	50.142484	0.002204
4	51.693475	51.69555	0.002075
5	52.997061	53.001053	0.003992
6	54.7304	54.73264	0.00224
7	56.124427	56.126606	0.002179
8	57.708206	57.712366	0.00416

9	59.291344	59.293394	0.00205
10	0.603194	0.605327	0.002133

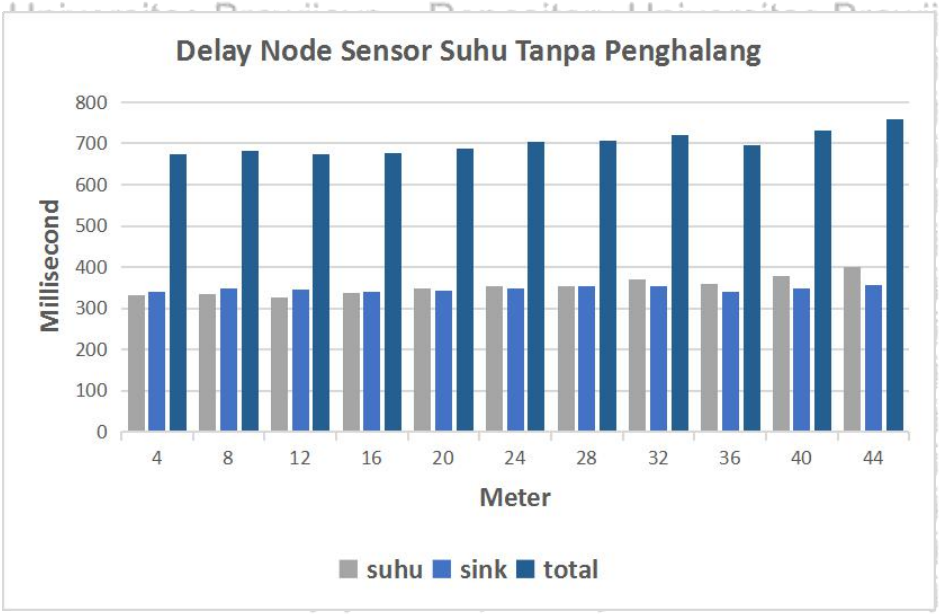
6.2 Pengujian Kinerja

Dalam pengujian kinerja ini dilakukan untuk mengetahui ke handalan sistem dalam melakukan pengiriman data dari *node sensor suhu* dan *node sensor* cahaya sampai ke *node fog* menggunakan modul komunikasi nRF24l01 dan *Coefficient of Variation* dari pengiriman data *sensor* menuju *cloud* mongoDB Atlas melalui *node fog*. Parameter yang digunakan pada pengujian kinerja modul komunikasi nRF24l01 yaitu menghitung *Delay* pengiriman, *Delay* Total, RTT (*Round trip-Time*), dan *Throughput* dengan variasi jarak 4m, 8m, 12m, 16m, 20m, 24m, 28m, 32m, 36m, 40m, dan 44m untuk tanpa penghalang, sedangkan ada penghalang dengan variasi jarak 4m, 8m, 16m, 20m, 24m, dan 28m dengan jumlah data yang dikirim sebanyak 100 kali dan besar paket data yang dikirim sebesar 45 *bytes*.

6.2.1 Pengujian Pengaruh Jarak Terhadap Delay

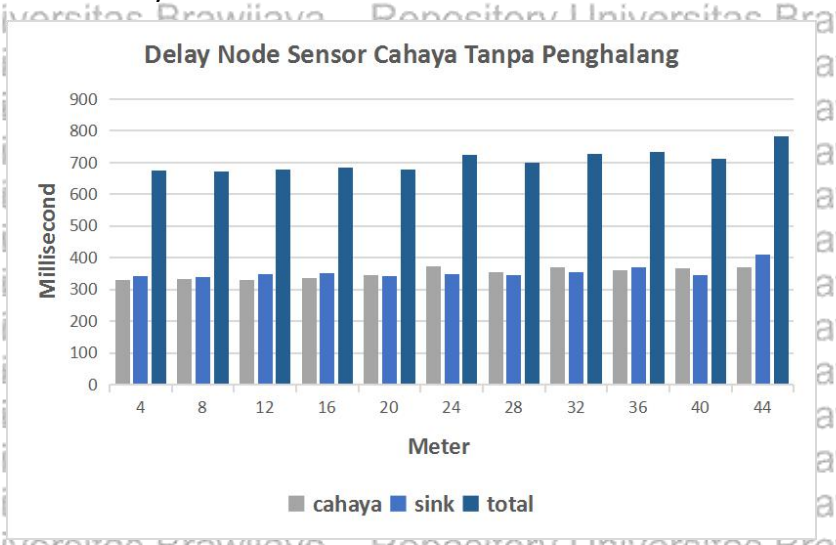
6.2.1.1 Tanpa Penghalang

Pada Gambar 6.26 menunjukkan grafik dari pengujian delay pengiriman *node sensor* suhu ke *node sink*, delay pengiriman *node sink* ke *node fog*, dan *delay* total yang dihasilkan dari penjumlahan delay pengiriman *node sensor*, dan *node sink*. Sesuai pada Gambar 6.26 delay pengiriman dari *node sensor* suhu, *node sink* dan *delay* total, nilai delay terkecil dari pengiriman *node sensor* suhu ke *node sink* terdapat pada jarak 4 meter yaitu bernilai 332 *millisecond*, sedangkan *delay* terbesar terdapat pada jarak 44 meter yaitu bernilai 400 *millisecond.* Pada *delay* pengiriman *node sink* ke *node fog*, jarak 36 meter yaitu bernilai 339 *millisecond* merupakan nilai terkecil dari delay pengiriman *node sink* ke *node fog,* sedangkan delay terbesar pada jarak 44 meter yaitu bernilai 355 *millisecond.* Sedangkan nilai terkecil dari delay total pengiriman terdapat pada jarak 12 meter yaitu bernilai 672 *millisecond*, sedangkan nilai terbesar terdapat pada jarak 44 meter yaitu bernilai 758 *millisecond*.



Gambar 6.1 *delay* pengiriman *node sensor* suhu tanpa penghalang

Pada Gambar 6.27 menunjukkan grafik dari pengujian delay pengiriman *node sensor* cahaya ke *node sink*, delay pengiriman *node sink* ke *node fog*, dan *delay* total yang dihasilkan dari penjumlahan delay pengiriman *node sensor*, dan *node sink*. Sesuai pada Gambar 6.27 delay pengiriman dari *node sensor* cahaya, *node sink* dan *delay* total, nilai *delay* terkecil dari pengiriman *node sensor* cahaya ke *node sink* terdapat pada jarak 4 meter yaitu bernilai 330 *millisecond*, sedangkan delay terbesar terdapat pada jarak 24 meter yaitu bernilai 373 *millisecond*. Pada delay pengiriman *node sink* ke *node fog*, jarak 8 meter yang bernilai 339 *millisecond* merupakan nilai terkecil dari delay pengiriman *node sink* ke *node fog,* sedangkan delay terbesar pada jarak 44 meter yaitu bernilai 408 *millisecond.* Sedangkan nilai terkecil dari *delay* total pengiriman terdapat pada jarak 8 meter yaitu bernilai 672 *millisecond*, sedangkan nilai terbesar terdapat pada jarak 44 meter yaitu bernilai 781 *millisecond*.

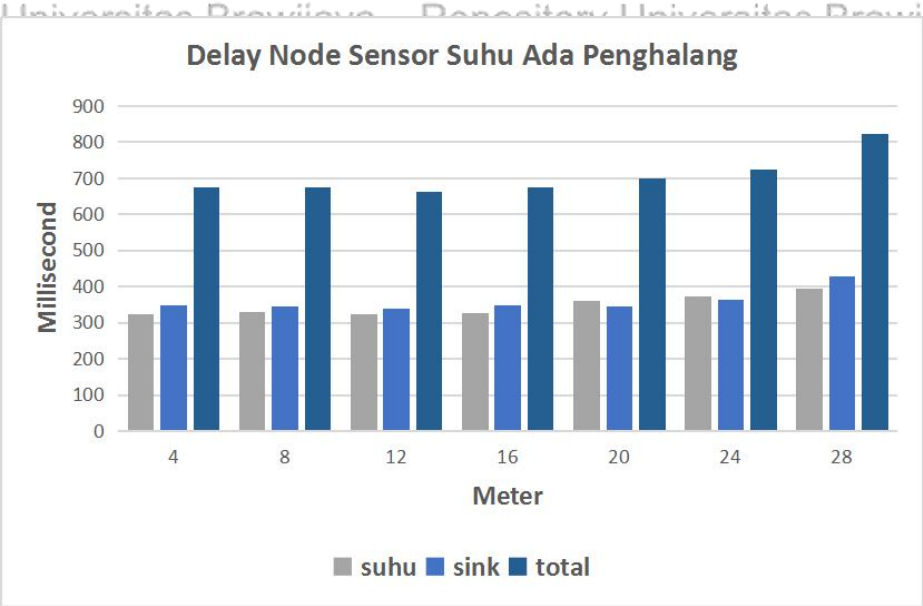


Gambar 6.2 *delay* pengiriman *node sensor* cahaya tanpa penghalang

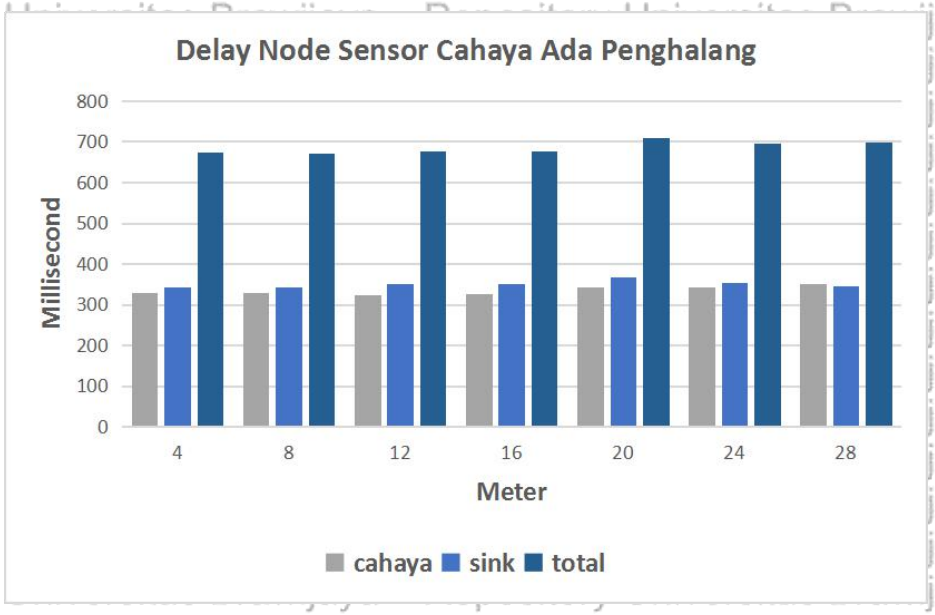
6.2.1.2 Ada Penghalang

Pada Gambar 6.28 menunjukkan grafik dari pengujian delay pengiriman *node sensor* suhu ke *node sink*, delay pengiriman *node sink* ke *node fog*, dan delay total yang dihasilkan dari penjumlahan delay pengiriman *node sensor*, dan *node sink*. Sesuai pada Gambar 6.28 delay pengiriman dari *node sensor* suhu, *node sink* dan *delay* total, nilai delay terkecil dari pengiriman *node sensor* suhu ke *node sink* terdapat pada jarak 12 meter yaitu bernilai 324 *millisecond*, sedangkan *delay* terbesar terdapat pada jarak 28 meter yaitu bernilai 395 *millisecond*. Pada *delay* pengiriman *node sink* ke *node fog*, jarak 12 meter yaitu bernilai 340 *millisecond* merupakan nilai terkecil dari delay pengiriman *node sink* ke *node fog,* sedangkan *delay* terbesar pada jarak 28 meter yaitu bernilai 427 *millisecond.* Sedangkan nilai terkecil dari delay total pengiriman terdapat pada jarak 12 meter yaitu bernilai 661 *millisecond*, sedangkan nilai terbesar terdapat pada jarak 44 meter yaitu bernilai 822 *millisecond*.

