IMPLEMENTASI PERANGKAT *GATEWAY* UNTUK PENGIRIMAN DATA SENSOR DARI LAPANGAN KE PUSAT DATA PADA JARINGAN *WIRELESS SENSOR NETWORK* BERBASIS PERANGKAT NRF24L01

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Tsany Afif
145150200111174

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

IMPLEMENTASI PERANGKAT *GATEWAY* UNTUK PENGIRIMAN DATA SENSOR DARI LAPANGAN KE PUSAT DATA PADA JARINGAN *WIRELESS SENSOR NETWORK* BERBASIS PERANGKAT NRF24L01

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Tsany Afif
NIM: 145150200111174

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Januari 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Adhitya Bhawiyuga, S.Kom, M.Sc	Reza Andria Siregar, S.T., M.Kom.
NIP. 19890720 201803 1 002	NIP. 19790621 200604 1 003

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 pustaka.

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

Malang, 8 Januari 2019

Tsany Afif

NIM: 145150200111174

PRAKATA

Puji syukur peneliti panjatkan kepada Tuhan Yang Maha Esa atas limpahan rahmat dan petunjuk-Nya, sehingga peneliti dapat menyelesaikan skripsi yang berjudul "Implementasi Perangkat *Gateway* Untuk Pengiriman Data Sensor Dari Lapangan Ke Pusat Data Pada Jaringan *Wireless Sensor Network* Berbasis Perangkat NRF24L01".

Dalam penyusunan dan penelitian skripsi ini tidak lepas dari bantuan moral dan materiil yang diberikan dari berbagai pihak, maka peneliti mengucapkan banyak terima kasih kepada:

1. Bapak Adhitya Bhawiyuga, S.Kom, M.Sc. selaku dosen pembimbing I yang telah memberikan pengarahan dan bimbingan kepada peneliti, sehingga dapat menyelesaikan skripsi ini dengan baik.
2. Bapak Reza Andria Siregar, S.T., M.Kom. selaku dosen pembimbing II yang telah memberikan pengarahan dan bimbingan kepada peneliti, sehingga dapat menyelesaikan skripsi ini dengan baik.
3. Kedua orang tua dan seluruh keluarga besar atas segala nasehat, kasih sayang, perhatian, dan kesabarannya memberikan semangat kepada peneliti, serta senantiasa tiada hentinya memberikan doa demi terselesaikannya skripsi ini.
4. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Bapak Heru Nurwarsito, Ir., M.Kom. selaku Wakil Ketua I Bidang Akademik Fakultas Ilmu Komputer Universitas Brawijaya Malang.
6. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
7. Seluruh teman kelas L yang telah memberikan dukungan moral maupun meteriil yang sangat membantu penulis di masa masa sulitnya.
8. Rekan sesama pengerjaan skripsi Andi M.T Akbar, Alldo Raafi'Ilman, Rexa Mei Bella, Fauzan Pahlawan, Haidar Ari, Alfian, Sukma Alamsyah, Wisnu Surya Wardhana dan Richad Gilang.
9. Teman selama SMA Eka Prasetya Bayu Adjie, Beta Arif M dan Yudistira Sugandi yang telah dengan sabar mendukung dan mengunggu agar terselesaikannya laporan skripsi ini.
10. Seluruh pihak yang tidak dapat diucapkan satu persatu, peneliti mengucapkan banyak terima kasih atas segala bentuk dukungan dan doa sehingga laporan skripsi ini dapat terselesaikan.

Peneliti menyadari bahwa dalam penyusunan skripsi ini masih terdapat banyak kekurangan, sehingga saran dan kritik yang membangun sangat peneliti harapkan.

Akhir kata peneliti berharap skripsi ini dapat membawa manfaat bagi semua pihak yang membutuhkannya.

Malang, 8 Januari 2019

Tsany Afif

tsany.afif@gmail.com

ABSTRAK

Tsany Afif, Implementasi Perangkat *Gateway* Untuk Pengiriman Data Sensor Dari Lapangan Ke Pusat Data Pada Jaringan *Wireless Sensor Network* Berbasis Perangkat NRF24L01

Pembimbing : Adhitya Bhawiyuga, S.Kom, M.Sc dan Reza Andria Siregar, S.T., M.Kom.

Internet of Things (IoT) adalah suatu teknologi yang sedang berkembang. Salah satu teknologi yang menggunakan skenario IoT adalah WSN (*Wireless Sensor Network*). Pada penerapan konsep WSN terdapat kendala pada terbatasnya sumberdaya yang dimiliki. Terbatasnya sumberdaya menyebabkan pengolahan data pada WSN perlu dialihkan pada sistem lain seperti pusat data. Karena itu untuk menghubungkan komunikasi antara *node* sensor dengan pusat data diperlukan adanya perangkat *gateway*. Pada penelitian ini akan mengimplementasikan perangkat *gateway* yang akan menjembatani komunikasi antara *node* sensor dengan pusat data. Komunikasi *gateway* dengan *node* sensor dapat menggunakan modul nRF24L01 yang memiliki kelebihan pada konsumsi daya yang rendah dan mendukung protokol RF24Mesh yang dapat diterapkan pada sensor *wireless.* Pada komunikasi antara *gateway* dan pusat data dapat menggunakan protokol MQTT, dimana protokol ini sesuai untuk alat dengan sumber daya terbatas dan *bandwidth* rendah. Hasil pengujian performa *successfull rate* dari *gateway* yang dibangun dengan modul komunikasi *wireless* nRF24L01 menunjukkan bahwa performa *gateway* pada jarak 20 meter dengan ukuran paket 50 Byte dan jeda waktu pengiriman paket 1 detik memiliki performa yang paling baik. Pengiriman pada jarak 30 meter dan ukuran paket 150 *Byte* tidak dapat dilakukan karena melebihi jangkauan jarak dan besar ukuran paket.

Kata kunci: *Internet of Things, Wireless Sensor Network, gateway,* MQTT, nRF24L01, RF24Mesh

ABSTRACT

Tsany Afif, ***Implementation of Gateway Device for Sending Sensor Data from the Field to Data Centers on Wireless Sensor Network Based on NRF24L01 Device***

Advisor **: Adhitya Bhawiyuga, S.Kom, M.Sc** ***and*** **Reza Andria Siregar, S.T., M.Kom.**

Internet of Things (IoT) is currently developing technology. Wireless Sensor Network (WSN) is one of technology that use IoT scenario. In the implementation of WSN concept there is constraint on the limited resources that they have. Limited resources cause WSN data processing need to be handled in other systems such as data center. Therefore, to connect communication between sensor node and the data center, gateway device is needed. In this research gateway device will be implemented to bridging communication between sensor nodes and data centers. Communication between gateway and sensor nodes can use the nRF24L01 module which has advantages in low power consumption and supports RF24Mesh protocol which can be applied to wireless sensors. Communication between the gateway and data center can use the MQTT protocol, where this protocol is suitable for devices with limited resources and low bandwidth. The successful rate performance of the gateway built with the nRF24L01 wireless communication module show that the gateway performance at 20 meters distance with 50 Byte packet size and 1 second delay packet delivery has the best performance. Delivery at 30 meters distance and 150 Byte packet size cannot be done because exceeding the distance range and the packet size.

Keywords: *Internet of Things*, *Wireless Sensor Network*, *gateway*, MQTT, nRF24L01, RF24Mesh

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Internet of Things (IoT) adalah sebuah istilah yang digunakan untuk menjelaskan sebuah lingkungan dimana terdapat banyak objek ("*things*") yang dapat terkoneksi satu sama lain dan dapat berinteraksi dengan sendirinya. Objek di dalam IoT dapat berupa objek fisikal dari benda tidak bergerak sampai ke makhluk hidup ataupun berasal dari objek virtual (Cruz, Rodrigues, Al-Muhtadi, Korotaev, & Albuquerque, 2018). Perkembangan IoT terbilang pesat dikarenakan potensi yang dimiliki sangat besar. Salah satu teknologi yang mendukung perkembangan IoT adalah *Wireless Sensor Network*.