Pada Gambar 6.29 menunjukkan grafik dari pengujian delay pengiriman *node sensor* cahaya ke *node sink*, delay pengiriman *node sink* ke *node fog*, dan delay total yang dihasilkan dari penjumlahan delay pengiriman *node sensor*, dan *node sink*. Sesuai pada Gambar 6.29 *delay* pengiriman dari *node sensor* cahaya, *node sink* dan delay total, nilai delay terkecil dari pengiriman *node sensor* suhu ke *node sink* terdapat pada jarak 12 meter yaitu bernilai 324 *millisecond*, sedangkan delay terbesar terdapat pada jarak 28 meter yaitu bernilai 351 *millisecond.* Pada delay pengiriman *node sink* ke *node fog*, jarak 8 meter yaitu bernilai 342 *millisecond* merupakan nilai terkecil dari delay pengiriman *node sink* ke *node fog,* sedangkan delay terbesar pada jarak 20 meter yaitu bernilai 367 *millisecond.* Sedangkan nilai terkecil dari delay total pengiriman terdapat pada jarak 8 meter yaitu bernilai 670 *millisecond*, sedangkan nilai terbesar terdapat pada jarak 20 meter yaitu bernilai 710 *millisecond*.



Gambar 6.1 *delay* pengiriman *node sensor* suhu ada penghalang

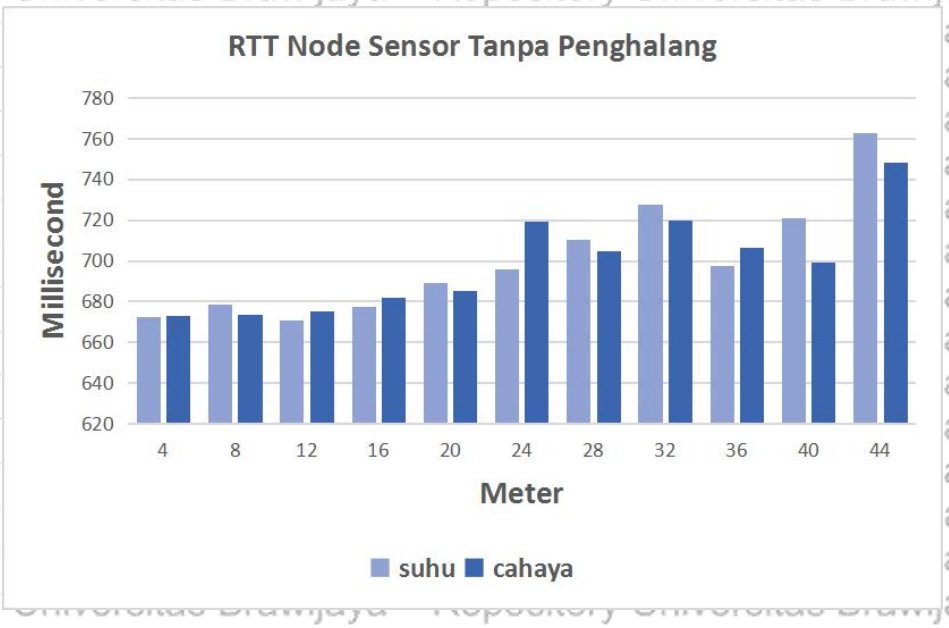


Gambar 6.2 *delay* pengiriman *node sensor* cahaya ada penghalang

6.2.2 Pengujian Pengaruh Jarak Terhadap RTT

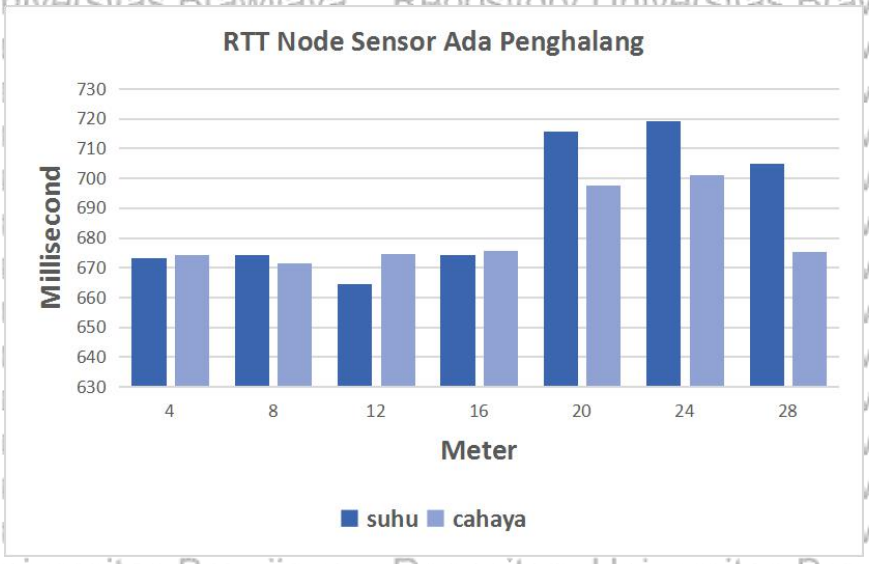
6.2.2.1 Pengaruh Tanpa Penghalang

Pada Gambar 6.30 menunjukkan grafik dari pengujian *round trip-time* (RTT) dari *node sensor* suhu dan *node sensor* cahaya. Sesuai pada Gambar 6.30 *round trip-time* (RTT) pada *node sensor* suhu terkecil terdapat pada jarak 12 meter dengan nilai 670 *millisecond* dan nilai terbesar pada jarak 44 meter dengan nilai 762 *millisecond*. Sedangkan *round trip-time* (RTT) terkecil pada *node sensor* cahaya terdapat pada jarak 4 meter dengan nilai 673 *millisecond* dan nilai terbesar terdapat pada jarak 44 meter dengan nilai 748 *millisecond*.



Gambar 6.1 RTT (*round trip-time*) pengiriman *node sensor* tanpa penghalang

6.2.2.2 Pengaruh Ada Penghalang

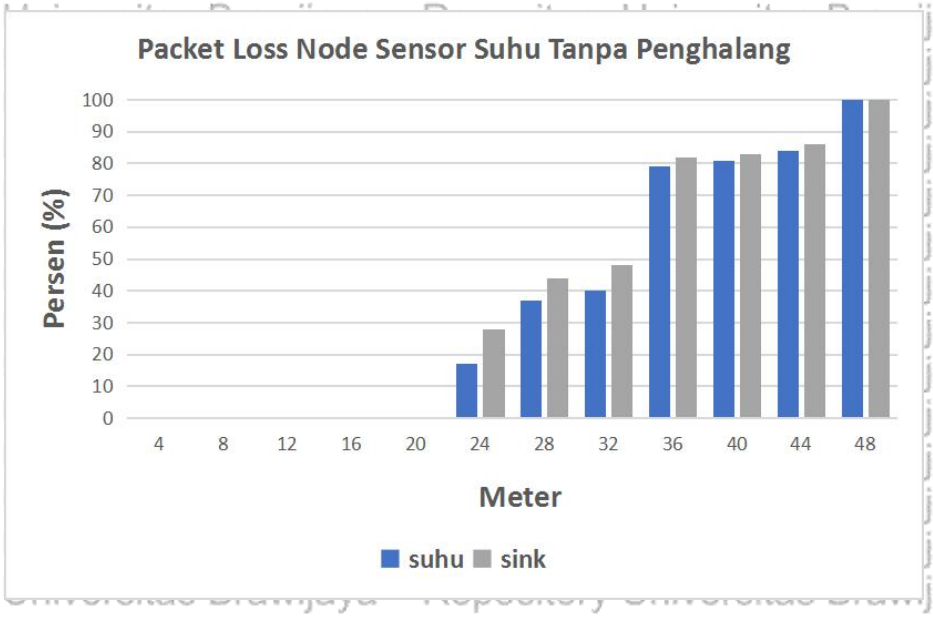


Gambar 6.1 RTT (*round trip-time*) pengiriman *node sensor* ada penghalang

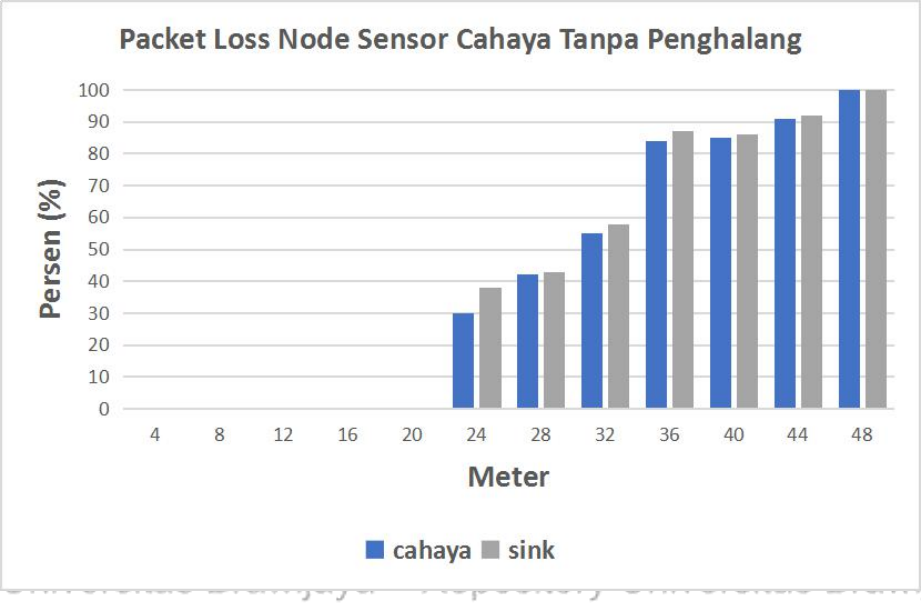
Pada Gambar 6.31 menunjukkan grafik dari pengujian RTT(*round trip-time)* dari *node sensor* suhu dan *node sensor* cahaya. Sesuai pada Gambar 6.31 RTT(*round trip-time*) pada *node sensor* suhu terkecil terdapat pada jarak 12 meter dengan nilai 664 *millisecond* dan nilai terbesar pada jarak 44 meter dengan nilai 719 *millisecond*. Sedangkan RTT(*round trip-time*) terkecil pada *node sensor* cahaya terdapat pada jarak 8 meter dengan nilai 671 *millisecond* dan nilai terbesar terdapat pada jarak 44 meter dengan nilai 701 *millisecond*.