Wireless Sensor Network (WSN) adalah sebuah teknologi yang dapat digunakan untuk melakukan pemantauan kondisi lingkungan dengan memanfaatkan *node* sensor yang mampu melakukan pengolahan data yang tersebar dan terhubung dalam sebuah sistem melalui jaringan nirkabel (Pratama, Akbar, & Bhawiyuga, 2017). Dengan kata lain *Wireless Sensor Network* dapat digunakan untuk melakukan pemantauan atau *monitoring* suatu data lingkungan secara otomatis. Salah satu contoh penerapan *monitoring* dengan memanfaatkan teknologi *Wireless Sensor Network* adalah pengambilan data lingkungan pada bidang pertanian.

Menurut (Mendez, Yunus, & Mukhopadhyay, 2012) di bidang pertanian pada pengambilan data yang dilakukan secara manual mempunyai resiko kesalahan sehingga membutuhkan sebuah sistem yang dapat melakukan *monitoring* atau pemantauan data secara otomatis. Dengan sistem ini tentunya waktu dan usaha yang dibutuhkan dalam melakukan pemantauan dapat dikurangi sehingga menjadi lebih efisien. Pada bidang pertanian terdapat beberapa parameter yang mempengaruhi produksi hasil panen yaitu tingkat kelembaban dan suhu udara. Pada suhu udara, bergantung pada jenis tumbuhan yang di kembangkan suhu akan mempengaruhi perkembangan pada perkecambahan, pertumbuhan tunas, perkembangan bunga dan perkembangan buah (Mendez, Yunus, & Mukhopadhyay, 2012). Pada tingkat kelembaban udara, terlalu rendah atau tingginya kelembaban akan mempengaruhi produk pertanian. Ketika kelembaban berada di bawah 50 persen selama beberapa jangka waktu tertentu pertumbuhan dapat terganggu karena air yang dilepas oleh daun lebih cepat daripada air yang masuk. Ketika kelembaban berada di atas 80 persen selama beberapa jangka waktu tertentu akan meningkatkan risiko penyakit (Mendez, Yunus, & Mukhopadhyay, 2012). Kedua parameter tersebut dapat dipantau menggunakan sensor DHT11 yang mengambil nilai suhu dan kelembaban udara. Sensor DH11 inilah yang akan digunakan sebagai sensor pada *node* sensor di *Wireless Sensor Network*.

Pemilihan media komunikasi dalam membangun *Wireless Sensor Network* perlu melihat beberapa aspek. Penerapan perangkat pada konsep WSN membutuhkan sumber daya yang konstan untuk dapat berjalan. Oleh karena itu

selalu terdapat kebutuhan untuk mengurangi penggunaan sumber daya. Salah satu modul komunikasi yang dapat berjalan dengan sumber daya yang rendah adalah perangkat nRF24L01. Perangkat nRF24L01 adalah modul komunikasi nirkabel yang memanfaatkan frekuensi radio 2,4 Ghz dan memiliki konsumsi daya super rendah dengan baterai yang dapat bertahan dalam jangka waktu bulan sampai dengan tahun (NordicSemiconduktor, 2014). Pada penelitian sebelumnya yang telah dilakukan oleh (Pratama, Akbar, & Bhawiyuga, 2017) telah berhasil dirancang sebuah *low power node* sensor dengan menggunakan perangkat nRF24L01 sebagai modul komunikasinya. (Hanifah, Akbar, & Amron, 2018) dalam penelitiannya juga menyebutkan bahwa modul nRF24L01 memiliki kelebihan pada rendahnya konsumsi sumber daya dan harga yang lebih murah jika dibandingkan dengan xbee. Karena itu modul nRF24L01 ini cocok digunakan dalam media komunikasi pada WSN.

Setelah data pantauan pada WSN dapat ditangkap menggunakan *node* sensor, data tersebut perlu disimpan atau diperoses untuk diubah menjadi informasi yang berguna. (Habibi, Bhawiyuga, & Basuki, 2018) Dalam penelitiannya menyebutkan adanya kendala pada perangkat IoT dalam melakukan komputasi kompleks memerlukan adanya sebuah mekanisme lain. Salah satu solusi yang ada adalah mengalihkan proses komputasi dan penyimpanan pada sistem lain contohnya *cloud platform*. *Cloud platform* di sini berperan sebagai pusat data yang menjadi tempat pengolahan dan penyimpanan data. Namun pada penerapannya tidak semua sensor dapat mengirimkan data langsung menuju ke pusat data. Pada perangkat seperti modul komunikasi nRF24L01 pengiriman data menggunakan gelombang radio. Metode yang dapat digunakan untuk menghubungkan *node* sensor dengan pusat data ialah dengan mengimplementasikan sebuah perangkat *gateway*.

Gateway adalah sebuah perangkat yang menguhubungkan komunikasi pada sistem yang menggunakan protokol berbeda. Karena itu *gateway* memiliki fungsi umum yaitu mengkonversi protokol (Nuratch, 2017). Selain itu gateway juga dapat digunakan sebagai tempat atau titik pengumpulan data dari sensor untuk diteruskan menuju ke pusat data yang berada pada jaringan internet. *Gateway* disini akan berperan sebagai konfersi protokol yang digunakan pada komunikasi antara sensor dengan perangkat *gateway* nRF24L01 yang menggunakan gelombang radio dan juga protokol komunikasi antara *gateway* dengan pusat data yang berada pada jaringan internet.

Dari beberapa hal yang telah di sebutkan di atas, peneliti mengajukan untuk melakukan sebuah penelitian dengan mengimplementasikan sebuah perangkat *gateway* yang bertujuan untuk menjadi perantara komunikasi dari lapangan menuju ke pusat data. Pada penelitian ini perangkat *gateway* dibangun dengan menggunakan mikrokontroler Raspberry Pi yang telah dipasangi dengan modul *wireless* yaitu nRF24L01. Sensor *node* dibangun dengan mikrokontroler Arduino Nano yang telah dipasangi dengan modul *wireless* nRF24L01 dan sensor DHT11. Protokol komunikasi yang digunakan pada pengiriman data yaitu RF24Mesh dan MQTT.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah dapat dituliskan sebagai berikut :

1. Bagaimana mengirimkan data sensor dari *node* sensor menuju ke *gateway* dengan menggunakan perangkat nRF24L01?
2. Bagaimana mengimplementasikan perangkat *gateway* untuk meneruskan data dari lapangan ke pusat data?
3. Bagaimana hasil kinerja *gateway* yang telah diterapkan dalam pengiriman data antara *node* sensor dan pusat data menggunakan perangkat nRF24L01?

1.3 Tujuan

Dengan dirumuskannya rumusan masalah, dapat ditentukan tujuan dari skripsi ini sebagai berikut :

1. Untuk menjelaskan cara pengiriman data sensor dari *node* sensor menuju ke perangkat *gateway* menggunakan perangkat nRF24L01.
2. Untuk menjelaskan cara pengimplemetasian perangkat *gateway* untuk meneruskan data pada pengiriman data sensor dari lapangan ke pusat data.
3. Untuk mengetahui hasil kinerja perangkat *gateway* yang telah dibangun menggunakan perangkat nRF24L01.

1.4 Manfaat

Dilihat dari fungsi yang akan diteliti penulis, didapatkan manfaat sebagai berikut :

1. Mengetahui cara pengimplementasian perangkat *gateway* sebagai penjembatan komunikasi pada pengiriman data sensor dari lapangan ke pusat data dengan menggunkan perangkat nRF24L01.
2. Mengetahui kinerja perangkat *gateway* yang dibangun untuk pengiriman data sensor dari lapangan ke pusat data berbasis perangkat nRF24L01.
3. Hasil dari penelitian ini dapat membantu para peneliti dalam pengembangan teknologi IoT khususnya pada perbandingan kinerja dan pengimplementasian perangkat nRF24L01.