6.2.3 Pengujian Pengaruh Jarak Terhadap Packet Loss

6.2.3.1 Pengaruh Tanpa Penghalang



Gambar 6.1 *packet loss* pengiriman *node sensor* suhu tanpa penghalang

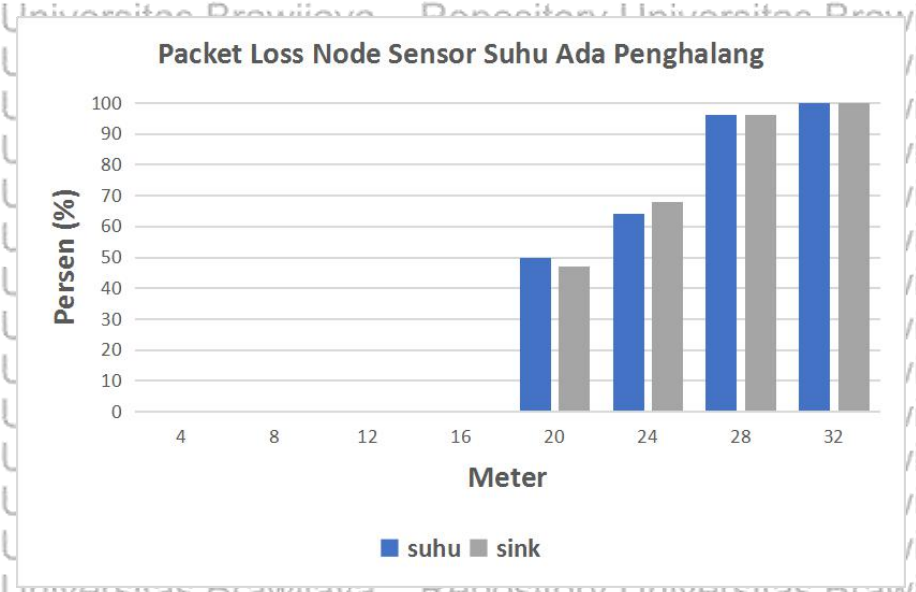


Gambar 6.2 *packet loss* pengiriman *node sensor* cahaya tanpa penghalang

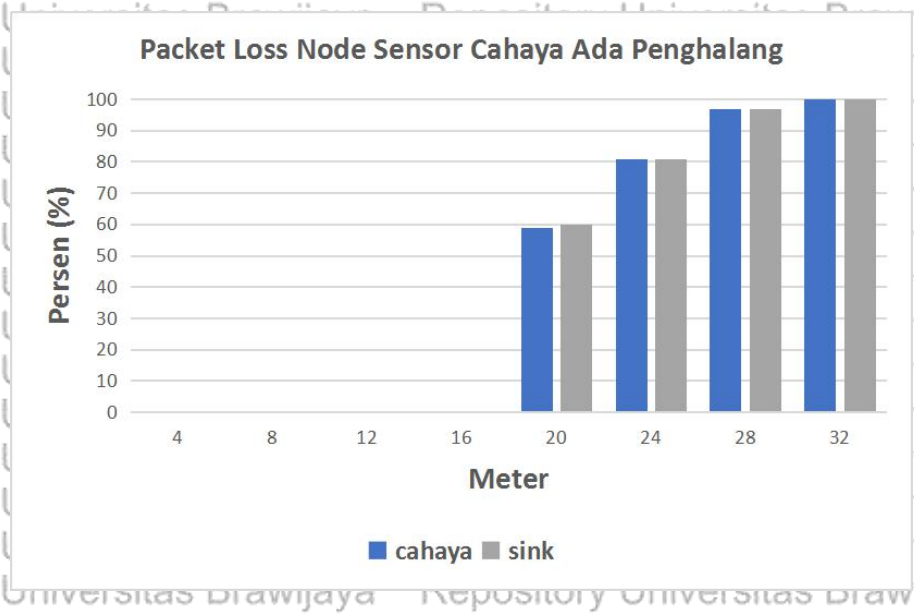
Pada Gambar 6.32 menunjukkan grafik dari pengujian *packet loss* dari *node sensor* suhu ke *node sink* dan *node sink* ke *node fog*. Sesuai pada Gambar 6.32 *packet loss* pada *node sensor* suhu pada jarak 4 sampai 20 meter dengan pengujian tanpa penghalang didapat hasil *packet loss* masih 0% sedangkan pada kondisi *packet loss* sebesar 100% saat pada jarak 48 meter.

Pada Gambar 6.33 menunjukkan grafik dari pengujian packet loss dari *node sensor* cahaya ke *node sink* dan *node sink* ke *node fog*. Sesuai pada Gambar 6.33 *packet loss* pada *node sensor* cahaya pada jarak 4 sampai 20 meter dengan pengujian tanpa penghalang didapat hasil *packet loss* masih 0% sedangkan pada kondisi *packet loss* sebesar 100% saat pada jarak 48 meter.

6.2.3.2 Pengaruh Ada Penghalang



Gambar 6.1 *packet loss* pengiriman *node sensor* suhu ada penghalang



Gambar 6.2 *packet loss* pengiriman *node sensor* cahaya ada penghalang

Pada Gambar 6.34 menunjukkan grafik dari pengujian *packet loss* dari *node sensor* suhu ke *node sink* dan *node sink* ke *node fog*. Sesuai pada Gambar 6.34 *packet loss* pada *node sensor* suhu pada jarak 4 sampai 12 meter dengan pengujian ada penghalang didapat hasil *packet loss* masih 0% sedangkan pada kondisi *packet loss* sebesar 100% saat pada jarak 32 meter.

Pada Gambar 6.35 menunjukkan grafik dari pengujian packet loss dari *node sensor* cahaya ke *node sink* dan *node sink* ke *node fog*. Sesuai pada Gambar 6.35 packet loss pada *node sensor* cahaya pada jarak 4 sampai 16 meter dengan pengujian ada penghalang didapat hasil *packet loss* masih 0% sedangkan pada kondisi *packet loss* sebesar 100% saat pada jarak 32 meter.

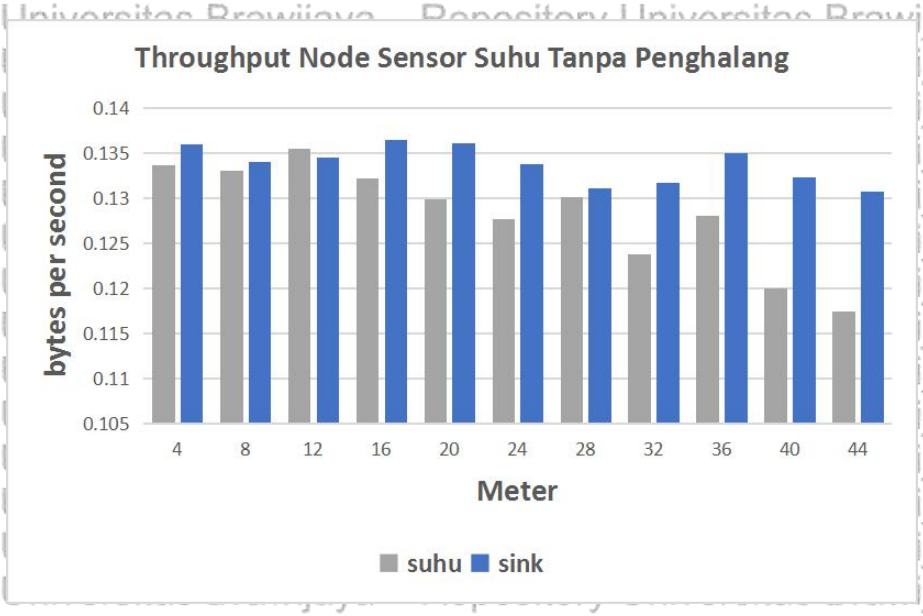
6.2.4 Pengujian Pengaruh Jarak Terhadap Throughput

6.2.4.1 Pengaruh Tanpa Penghalang

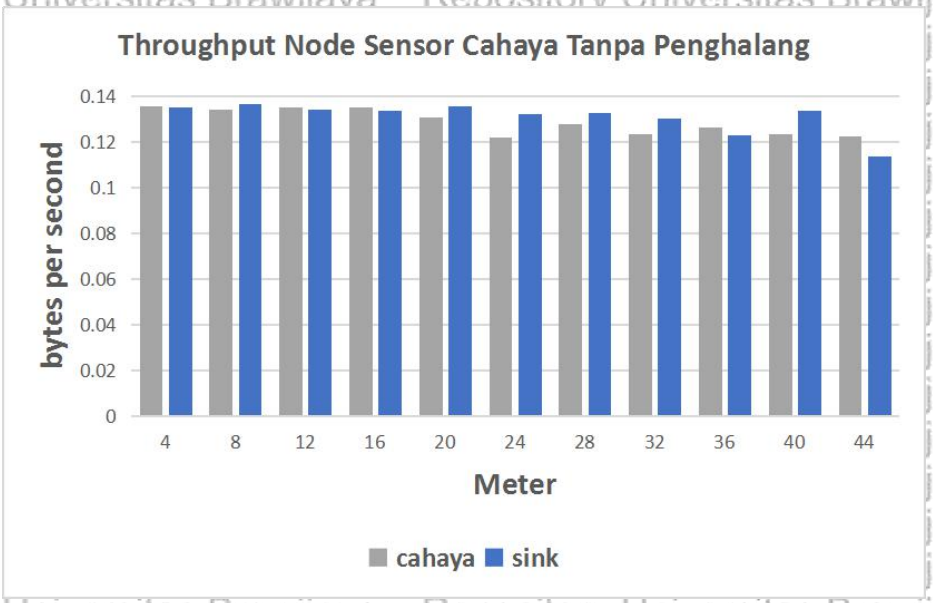
Pada Gambar 6.36 menunjukkan grafik dari pengujian *throughput* dari *node sensor* suhu ke *node sink* dan *node sink* ke *node fog*. Sesuai pada Gambar 6.36 nilai *throughput* tertinggi pada pengiriman *node sensor* suhu ke *node sink* terjadi pada jarak 12 meter dengan nilai throughput sebesar 0.1355 bps dan nilai *throughput* terkecil terdapat pada jarak 44 meter dengan nilai0.1175 bps. Sedangkan nilai throughput pada pengiriman *node sink* ke *node fog* terbesar terdapat pada jarak 16 meter dengan nilai *throughput* 0.1364 bps dan nilai terkecil terdapat pada jarak 44 meter dengan nilai *throughput* 0.1307 bps.

Pada Gambar 6.37 menunjukkan grafik dari pengujian *throughput* dari *node sensor* cahaya ke *node sink* dan *node sink* ke *node fog*. Sesuai pada Gambar 6.37 nilai throughput tertinggi pada pengiriman *node sensor* cahaya ke *node sink* terjadi pada jarak 4 meter dengan nilai throughput sebesar 0.1355 bps dan nilai *throughput* terkecil terdapat pada jarak 24 meter dengan nilai 0.1218 bps. Sedangkan nilai *throughput* pada pengiriman *node sink* ke *node fog* terbesar

terdapat pada jarak 8 meter dengan nilai *throughput* 0.1366 bps dan nilai terkecil terdapat pada jarak 44 meter dengan nilai *throughput* 0.11375 bps.



Gambar 6.1 *throughput* pengiriman *node sensor* suhu tanpa penghalang

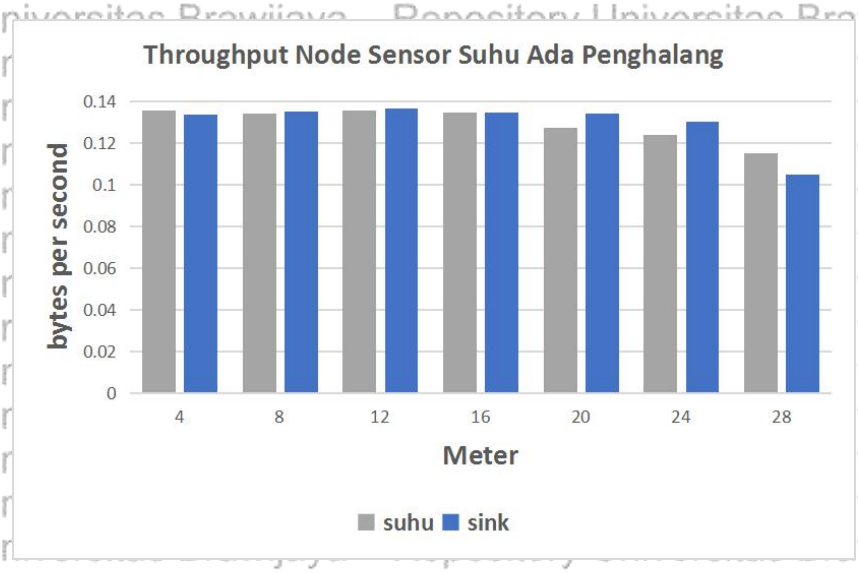


Gambar 6.2 *throughput* pengiriman *node sensor* cahaya tanpa penghalang

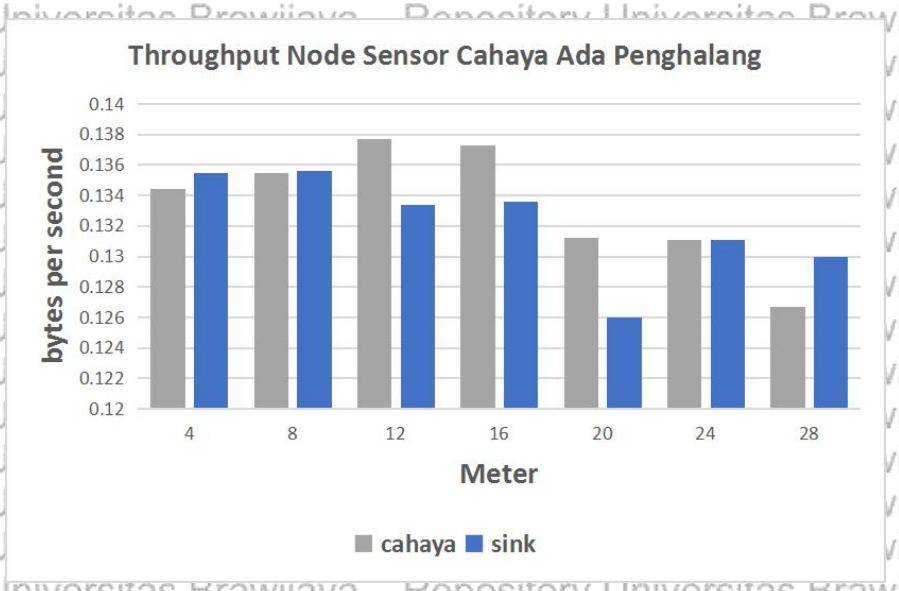
6.2.4.2 Pengaruh Ada Penghalang

Pada Gambar 6.38 menunjukkan grafik dari pengujian *throughput* dari *node sensor* suhu ke *node sink* dan *node sink* ke *node fog*. Sesuai pada Gambar 6.38 nilai *throughput* tertinggi pada pengiriman *node sensor* suhu ke *node sink* terjadi pada jarak 4 dan 12 meter dengan nilai *throughput* sebesar 0.1357 bps dan nilai *throughput* terkecil terdapat pada jarak 28 meter dengan nilai 0.115 bps. Sedangkan nilai *throughput* pada pengiriman *node sink* ke *node fog* terbesar

terdapat pada jarak 12 meter dengan nilai *throughput* 0.1377 bps dan nilai terkecil terdapat pada jarak 28 meter dengan nilai *throughput* 0.1367 bps.



Gambar 6.1 *throughput* pengiriman *node sensor* suhu ada penghalang



Gambar 6.2 *throughput* pengiriman *node sensor* suhu ada penghalang

Pada Gambar 6.39 menunjukkan grafik dari pengujian *throughput* dari *node sensor* cahaya ke *node sink* dan *node sink* ke *node fog*. Sesuai pada Gambar 6.39 nilai *throughput* tertinggi pada pengiriman *node sensor* cahaya ke *node sink* terjadi pada jarak 12 meter dengan nilai *throughput* sebesar 0.1377 bps dan nilai *throughput* terkecil terdapat pada jarak 28 meter dengan nilai 0.126 bps. Sedangkan nilai *throughput* pada pengiriman *node sink* ke *node fog* terbesar terdapat pada jarak 8 meter dengan nilai *throughput* 0.1356 bps dan nilai terkecil terdapat pada jarak 20 meter dengan nilai *throughput* 0.126 bps.

6.2.5 Pengujian Variasi Waktu Pengiriman Data Dari *Node Sensor* Sampai ke *Cloud* Database

6.2.5.1 Pengaruh Tanpa Penghalang

Pada tabel 6.13 menunjukkan rata-rata waktu pengiriman data dari *node sensor* suhu dan cahaya menuju *cloud* database mongoDB atlas berdasarkan jarak tanpa penghalang. Sesuai dengan tabel 6.13 rata-rata pengujian waktu pengiriman data sebesar 704.29, standar deviasi dari pengujian waktu pengiriman data sebesar 29.63, dan hasil *Coefficient of Variation* pada proses pengiriman data dari *node sensor* ke *cloud* database mongoDB atlas dengan tanpa penghalang sebesar 0.04 yang menunjukkan penyebaran waktu pengiriman data dari *node sensor* sampai ke cloud memiliki variasi yang rendah.

Tabel 6.1 Variasi Pengiriman Data Ke *Cloud* Tanpa Penghalang

Jarak	Waktu Pengiriman (Millisecond)
4	674
8	678
12	676
16	681
20	684
24	715
28	705
32	724
36	715
40	723
44	771
Standar Deviasi:	29.63
Rata-rata:	704.29
Coefficient of Variation:	0.04

6.2.5.2 Pengaruh Ada Penghalang

Pada tabel 6.14 menunjukkan rata-rata waktu pengiriman data dari *node sensor* suhu dan cahaya menuju *cloud* database mongoDB atlas berdasarkan jarak dengan ada penghalang. Sesuai dengan tabel 6.14 rata-rata pengujian waktu pengiriman data sebesar 695.78 standar deviasi dari pengujian waktu pengiriman data sebesar 33.14, dan hasil *Coefficient of Variation* pada proses pengiriman data dari *node sensor* ke *cloud* database mongoDB atlas dengan ada

penghalang sebesar 0.05 yang menunjukkan penyebaran waktu pengiriman data dari *node sensor* sampai ke *cloud* memiliki variasi yang rendah.

Tabel 6.1 Variasi Pengiriman Data Ke *Cloud* Ada Penghalang

Jarak	Waktu Pengiriman (Millisecond)
4	674
8	673
12	669
16	677
20	705
24	711
28	761
Standar Deviasi:	33.14
Rata-rata:	695.78
Coefficient of Variation:	0.05

BAB 7 PENUTUP

Pada Bab penutup akan menjelaskan mengenai kesimpulan dan saran mengenai penelitian yang telah dilakukan. Pada bab ini terdapat sub-bab kesimpulan yang menjelaskan atau menjawab tentang rumusan masalah yang telah diajukan sebelumnya dan sub-bab Saran yang menjelaskan mengenai saran-saran untuk kepentingan pengembangan sistem ke depan.

7.1 Kesimpulan

Berdasarkan dari hasil perancangan, implementasi, dan pengujian sistem yang sudah dilakukan sebelumnya, dapat ditarik kesimpulan sebagai berikut:

1. Penerapan *fog computing* pada rumah cerdas dapat menggunakan *mini* pc seperti raspberry pi. Pada raspberry pi sebagai *node fog* dilakukan konfigurasi untuk dapat menyimpan data, menyajikan data, analisis data, dan kontrol sistem secara lokal supaya membantu kinerja dari cloud, selain itu raspberry pi sebagai *node fog* di konfigurasi untuk dapat meneruskan data ke cloud. Sedangkan pada proses komunikasi dengan *node sensor* dan *node sink*, *node fog* dapat menggunakan modul komunikasi nRF24l01.
2. Pengaruh jarak dan dengan tanpa ada penghalang pada pengukuran kinerja *delay* didapat nilai terbaik pada jarak 12 meter yaitu dengan nilai rata-rata 448.52 *millisecond* untuk pengiriman data dari *node sensor* suhu sampai ke *node fog,* sedangkan untuk pengiriman data dari *node sensor* cahaya sampai ke *node fog* didapat jarak terbaik yaitu 8 meter dengan nilai rata-rata 448.0066667 *millisecond*. Sedangkan pengaruh jarak dan dengan ada penghalang pada pengukuran kinerja *delay* didapat nilai terbaik pada jarak 12 meter yaitu dengan nilai rata-rata 442.1272054 *millisecond* untuk pengiriman data dari *node sensor* suhu sampai ke *node fog,* sedangkan untuk pengiriman data dari *node sensor* cahaya sampai ke *node fog* didapat jarak terbaik yaitu 8 meter dengan nilai rata-rata 447.0266667 *millisecond*. Oleh sebab itu dapat ditarik kesimpulan bahwa besar kecilnya nilai dari kinerja *delay* tidak dipengaruhi oleh jarak pengiriman dan ada penghalang atau tanpa penghalang.
3. Pengaruh jarak dan dengan tanpa ada penghalang pada pengukuran kinerja RTT (*round trip-time*) didapat nilai terbaik pada jarak 12 meter yaitu dengan nilai rata-rata 670.84 *millisecond* untuk pengiriman data dari *node sensor* suhu sampai ke *node fog,* sedangkan untuk pengiriman data dari *node sensor* cahaya sampai ke *node fog* didapat jarak terbaik yaitu 4 meter dengan nilai rata-rata 673.15 *millisecond*. Sedangkan pengaruh jarak dan dengan ada penghalang pada pengukuran kinerja RTT (*round trip-time*) didapat nilai terbaik pada jarak 12 meter yaitu dengan nilai rata-rata 664.49 *millisecond* untuk pengiriman data dari *node sensor* suhu sampai ke *node fog,* sedangkan untuk pengiriman data dari *node sensor* cahaya sampai ke *node fog* didapat jarak terbaik yaitu 8 meter dengan nilai rata-rata 671.39

millisecond. Oleh sebab itu dapat ditarik kesimpulan bahwa besar kecilnya nilai dari kinerja RTT (*round trip-time*) tidak dipengaruhi oleh jarak pengiriman dan ada penghalang atau tanpa penghalang.

4. Pengaruh jarak dan dengan tanpa ada penghalang pada pengukuran kinerja *throughput* didapat nilai terbaik pada jarak 12 meter yaitu dengan nilai rata-rata 0.135 *bytes per second* untuk pengiriman data dari *node sensor* suhu sampai ke *node fog,* sedangkan untuk pengiriman data dari *node sensor* cahaya sampai ke *node fog* didapat jarak terbaik yaitu 4 meter dengan nilai rata-rata 0.13535 *bytes per second*. Sedangkan pengaruh jarak dan dengan ada penghalang pada pengukuran kinerja *throughput* didapat nilai terbaik pada jarak 12 meter yaitu dengan nilai rata-rata 0.1362 *bytes per second* untuk pengiriman data dari *node sensor* suhu sampai ke *node fog,* sedangkan untuk pengiriman data dari *node sensor* cahaya sampai ke *node fog* didapat jarak terbaik yaitu 8 dan 12 meter dengan nilai rata-rata 0.13555 *bytes per second*. Oleh sebab itu dapat ditarik kesimpulan bahwa besar kecilnya nilai dari kinerja *throughput* tidak dipengaruhi oleh jarak pengiriman dan ada penghalang atau tanpa penghalang.
5. Pengaruh jarak dan dengan tanpa ada penghalang pada pengukuran kinerja *paket loss* didapat jarak terbaik yaitu 4, 8, 12, 16, dan 20 meter dengan packet loss sebesar 0% baik pengiriman data dari *node sensor* suhu atau *node sensor* cahaya sampai ke *node fog*. Sedangkan pengaruh jarak dan dengan ada penghalang pada pengukuran kinerja *paket loss* didapat jarak terbaik yaitu 4, 8, 12, dan 16 meter dengan packet loss sebesar 0% baik pengiriman data dari *node sensor* suhu atau *node sensor* cahaya sampai ke *node fog*. Oleh sebab itu dapat ditarik kesimpulan bahwa besar kecilnya nilai dari kinerja *packet loss* dipengaruhi oleh jarak pengiriman dan ada penghalang atau tanpa penghalang, semakin besar jangkauan pengiriman dan banyaknya penghalang akan meningkatkan nilai *packet loss*.
6. Variasi waktu pengiriman data *sensing* dari *node sensor* sampai *cloud* pada jarak pengiriman mana pun dan dengan ada penghalang atau tidak ada penghalang tidak mengalami banyak perubahan waktu pengiriman dalam kondisi akses internet bagus, hal ini disebabkan nilai *Coefficient of Variation* untuk tanpa penghalang sebesar 0.019550617 dan ada penghalang sebesar 0.021990743.

7.2 Saran

Berdasarkan dari hasil perancangan, implementasi, pengujian, dan kesimpulan pada penelitian ini, dihasilkan beberapa saran untuk pengembangan lanjut dari penelitian berikutnya sebagai berikut:

1. Dapat dikembangkan modul komunikasi selain nRF24l01 seperti LoRa, ZigBee, atau menggunakan protokol komunikasi COAP, MQTT, BLE, dan RPC sebagai komunikasi data antar node supaya dapat mengetahui perbedaan kinerja nRF24l01 dengan metode komunikasi lain.

2. Perlu dilakukan analisis data pada *node fog* dengan algoritma *machine learning* seperti Naive bayes, K-NN, Jaringan saraf tiruan, SVM, atau algoritma *machine learning* lain supaya analisis data lebih akurat.

3. Perlu penambahan *node sensor* selain *sensor* suhu dan cahaya pada rumah cerdas seperti sensor kelembaban, arus listrik, atau *sensor* pendukung rumah cerdas lain.

4. Dapat dikembangkan pada *node fog* untuk membuat *cloud gateway* untuk mendukung heterogen dari teknologi *cloud* dan banyaknya penyedia *cloud* yang tersedia.

DAFTAR REFERENSI

Afif, T., Bhawiyuga, A., Siregar, R.A., 2019 Implementasi Perangkat Gateway Untuk Pengiriman Data Sensor Dari Lapangan Ke Pusat Data Pada Jaringan *Wireless Sensor Network* Berbasis Perangkat nRF24L01. Jurnal Pengembangan Teknlogi Informasi dan Ilmu Komputer

Alzaid, H. 2011. *Security of Self-Organizing Networks: MANET, WSN, WMN, VANET*. [e-book]. AL-Sakib Khan Pathan. Tersedia melalui: <http://index-of.es/Hack/Security%20of%20Self-Organizing%20Networks%20MANET,%20WSN,%20WMN,%20VANET.pdf> [Diakses 20 Desember 2019]

arduino, 2019. *arduino - software.* [Online] Available at: https://www.arduino.cc [Diakses 23 Februari 2019].

arduino, 2019. *Getting Started with the Arduino Nano.* [Online] Available at: https://www.arduino.cc/en/Guide/ArduinoNano [Diakses 23 Februari 2019].

F, Javier., M, Higinio., J, Antonio., V, Bruno. 2018. *Deployment of IoT Edge and Fog Computing Technologies to Develop Smart Building Services*. In: MDPI, 2018 *The Deployment of IoT in Smart Buildings*. Alicante, Spanyol, 23 October 2018. MDPI: *journals*

Flask. 2020. *Welcome to Flask*. [foto] (koleksi Flask)

Flask, 2020. *Welcome to Flask* . [Online] Available at: http://flask.palletsprojects.com/en/1.1.x/ [Diakses 14 Januari 2020].

Homeautotechs. 2018. *smart home technolog*. [foto] (koleksi Homeautotechs)

L, Mingfu., L, Hung-Ju. 2015. *Design and Implementation of Smart Home Control Systems Based on Wireless Sensor Networks and Power Line Communications*. In: IEEE, 2015 *Transactions on Industrial Electronics*. *ministry of science and technology,* Taiwan, 10 December 2014. IEEE Xplore: IEEE

Ibrahim, D.M., Primananda, R., Data, M., 2018 Perbandingan Performa Database Apache HBase dan Apache Cassandra Sebagai Media Penyimpanan Data S ensor Internet of Things Jurnal Pengembangan Teknlogi Informasi dan Ilmu Komputer

Ichsan, M.H.H., Kurniawan, W., Akbar, S.R. 2018 *Udp pervasive protocol integration with IoT for smart home environment using labview*. *International Journal of Electrical and Computer Engineering*.

Ichsan, M.H.H., Kurniawan, W., Setyawan, G.E. 2019 WSN *performance based on node placement by genetic algorithm at smart home environment*. Jurnal TELKOMNIKA

Instructables. 2017. DHT11. [foto] (koleksi RudraNarayanG)

Masykur, Fauzan. Prasetiyowati, F, 2016. Aplikasi Rumah Pintar (*Smart Home*) Pengendali Peralatan Elektronik Rumah Tangga Berbasis Web. [online] Tersedia di: <http://jtiik.ub.ac.id/index.php/jtiik/article/view/156> [Diakses 15 November2019].

Medium. 2017. Logo SQLite. [foto] (koleksi Mahesa Iqbal Ridwansyah)

MongoDB . 2019. Mongodb Atlas Create Cluster. [foto] (koleksi MongoDB)

MongoDB, 2019. *The database for modern applications* . [Online] Available at: https://www.mongodb.com/ [Diakses 25 Februari 2019].

Nordic Semiconductor ASA., 2006. *Single Chip 2.4 GHz Transceiver.* [Online] Available at:http://data.mecheng.adelaide.edu.au/robotics/WWW_Devs/Dragon12/rtmc9S1 2Target/nRF24L01_prelim_prod_spec_1_2.pdf [Diakses 10 Agustus 2019].

OpenFog. 2019. *OpenFog Consortium I* [Online] Available at: https://www.openfogconsortium.org/ [Diakses 25 Februari 2019].

P, Anand., P, Hameed., H, Won-Hwa., S, HyunCheol., R, Seungmin. 2018. *Fog Computing-Based IoT for Health Monitoring System*. In: Hindawi, 2018 *Journal of Sensors*. daegu, Korea Selatan, 22 Oktober 2018. Hindawi: *journals*

Raspberrypi, 2018. *THE OFFICIAL Raspberry Pi Beginner's Guide How to use your new compute.* [e-book]. Raspberrypi. Tersedia melalui: <https://www.raspberrypi.org/magpi-issues/Beginners_Guide_v1.pdf> [Diakses 20 Desember 2019]

Rovai, Marcelo. 2018. Python WebServer *With* Flask and Raspberry Pi. [Online] Tersedia di: <https://towardsdatascience.com/python- webserver-with-flask-and-raspberry-pi-398423cc6f5d> [Diakses 20 November2019].

Rangkuti Syahban, 2016, Arduino dan Proteus Simulasi dan Praktik, Bandung, Informatika

ScienceProg, 2019. *Powerful open source SQLite manager for Raspberry Pi* [Online] Tersedia di: <https://scienceprog.com/powerful-open-source-sqlite-manager-for-raspberry-pi/> [Diakses 2 Februari 2020].

Semanticscholar. 2018. *High level overview of fog based IoT*. [foto] (koleksi Semanticscholar)

Shemabook. 2017. *photoresistor*. [foto] (koleksi Shemabook)

Siregar, R., Akbar, S.R., Maulana, R., 2019 Implementasi Alexa Voice Command Untuk Pembacaan Informasi Sensor Pada Rumah Pintar. Jurnal Pengembangan Teknlogi Informasi dan Ilmu Komputer

Suakanto, S, 2015, Wireless Sensor Network Teori dan Praktik Berbasiskan open Source, Bandung, Informatika

SUNROM, 2008. *Light Dependent Resistor - LDR.* [pdf]. SUNROM. Tersedia di: <https://www.sunrom.com/get/443700> [Diakses 18 juli 2019]

Team, Editoral, 2019. *With Interoperability, You'll Feel Right At (Smart) Home.* [Online] Tersedia di: <https://datamakespossible.westerndigital.com/interoperability-for-smart-homes/> [Diakses 20 November2019].

Telcominded. 2017. Arsitektur WSN. [foto] (koleksi Telcominded)

Texas Instrument, 2017. *DHT11 Precision Centigrade Temperature Sensors.* [pdf]. Texas Instrument. Tersedia di: <http://www.ti.com/lit/ds/symlink/DHT11.pdf> [Diakses 14 juli 2019]

Tian, A, 2018. Teknologi SmartHome dan Manfaatnya. [Online] Tersedia di: https://www.immersa-lab.com/teknologi-smarthome-dan-manfaatnya.htm [Diakses 24 maret 2019].

Tutorialspoint, 2018. *SQLite SQL database engine tutorialpoint simplyeasylearning* [pdf]. Tutorialspoint. Tersedia di: <https://www.tutorialspoint.com/sqlite/sqlite_tutorial.pdf> [Diakses 14 juli 2019]

W, Xinlong., W, Yunliang., 2018. *Design of smart home environment monitoring system based on raspberry Pi*. In: IEEE, 2018 *Chinese Control And Decision Conference* (CCDC), Shenyang, China 9-11 June 2018. IEEE Xplore: IEEE

LAMPIRAN A KODE PROGRAM

A.1. Kode Program Node Sensor Cahaya

No.	Kode Program
1	`#include <RF24.h>`
2	`#include <RF24Network.h>`
3	`#include <SPI.h>`
4	`#include "printf.h"`
5	`#include <string.h>`
6	
7	`RF24 radio(10, 9);`
8	`RF24Network network(radio);`
9	
10	`const uint16_t nodeSHU = 021;`
11	`const uint16_t nodeSink = 011;`
12	`const uint16_t this_node = 01;`
13	
14	`int iterasi = 0;`
15	`char text[45];`
16	`bool ok;`
17	
18	`unsigned long waktu_kirim_detik = 0;`
19	`unsigned long waktu_terima_detik = 0;`
20	`unsigned long delay_pengiriman_detik;`
21	`unsigned long waktu_kirim_millisecond = 0;`
22	`unsigned long waktu_terima_millisecond = 0;`
23	`unsigned long delay_pengiriman_millisecond;`
24	`float Throughput_pengiriman;`
25	
26	`const unsigned long batasWaktu = 2000;`
27	`unsigned long waktuTerakhiKirim = 0;`
28	
29	`int state = 0;`
30	`int Selisih;`
31	`int Time1, Time2, Time3, Time4, SelisihWaktu, DelayPropagasi,`
32	`hasil, hasil3;`
33	`unsigned long Latency, milisec, second, menit, jam;`
34	
35	`typedef struct {`
36	`  int T2;`
37	`  int T3;`
38	`} A_t;`
39	
40	`A_t duino1;`
41	
42	`void setup() {`
43	`  Serial.begin(9600);`
44	`  printf_begin();`
45	`  SPI.begin();`

```
46    radio.begin();
47    network.begin(90, this_node);
48    radio.printDetails();
49  }
50  void loop() {
51    network.update();
52    waktu();
53
54      while ( network.available() ) {
55        RF24NetworkHeader header;
56        char incomingData[45];
57        network.read(header,                    &incomingData,
58  sizeof(incomingData));
59        waktu_terima_detik = millis() / 1000;
60        waktu_terima_millisecond = millis();
61
62        if (header.from_node == nodeSink) {
63          Serial.print("-");
64          Serial.print(incomingData);
65          delay_pengiriman_millisecond = waktu_terima_millisecond
66    - waktu_kirim_millisecond;
67          delay_pengiriman_detik      =      waktu_terima_detik      -
68  waktu_kirim_detik;
69          Throughput_pengiriman  =  float(sizeof(text))  /  float
70  (delay_pengiriman_millisecond);
71          //Serial.print("delay: ");
72          Serial.print("-");
73          Serial.print(delay_pengiriman_detik);
74          Serial.print("-");
75          Serial.print(delay_pengiriman_millisecond);
76          Serial.print("-");
77          //Serial.print("throughput: ");
78          Serial.print(Throughput_pengiriman);
79        }
80        if (header.from_node == nodeSHU) {
81          if (strcmp(incomingData, "syc") == 0) {
82            state = 1;
83             delay(10);
84          }
85        }
86    }
87      if (state == 0) {
88        Serial.println("tunggu....");
89        duino1.T2 = second;
90      }
91      if (state == 1) {
93        duino1.T3 = second;
94        RF24NetworkHeader waktuSingkron(nodeSHU);
95        network.write(waktuSingkron,          (uint8_t*)&duino1,
96  sizeof(duino1));
97
```

```
    state = 3;
  }
  if (state == 3) {
    if ( second == 4 || second == 8 || second == 12 || second
== 16 || second == 20 || second == 24 || second == 28 ||
        second == 32 || second == 36 || second == 40 ||
second == 44 || second == 48 || second == 52 || second == 56
|| second == 0 ) {
      unsigned long sekarang = millis();
      if (sekarang - waktuTerakhiKirim >= batasWaktu) {
        waktuTerakhiKirim = sekarang;
        int cahaya = analogRead(A0);
        itoa(cahaya, text, 10);
        char kirim [] = " LDR";
        strcat(text, kirim);
        delay(250);
        RF24NetworkHeader header(nodeSink);
        ok = network.write(header, &text, sizeof(text));
        waktu_kirim_detik = millis() / 1000;
        waktu_kirim_millisecond = millis();

        Serial.print("-");
        Serial.print(ok);
        iterasi += 1;
        //        Serial.print("second: ");
        //        Serial.print(second);
        Serial.print("-");
        Serial.print(iterasi);
        Serial.println();
      }
    }
  }
}
void waktu() {
  milisec = millis() % 1000;
  second = millis() / 1000;
  menit = millis() / 60000;
  jam = millis() / 1440000;

  if (second > 59) { //reset second
    second = second % 60;
  }

  if (menit > 59) { //reset menit
    menit = menit % 60;
  }

  if (jam > 23) { //reset jam
    jam = jam % 24;
  }
}
```

A.2. Kode Program Node Sensor Suhu

No.	Kode Program
1	`#include <RF24.h>`
2	`#include <RF24Network.h>`
3	`#include <SPI.h>`
4	`#include "printf.h"`
5	`#include <string.h>`
6	`RF24 radio(10, 9);`
7	`RF24Network network(radio);`
8	`const uint16_t this_node = 021;`
9	`const uint16_t nodeSink = 011;`
10	`const uint16_t nodeCHY = 01;`
11	`int iterasi = 0;`
12	`char text[45];`
13	`bool ok;`
14	`unsigned long waktu_kirim_detik = 0;`
15	`unsigned long waktu_terima_detik = 0;`
16	`unsigned long delay_pengiriman_detik;`
17	`unsigned long waktu_kirim_millisecond = 0;`
18	`unsigned long waktu_terima_millisecond = 0;`
19	`unsigned long delay_pengiriman_millisecond;`
20	`float Throughput_pengiriman;`
21	`const unsigned long batasWaktu = 2000;`
22	`unsigned long waktuTerakhiKirim = 0;`
23	`int state = 0;`
24	`int Selisih;`
25	`int waktu1, waktu2, waktu3, waktu4, SelisihWaktu,`
26	`DelayPropagasi, hasil, hasil3;`
27	`unsigned long milisec, second, mt, hour;`
28	`typedef struct {`
29	`  int T2;`
30	`  int T3;`
31	`} A_t;`
32	
33	`A_t duino1;`
34	
35	`void setup() {`
36	`  Serial.begin(9600);`
37	`  printf_begin();`
38	`  SPI.begin();`
39	`  radio.begin();`
40	`  network.begin(90, this_node);`
41	`  radio.printDetails();`
42	`}`
43	`void loop() {`
44	`  network.update();`
45	`  waktu();`
46	`  while ( network.available() ) {`
47	`    RF24NetworkHeader waktuSingkron;`
48	`    network.read(waktuSingkron, (uint8_t *)&duino1,`

```
sizeof(duino1));
      waktu_terima_detik = millis() / 1000;
      waktu_terima_millisecond = millis();

      if (waktuSingkron.from_node == nodeSink) {
        Serial.print("-");
        Serial.print(duino1.T2);
delay_pengiriman_millisecond = waktu_terima_millisecond -
waktu_kirim_millisecond;
        delay_pengiriman_detik = waktu_terima_detik -
waktu_kirim_detik;
        Throughput_pengiriman = float(sizeof(text)) / float
(delay_pengiriman_millisecond);
        Serial.print("-");
        Serial.print(delay_pengiriman_detik);
        Serial.print("-");
        Serial.print(delay_pengiriman_millisecond);
        Serial.print("-");
        Serial.print(Throughput_pengiriman);
      }
      if (waktuSingkron.from_node == nodeCHY) {
        waktu4 = second;
        state = 3;
        waktu2 = duino1.T2;
        waktu3 = duino1.T3;

        SynchCalculation(waktu1, waktu2, waktu3, waktu4);
      }
    }
    if (state == 0) {
      char singkron[45] = "syc";
      RF24NetworkHeader header(nodeCHY);
      ok = network.write(header, &singkron, sizeof(singkron));
      Serial.println(ok);
      waktu1 = second;
      state = 1;
    }
     if (state == 3) {
      if ( (second + SelisihWaktu) == 2 || (second +
SelisihWaktu) == 6 || (second + SelisihWaktu) == 10 ||
(second + SelisihWaktu) == 14 ||
          (second + SelisihWaktu) == 18 || (second +
SelisihWaktu) == 22 || (second + SelisihWaktu) == 26 ||
(second + SelisihWaktu) == 30 ||
          (second + SelisihWaktu) == 34 || (second +
SelisihWaktu) == 38 || (second + SelisihWaktu) == 42 ||
(second + SelisihWaktu) == 46 ||
          (second + SelisihWaktu) == 50 || (second +
SelisihWaktu) == 54 || (second + SelisihWaktu) == 58 ||
(second + SelisihWaktu) == 62 ||
          (second + SelisihWaktu) == 66 || (second +
```

```
SelisihWaktu) == 70 || (second + SelisihWaktu) == 74 ||
(second + SelisihWaktu) == 78 ||
    (second + SelisihWaktu) == 82 ) {
      unsigned long sekarang = millis();
      if (sekarang - waktuTerakhiKirim >= batasWaktu) {
        waktuTerakhiKirim = sekarang;
        int suhu = analogRead(A0) * (500.0 / 1023.0);
        itoa(suhu, text, 10);
        char kirim [] = " DHT";
        strcat(text, kirim);
        delay(250);
        RF24NetworkHeader header(nodeSink);
        ok = network.write(header, &text, sizeof(text));
        waktu_kirim_detik = millis() / 1000;
        waktu_kirim_millisecond = millis();

        Serial.print("-");
        Serial.print(ok);
        iterasi += 1;
        //Serial.print("second: ");
        //Serial.print(second + SelisihWaktu);
        Serial.print("-");
        Serial.print(iterasi);
        Serial.println();
      }
    }
  }
}
void SynchCalculation(int a, int b, int c, int d) {
  hasil = (b - a) - (d - c);
  SelisihWaktu = hasil / 2;
  Selisih = SelisihWaktu;
  hasil3 = (b - a) + (d - c);
  DelayPropagasi = hasil3 / 2;
  Serial.print("Selisih Waktu : ");
  Serial.println(SelisihWaktu);
  delay(1000);
}
void waktu() {
  milisec = millis() % 1000;
  second = millis() / 1000;
  mt = millis() / 60000;
  hour = millis() / 1440000;

  if (second > 59) {
    second = second % 60;
  }

  if (mt > 59) {
    mt = mt % 60;
  }
```

```
154
155   if (hour > 23) {
156     hour = hour % 24;
157   }
158 }
```

A.3. Kode Program Node Sink

```
1   #include <RF24.h>
2   #include <RF24Network.h>
3   #include <SPI.h>
4   #include "printf.h"
5   #include <string.h>
6
7   RF24 radio(10, 9); // nRF24L01 (CE,CSN)
8   RF24Network network(radio);
9
10  const uint16_t nodeCHY = 01;
11  const uint16_t nodeSHU = 021;
12  const uint16_t this_node = 011;
13  const uint16_t nodeFog = 00;
14
15  int iterasiLDR = 0;
16  int iterasiDHT = 0;
17  unsigned long waktu_kirim_detik = 0;
18  unsigned long waktu_terima_detik = 0;
19  unsigned long delay_pengiriman_detik;
20  unsigned long waktu_kirim_millisecond = 0;
21  unsigned long waktu_terima_millisecond = 0;
22  unsigned long delay_pengiriman_millisecond;
23
24  bool ok;
25  char incomingData[45];
26  float Throughput_pengiriman;
27
28  typedef struct {
29    int T2;
30    int T3;
31  } A_t;
32
33  A_t duino1;
34
35  void setup() {
36    Serial.begin(9600);
37    printf_begin();
38    SPI.begin();
39    radio.begin();
40    network.begin(90, this_node);
41    radio.printDetails();
42  }
```

```
43
44  void loop() {
45    network.update();
46    while (network.available()) {
47      RF24NetworkHeader header;
48      network.read(header, &incomingData,
49  sizeof(incomingData));
50      waktu_terima_detik = millis() / 1000;
51      waktu_terima_millisecond = millis();
52
53      if (header.from_node == nodeFog) {
54        Serial.print("-");
55        Serial.print(incomingData);
56
57        delay_pengiriman_millisecond = waktu_terima_millisecond
58  - waktu_kirim_millisecond;
59        delay_pengiriman_detik = waktu_terima_detik -
60  waktu_kirim_detik;
61        Throughput_pengiriman = float(sizeof(incomingData)) /
62  float (delay_pengiriman_millisecond);
63
64        //Serial.print("delay: ");
65        Serial.print("-");
66        Serial.print(delay_pengiriman_detik);
67        Serial.print("-");
68        Serial.print(delay_pengiriman_millisecond);
69        Serial.print("-");
70        Serial.print(Throughput_pengiriman);
71
72        if (strcmp(incomingData, "CHY") == 0) {
73          //konfirmasi
74          char balasDariFog[45] = "Sukses";
75          delay(300);
76          RF24NetworkHeader header(nodeCHY);
77          ok = network.write(header, &balasDariFog,
78  sizeof(balasDariFog));
79
80        } else {
81          duino1.T2 = 2;
82          duino1.T3 = 1;
83          delay(300);
84          RF24NetworkHeader header(nodeSHU);
85          ok = network.write(header, (uint8_t*)&duino1,
86  sizeof(duino1));
87        }
88      }
89      if (header.from_node == nodeCHY) {
90        Serial.print("-");
91        Serial.print(incomingData);
93        RF24NetworkHeader headerFog(nodeFog);
94        ok = network.write(headerFog, &incomingData,
```

```
sizeof(incomingData));
        waktu_kirim_detik = millis() / 1000;
        waktu_kirim_millisecond = millis();
        Serial.print("-");
        Serial.print(ok);
        char balas[45] = "ok";
        delay(300);
        RF24NetworkHeader header(nodeCHY);
        ok = network.write(header, &balas, sizeof(balas));
        Serial.print("-LDR-");
        iterasiLDR += 1;
        Serial.print(iterasiLDR);
        Serial.println();
    }

    if (header.from_node == nodeSHU) {
        Serial.print("-");
        Serial.print(incomingData);

        RF24NetworkHeader headerFog(nodeFog);
        ok     =     network.write(headerFog,      &incomingData,
sizeof(incomingData));
        waktu_kirim_detik = millis() / 1000;
        waktu_kirim_millisecond = millis();
        Serial.print("-");
        Serial.print(ok);
        duino1.T2 = 1;
        duino1.T3 = 2;
        delay(300);
        RF24NetworkHeader header(nodeSHU);
        ok     =     network.write(header,      (uint8_t*)&duino1,
sizeof(duino1));
        iterasiDHT += 1;
        Serial.print(iterasiDHT);
        Serial.println();
    }
  }
}
```

A.4. Kode Program Node Fog Untuk Web Server

No.	Kode Program
1	`# suhu GPIO 20`
2	`# cahaya GPIO 21`
3	`from flask import Flask, render_template, redirect, request,`
4	`jsonify`
5	`from datetime import datetime`
6	`import sqlite3`
7	`# import RPi.GPIO as GPIO`
8	`import time`
9	

```
# inisialisasi flask
app = Flask(__name__, template_folder='template')

#kumpulan variabel
kondisi_1 = "mati"
kondisi_2 = "mati"
kondisi_1_seb = ""
kondisi_2_seb = ""

# GPIO.setmode(GPIO.BCM)
# GPIO.setup(16, GPIO.OUT)
# GPIO.setup(20, GPIO.OUT)

@app.route("/", methods = ['GET','POST'])
def index():
    now = datetime.now()
    global kondisi_1
    global kondisi_2
    global kondisi_1_seb
    global kondisi_2_seb
    try:
        with sqlite3.connect('mainDB.db') as conn:
            kurs = conn.cursor()
            kurs.execute("select * from aktuator order by id
desc LIMIT 8")
    except:
        print("gagal")
    finally:
        rows = kurs.fetchall()
        kondisi_1 = rows[0][1]
        kondisi_2 = rows[0][2]
        conn.close()

    if request.method == 'POST':
        if request.form['submit'] == 'hidupKipas':
            kondisi_1_seb = kondisi_1
            kondisi_1 = "hidup"
            # GPIO.output(20, True)
            # time.sleep(0.5)
            if(kondisi_1_seb != kondisi_1):
                try:
                    with sqlite3.connect('mainDB.db') as
conn:
                        kurs = conn.cursor()
                        kurs.execute(
                            "insert into aktuator
(statusSuhu, statusCahaya, tgl, jam) values(?,?,?, ?)",
                            (kondisi_1, kondisi_2,
now.strftime("%d/%m/%Y"), now.strftime("%H:%M:%S")))
                        conn.commit()
```

```
                except:
                    print("gagal")
                finally:
                    conn.close()

        if request.form['submit'] == 'matiKipas':
            kondisi_1_seb = kondisi_1
            kondisi_1 = "mati"
            # GPIO.output(20, False)
            # time.sleep(0.5)
            if (kondisi_1_seb != kondisi_1):
                try:
                    with sqlite3.connect('mainDB.db') as
conn:
                        kurs = conn.cursor()
                        kurs.execute(
                            "insert into aktuator
(statusSuhu, statusCahaya, tgl, jam) values(?,?,?, ?)",
                            (kondisi_1, kondisi_2,
now.strftime("%d/%m/%Y"), now.strftime("%H:%M:%S")))
                        conn.commit()
                except:
                    print("gagal")
                finally:
                    conn.close()

        if request.form['submit'] == 'hidupLampu':
            kondisi_2_seb = kondisi_2
            kondisi_2 = "hidup"
            # GPIO.output(16, True)
            # time.sleep(0.5)
            if (kondisi_2_seb != kondisi_2):
                try:
                    with sqlite3.connect('mainDB.db') as
conn:
                        kurs = conn.cursor()
                        kurs.execute(
                            "insert into aktuator
(statusSuhu, statusCahaya, tgl, jam) values(?,?,?, ?)",
                            (kondisi_1, kondisi_2,
now.strftime("%d/%m/%Y"), now.strftime("%H:%M:%S")))
                        conn.commit()
                except:
                    print("gagal")
                finally:
                    conn.close()

        if request.form['submit'] == 'matiLampu':
            kondisi_2_seb = kondisi_2
            kondisi_2 = "mati"
            # GPIO.output(16, False)
```

```
114         # time.sleep(0.5)
115         if (kondisi_2_seb != kondisi_2):
116             try:
117                 with sqlite3.connect('mainDB.db') as
118 conn:
119                     kurs = conn.cursor()
120                     kurs.execute(
121                         "insert into aktuator
122 (statusSuhu, statusCahaya, tgl, jam) values(?,?,?, ?)",
123                         (kondisi_1, kondisi_2,
124 now.strftime("%d/%m/%Y"), now.strftime("%H:%M:%S")))
125                     conn.commit()
126             except:
127                 print("gagal")
128             finally:
129                 conn.close()
130
131     return render_template('index.html', kondisi= [kondisi_1,
132 kondisi_2], rows = rows)
133
134 if __name__ == '__main__':
135     app.run(host='0.0.0.0', port=8000, debug=True)
```

A.5. Kode Program Node Fog Untuk Menerima Data Dari Node Sink

No.	Kode Program
1	`#include <wiringPi.h>`
2	`#include <chrono>`
3	`#include <RF24/RF24.h>`
4	`#include <RF24Network/RF24Network.h>`
5	`#include <iostream>`
6	`#include <string.h>`
7	`#include <sstream>`
8	`#include <ctime>`
9	`#include <stdio.h>`
10	`#include <time.h>`
11	`#include <sqlite3.h>`
12	
13	`#define indikatorSuhu 19`
14	`#define indikatorLDR 26`
15	
16	`using namespace std;`
17	
18	`RF24 radio(RPI_V2_GPIO_P1_15, BCM2835_SPI_CS0,`
19	`BCM2835_SPI_SPEED_8MHZ);`
20	
21	`RF24Network network(radio);`
22	
23	`const uint16_t this_node = 00;`
24	`const uint16_t nodeSink = 01;`
25	

```
26  static int callback(void *NotUsed, int argc, char **argv,
27  char **azColName) {
28     int i;
29     for(i = 0; i<argc; i++) {
30        printf("%s = %s\n", azColName[i], argv[i] ? argv[i] :
31  "NULL");
32     }
33     printf("\n");
34     return 0;
35  }
36
37  int main(int argc, char** argv) {
38        radio.begin();
39        wiringPiSetupGpio();
40        pinMode(indikatorSuhu, OUTPUT);
41        pinMode(indikatorLDR, OUTPUT);
42
43        delay(10);
44        network.begin(90, this_node);
45
46        radio.printDetails();
47
48        while(1) {
49        // awal devinisi dataBase
50        sqlite3 *db;
51        char *zErrMsg = 0;
52        int rc;
53        // akhir devinisi dataBase
54
55        // awal devinisi waktu
56        time_t now = time(0);
57        tm *ltm = localtime(&now);
58        string tahunS, bulanS, hariS, jamS, menitS, detikS;
59
60        stringstream cvt1;
61        int tahun = 1900 + ltm->tm_year;
62        cvt1 << tahun;
63        tahunS = cvt1.str();
64
65        stringstream cvt2;
66        int bulan = 1 + ltm->tm_mon;
67        cvt2 << bulan;
68        bulanS = cvt2.str();
69
70        stringstream cvt3;
71        int hari = ltm->tm_mday;
72        cvt3 << hari;
73        hariS = cvt3.str();
74
75        stringstream cvt4;
76        int jam = 1 + ltm->tm_hour;
```

```
77      cvt4 << jam;
78      jamS = cvt4.str();
79
80      stringstream cvt5;
81      int menit = ltm->tm_min;
82      cvt5 << menit;
83      menitS = cvt5.str();
84
85      stringstream cvt6;
86      int detik = ltm->tm_sec;
87      cvt6 << detik;
88      detikS = cvt6.str();
89
90      hariS += ":";
91      hariS += bulanS;
93      hariS += ":";
94      hariS += tahunS;
95
96      jamS += ":";
97      jamS += menitS;
98      jamS += ":";
99      jamS += detikS;
100       // akhir devinisi waktu;
101
102             network.update();
104             while ( network.available() ) {
105                 // awal waktu untuk pengujian
106                 auto                epoch                =
107 chrono::high_resolution_clock::from_time_t(0);
108                 auto      now                =
109 chrono::high_resolution_clock::now();
110                 auto                mseconds             =
111 chrono::duration_cast<chrono::milliseconds>(now             -
112 epoch).count();
113                 int detikPengujian = ltm->tm_sec;
114                 cout << "Ambil: " << detikPengujian << ":"
115 << mseconds << endl;
116                 // akhir waktu untuk pengujian
117
118                 //awal menerima data
119             char dataTerima[25];
120                 RF24NetworkHeader header;
121                 network.read(header,              &dataTerima,
122 sizeof(dataTerima));
123
124                 cout << "data nodeSink: " << dataTerima <<
125 endl;
126                 //akhir menerima data
127
128                 //awal bermain string
129                 string dataTerimaString(dataTerima);
```

```
130                 string arrDataTerimaString[2];
131
132                 int i = 0;
133                 stringstream ssin(dataTerimaString);
134                 while(ssin.good() && i < 2) {
135                     ssin >> arrDataTerimaString[i];
136                     ++i;
137                 }
138
139                 stringstream
140 strToInt(arrDataTerimaString[0]);
141                 int nilaiSensor = 0;
143                 strToInt >> nilaiSensor;
144
145                 cout  <<  "integer:  "  <<  nilaiSensor  <<
146 "lainya:  "  <<  arrDataTerimaString[0]  <<  "  dan  "  <<
147 arrDataTerimaString[1] << endl;
148
149                 //awal mengirim balasan ke node sink
150                 if(arrDataTerimaString[1].compare("LDR") ==
151 0) {
152                     char balik[15] = "DHT";
153                     RF24NetworkHeader
154 headerBalik(nodeSink);
155                     bool ok = network.write(headerBalik,
156 &balik, sizeof(balik));
157                     if (ok) {
158                         cout << "oke" << endl;
159                     } else {
160                         cout << "fail" << endl;
161                     }
162                 } else {
163                     char balik[15] = "LDR";
162                     RF24NetworkHeader
165 headerBalik(nodeSink);
166                     bool ok = network.write(headerBalik,
167 &balik, sizeof(balik));
168                     if (ok) {
169                         cout << "oke" << endl;
170                     } else {
171                         cout << "fail" << endl;
172                     }
173                 }
174                 //akhir mengirim balasan ke node sink
175
176                 // awal waktu untuk pengujian setelah fake
177                 epoch                                   =
178 chrono::high_resolution_clock::from_time_t(0);
179                 now                                     =
180 chrono::high_resolution_clock::now();
181                 mseconds                                =
```

```
chrono::duration_cast<chrono::milliseconds>(now - epoch).count();
                detikPengujian = ltm->tm_sec;
                cout << "Habis Analisis: " << detikPengujian << ":" << mseconds << endl;
                // akhir waktu untuk pengujian setelah fake
                //string bandingData(arrDataTerimaString[1]);
                //awal fake analisis data
                if(arrDataTerimaString[1].compare("LDR") == 0) {
                    if(nilaiSensor < 600) {
                        cout << "gelap" << endl;
                        digitalWrite(indikatorLDR, HIGH);
                        delay(500);
                    } else {
                        cout << "terang" << endl;
                        digitalWrite(indikatorLDR, LOW);
                        delay(500);
                    }
                } else {
                    if(nilaiSensor > 30 ) {
                        cout << "panas" << endl;
                        digitalWrite(indikatorSuhu, HIGH);
                        delay(500);
                    } else {
                        cout << "dingin" << endl;
                        digitalWrite(indikatorSuhu, LOW);
                        delay(500);
                    }
                }
                //akhir fake analisis data
                //akhir bermain string
                /* Open database */
                rc = sqlite3_open("mainDB.db", &db);
                if(rc) {
                    fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
                    return(0);
                } else {
                    fprintf(stderr, "Opened database successfully\n");
                }

```

```
233                     string sql("insert into sensor (nilai, tgl,
234 jam, sensor) values(");
235                     sql += arrDataTerimaString[0];
236                     sql += ",'";
237                     sql += hariS;
238                     sql += "','";
239                     sql += jamS;
240                     sql += "','";
241                     sql += arrDataTerimaString[1];
242                     sql += "');";
243
244                     cout << sql << endl;
245
246                     rc    =    sqlite3_exec(db,    sql.c_str(),
247 callback, 0, &zErrMsg);
248
249                     if( rc != SQLITE_OK ){
250                       fprintf(stderr,  "SQL  error:  %s\n",
251 zErrMsg);
252                       sqlite3_free(zErrMsg);
253                     } else {
254                       fprintf(stdout,    "Records    created
255 successfully\n");
256                     }
257                     sqlite3_close(db);
258
259                     //  awal  waktu  untuk  pengujian  setelah
260 SQLite
261                     epoch                                    =
262 chrono::high_resolution_clock::from_time_t(0);
263                     now                                      =
264 chrono::high_resolution_clock::now();
265                     mseconds                                 =
266 chrono::duration_cast<chrono::milliseconds>(now              -
267 epoch).count();
268                     detikPengujian = ltm->tm_sec;
269                     cout << "Habis SQLite: " << detikPengujian
270 << ":" << mseconds << endl;
271                     //  akhir  waktu  untuk  pengujian  setelah
272 SQLite
273                   }
274                 delay(500);
275         }
276         return 0;
277 }
```

A.6. Kode Program Node Fog Untuk Mengirimkan Ke Cloud

No.	Kode Program
1	# idBefore == batas bawah
2	# idNow == batas atas

```
import pymongo
import sqlite3
import time
from datetime import datetime

with sqlite3.connect('mainDB.db') as conn:
        dhtAmbil = conn.cursor()
        dhtAmbil.execute("select id from sensor where sensor ==
'DHT' order by id desc LIMIT 1")
        rowsDHT = dhtAmbil.fetchone()
        idBeforeDHT = rowsDHT[0]
        idNowDHT = idBeforeDHT

with sqlite3.connect('mainDB.db') as conn:
        ldrAmbil = conn.cursor()
        ldrAmbil.execute("select id from sensor where sensor ==
'LDR' order by id desc LIMIT 1")
        rowsLDR = ldrAmbil.fetchone()
        idBeforeLDR = rowsLDR[0]
        idNowLDR = idBeforeLDR

while True:
        try:
                myClient = pymongo.MongoClient(
                "mongodb+srv://user:user@cluster0-
zwzxj.mongodb.net/test?retryWrites=true&w=majority"
                )

                conn = sqlite3.connect('mainDB.db')
                conn.row_factory = sqlite3.Row
                dhtAmbil = conn.cursor()
                dhtAmbil.execute("select * from sensor where
sensor == 'DHT' and id > %s order by id desc" % idBeforeDHT)
                rowsDHT = dhtAmbil.fetchall()

                ldrAmbil = conn.cursor()
                ldrAmbil.execute("select * from sensor where
sensor == 'LDR' and id > %s order by id desc" % idBeforeLDR)
                rowsLDR = ldrAmbil.fetchall()
        except:
                print("gagal konek dengan monggoDB")
        finally:
                try:
                        rowsDataDHT = [ dict(rec) for rec in
rowsDHT ]
                        rowsDataLDR = [ dict(rec) for rec in
rowsLDR ]

                        if(len(rowsDataDHT) > 0):
                                idNowDHT = rowsDataDHT[0]['id']
                        elif(len(rowsDataLDR) > 0):
```

```
                    idNowLDR = rowsDataLDR[0]['id']
                else:
                    print("tidak berubah batas atasnya")

            if(idNowDHT == idBeforeDHT):
                print("DHT do noting ? - ?", (idBeforeDHT, idNowDHT))
            else:
                myDatabase = myClient['dbRumah']
                myKoleksi = myDatabase['dbRuangan']
                ujiWaktu = {"timeStamp": time.clock() , "dateTime": datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')}
                rowsDataDHT.append(ujiWaktu)
                myKoleksi.insert(rowsDataDHT)
                print(datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f'))
                idBeforeDHT = idNowDHT

            if(idNowLDR == idBeforeLDR):
                print("LDR do noting ? - ?", (idBeforeLDR, idNowLDR))
            else:
                myDatabase = myClient['dbRumah']
                myKoleksi = myDatabase['dbRuangan']
                ujiWaktu = {"timeStamp": time.clock() , "dateTime": datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')}
                rowsDataLDR.append(ujiWaktu)
                myKoleksi.insert(rowsDataLDR)
                print(datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f'))
                idBeforeLDR = idNowLDR
        except:
            print("gagal memasukan data ke monggoDB")
        finally:
            conn.close()
            time.sleep(1)
```

LAMPIRAN B TABEL

B.1. Tabel Hasil Pengujian 4 Meter

Rata rata	Suhu tanpa penghalang	Suhu ada penghalagn	Cahaya tanpa penghalang	Cahaya ada penghalang
Delay(s->ns)	**332.15 ms**	**325.21 ms**	**330.09 ms**	**329.78 ms**
Delay(ns->nf)	**340.94 ms**	**348.48 ms**	**343.42 ms**	**343.86 ms**
RTT(round trip-time)	**672.7 ms**	**673.07 ms**	**673.15 ms**	**674.2 ms**
Throughput(s->ns)	**0.1337 ms**	**0.1357 ms**	**0.1355 ms**	**0.1344 ms**
Throughput(ns->nf)	**0.136 ms**	**0.1336 ms**	**0.1352 ms**	**0.1355 ms**
Delay(s->ns) + Delay(ns->nf)	**673.09 ms**	**673.69 ms**	**673.51 ms**	**673.64 ms**
Packet loss (sink)	**0%**	**0%**	**0%**	**0%**
Packet loss (sensor)	**0%**	**0%**	**0%**	**0%**

B.2. Tabel Hasil Pengujian 8 Meter

Rata rata	Suhu tanpa penghalang	Suhu ada penghalagn	Cahaya tanpa penghalang	Cahaya ada penghalang
Delay(s->ns)	**334.31 ms**	**330.08 ms**	**332.81 ms**	**327.9 ms**
Delay(ns->nf)	**348.06 ms**	**344.06 ms**	**339.2 ms**	**342.64 ms**
RTT(round trip-time)	**678.8 ms**	**674.18 ms**	**673.46 ms**	**671.39 ms**
Througput(s->ns)	**0.1331 ms**	**0.1341 ms**	**0.134 ms**	**0.1355 ms**
Througput(ns->nf)	**0.134 ms**	**0.1351 ms**	**0.1366 ms**	**0.1356 ms**
Delay(s->ns) + delay(ns->nf)	**682.37 ms**	**674.14 ms**	**672.01 ms**	**670.54 ms**
Packet loss (sink)	**0%**	**0%**	**0%**	**0%**
Packet loss (sensor)	**0%**	**0%**	**0%**	**0%**

B.3. Tabel Hasil Pengujian 12 Meter

Rata rata	Suhu tanpa penghalang	Suhu ada penghalagn	Cahaya tanpa penghalang	Cahaya ada penghalang
Delay(s->ns)	**327.19 ms**	**324.73 ms**	**330.48 ms**	**324.59 ms**
Delay(ns->nf)	**345.62 ms**	**340.1616162 ms**	**346.95 ms**	**350.77 ms**

RTT(round trip-time)	**670.84 ms**	**664.49 ms**	**675.03 ms**	**674.6 ms**
Througput(s->ns)	**0.1355 ms**	**0.1357 ms**	**0.1351 ms**	**0.1377 ms**
Througput(ns->nf)	**0.1345 ms**	**0.1367 ms**	**0.1343 ms**	**0.1334 ms**
Delay(s->ns) + delay(ns->nf)	**672.81 ms**	**661.49 ms**	**677.43 ms**	**675.36 ms**
Packet loss (sink)	**0%**	**0%**	**0%**	**0%**
Packet loss (sensor)	**0%**	**0%**	**0%**	**0%**

B.4. Tabel Hasil Pengujian 16 Meter

Rata rata	Suhu tanpa penghalang	Suhu ada penghalagn	Cahaya tanpa penghalang	Cahaya ada penghalang
Delay(s->ns)	**337.16 ms**	**327.83 ms**	**334.62 ms**	**325.95 ms**
Delay(ns->nf)	**339.52 ms**	**347.54 ms**	**349.96 ms**	**350.36 ms**
RTT(round trip-time)	**677.39 ms**	**674.37 ms**	**681.98 ms**	**675.78 ms**
Througput(s->ns)	**0.1322 ms**	**0.1347 ms**	**0.1353 ms**	**0.1373 ms**
Througput(ns->nf)	**0.1364 ms**	**0.1344 ms**	**0.1335 ms**	**0.1336 ms**
Delay(s->ns) + delay(ns->nf)	**676.68 ms**	**675.37 ms**	**684.58 ms**	**676.31 ms**
Packet loss (sink)	**0%**	**0%**	**0%**	**0%**
Packet loss (sensor)	**0%**	**0%**	**0%**	**0%**

B.5. Tabel Hasil Pengujian 20 Meter

Rata rata	Suhu tanpa penghalang	Suhu ada penghalagn	Cahaya tanpa penghalang	Cahaya ada penghalang
Delay(s->ns)	**347.14 ms**	**360.8679245 ms**	**344.57 ms**	**343.8292683 ms**
Delay(ns->nf)	**341.46 ms**	**343.8 ms**	**343.0103093 ms**	**367.55 ms**
RTT(round trip-time)	**689.11 ms**	**715.7924528 ms**	**685.15 ms**	**697.7073171 ms**
Througput(s->ns)	**0.1299 ms**	**0.127169811 ms**	**0.1307 ms**	**0.131219512 ms**
Througput(ns->nf)	**0.1361 ms**	**0.134 ms**	**0.135463918 ms**	**0.126 ms**
Delay(s->ns) + delay(ns->nf)	**688.6 ms**	**698.0980392 ms**	**677.29 ms**	**710.325 ms**

Packet loss (sink)	**0%**	**47%**	**0%**	**60%**
Packet loss (sensor)	**0%**	**50%**	**0%**	**59%**

B.6. Tabel Hasil Pengujian 24 Meter

Rata rata	Suhu tanpa penghalang	Suhu ada penghalagn	Cahaya tanpa penghalang	Cahaya ada penghalang
Delay(s->ns)	**353.4819277 ms**	**371.75 ms**	**373.0571429 ms**	**342.1052632 ms**
Delay(ns->nf)	**347.125 ms**	**363.0909091 ms**	**347.9354839 ms**	**352.6315789 ms**
RTT(round trip-time)	**696.0120482 ms**	**719.1944444 ms**	**718.9857143 ms**	**701 ms**
Througput(s->ns)	**0.127710843 ms**	**0.123888889 ms**	**0.121857143 ms**	**0.131052632 ms**
Througput(ns->nf)	**0.13375 ms**	**0.13 ms**	**0.132419355 ms**	**0.131052632 ms**
Delay(s->ns) + delay(ns->nf)	**703.8611111 ms**	**725.7272727 ms**	**724.6290323 ms**	**694.7368421 ms**
Packet loss (sink)	**28%**	**68%**	**38%**	**81%**
Packet loss (sensor)	**17%**	**64%**	**30%**	**81%**

B.7. Tabel Hasil Pengujian 28 Meter

Rata rata	Suhu tanpa penghalang	Suhu ada penghalagn	Cahaya tanpa penghalang	Cahaya ada penghalang
Delay(s->ns)	**352.4126984 ms**	**395.5 ms**	**354.9310345 ms**	**351 ms**
Delay(ns->nf)	**352.9642857 ms**	**427.25 ms**	**344.7017544 ms**	**346.3333333 ms**
RTT(round trip-time)	**710.1904762 ms**	**705 ms**	**704.4827586 ms**	**675.3333333 ms**
Througput(s->ns)	**0.13015873 ms**	**0.115 ms**	**0.127931034 ms**	**0.126666667 ms**
Througput(ns->nf)	**0.131071429 ms**	**0.105 ms**	**0.132631579 ms**	**0.13 ms**
Delay(s->ns) + delay(ns->nf)	**707.6607143 ms**	**822.75 ms**	**699.8596491 ms**	**697.3333333 ms**
Packet loss (sink)	**44%**	**96%**	**43%**	**97%**
Packet loss (sensor)	**37%**	**96%**	**42%**	**97%**

B.8. Tabel Hasil Pengujian 32 Meter

Rata rata	Suhu tanpa penghalang	Cahaya tanpa penghalang
Delay(s->ns)	**371.1166667 ms**	**369.5333333 ms**
Delay(ns->nf)	**353 ms**	**355.9047619 ms**
RTT(round trip-time)	**727.75 ms**	**720.0666667 ms**
Througput(s->ns)	**0.123833333 ms**	**0.123333333 ms**
Througput(ns->nf)	**0.131730769 ms**	**0.130238095 ms**
Delay(s->ns) + delay(ns->nf)	**720.1730769 ms**	**727.0238095 ms**
Packet loss (sink)	**48%**	**58%**
Packet loss (sensor)	**40%**	**55%**

B.9. Tabel Hasil Pengujian 36 Meter

Rata rata	Suhu tanpa penghalang	Cahaya tanpa penghalang
Delay(s->ns)	**357.7619048 ms**	**359.5 ms**
Delay(ns->nf)	**339.1111111 ms**	**369.9230769 ms**
RTT(round trip-time)	**697.5238095 ms**	**706.4375 ms**
Througput(s->ns)	**0.128095238 ms**	**0.12625 ms**
Througput(ns->nf)	**0.135 ms**	**0.123076923 ms**
Delay(s->ns) + delay(ns->nf)	**694.7777778 ms**	**733.6923077 ms**
Packet loss (sink)	**82%**	**87%**
Packet loss (sensor)	**79%**	**84%**

B.10. Tabel Hasil Pengujian 40 Meter

Rata rata	Suhu tanpa penghalang	Cahaya tanpa penghalang
Delay(s->ns)	**377.6842105 ms**	**367.0666667 ms**
Delay(ns->nf)	**349.0588235 ms**	**345.2857143 ms**
RTT(round trip-time)	**720.6842105 ms**	**699.2 ms**

Througput(s->ns)	**0.12 ms**	**0.123333333 ms**
Througput(ns->nf)	**0.132352941 ms**	**0.133571429 ms**
Delay(s->ns) + delay(ns->nf)	**732.1764706 ms**	**713 ms**
Packet loss (sink)	**83%s**	**86%**
Packet loss (sensor)	**81%**	**85%**

B.11. Tabel Hasil Pengujian 44 Meter

Rata rata	Suhu tanpa penghalang	Cahaya tanpa penghalang
Delay(s->ns)	**400.5625 ms**	**370 ms**
Delay(ns->nf)	**355.5714286 ms**	**408.875 ms**
Rtt(round trip-time)	**762.875 ms**	**748 ms**
Througput(s->ns)	**0.1175 ms**	**0.122222222 ms**
Througput(ns->nf)	**0.130714286 ms**	**0.11375 ms**
Delay(s->ns) + delay(ns->nf)	**758.6428571 ms**	**781.375 ms**
Packet loss (sink)	**86%**	**92%**
Packet loss (sensor)	**84%**	**91%**

Keterangan:

s = node sensor, ns = node sink, nf = node fog, ms = millisecond