1.5 Batasan Masalah

Pada penelitian ini terdapat beberapa batasan masalah berdasarkan latar belakang dan rumusan masalah di atas, yaitu :

1. Menggunakan modul *wireless* nRF24L01 untuk media komunikasi antara *gateway* dan *node* sensor.

2. Menggunakan protokol RF24Mesh untuk pengiriman data antara *node* sensor dengan *gateway*.
3. Menggunakan protokol komunikasi *Message Queuing Telemetry Transport* (MQTT) untuk media komunikasi antara *gateway* dan pusat data.
4. Sensor yang digunakan dalam penelitian adalah sensor DHT11.
5. Mikrokontroler yang digunakan pada *gateway* adalah raspberry pi 3 model b, karena pada model ini telah menyediakan pin yang dibutuhkan untuk menerapkan modul nRF24L01 dan telah memiliki modul *wifi* di dalamnya.
6. Protokol MQTT menggunakan qos 1.

1.6 Sistematika Pembahasan

Sistematika pembahasan dari Implementasi Perangkat *Gateway* Untuk Pengiriman Data Sensor Dari Lapangan Ke Pusat Data Pada Jaringan *Wireless* Sensor *Network* Berbasis Perangkat nRF24L01 adalah sebagai berikut:

BAB 1 PENDAHULUAN

Pendahuluan menjelaskan mengenai latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan dari Implementasi Perangkat *Gateway* Untuk Pengiriman Data Sensor Dari Lapangan Ke Pusat Data Pada Jaringan *Wireless* Sensor *Network* Berbasis Perangkat nRF24L01.

BAB 2 LANDASAN KEPUSTAKAAN

Membahas berbagai landasan teori dasar dan refrensi literatur pendukung yang terkait dengan Implementasi Perangkat *Gateway* Untuk Pengiriman Data Sensor Dari Lapangan Ke Pusat Data Pada Jaringan *Wireless* Sensor *Network* Berbasis Perangkat nRF24L01.

BAB 3 METODOLOGI PENELITIAN

Membahas tahapan yang dilakukan dalam penulisan diantaranya studi literatur, analisis kebutuhan sistem, desain sistem dan Implementasi Perangkat *Gateway* Untuk Pengiriman Data Sensor Dari Lapangan Ke Pusat Data Pada Jaringan *Wireless* Sensor *Network* Berbasis Perangkat nRF24L01.

BAB 4 REKAYASA KEBUTUHAN

Bagian ini membahas analisis kebutuhan yang dibutuhkan dalam pengembangan sistem agar sistem implementasi *gateway* dapat lebih terarah dan sistematis.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Membahas perancangan dan implementasi *gateway* menggunakan protokol RF24Mesh-MQTT pada perangkat nRF24L01.

BAB 6 PENGUJIAN DAN ANALISIS

Membahas pengujian dan analisis berdasarkan perancangan dan implementasi.

BAB 7 PENUTUP

Memuat kesimpulan yang merangkum jawaban dari rumusan masalah berdasarkan hasil penelitian yang sudah di dapatkan dan juga saran untuk pengembangan penelitian lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi tentang studi literatur dan dasar dasar teori yang terkait dengan metode yang berhubungan dengan penelitian. Pada studi literatur akan dibahas beberapa penelitian terdahulu yang memiliki kemiripan dengan tema penelitian yang diangkat. Pada dasar teori akan dilakukan penjabaran tentang metode dan teori yang digunakan pada penelitian.

2.1 Tinjauan Pustaka

Beberapa penelitian terdahulu yang memiliki kemiripan dengan tema penelitian penulis dapat dilihat pada Tabel 2.1 Tinjauan Pustaka sebagai berikut :

Tabel 2.1 Tinjauan Pustaka

No	Judul	Kemiripan	Penelitan Terdahulu	Penelitian Penulis
1	Rancang Bangun *Low Power* Sensor *Node* Menggunakan MSP430 Berbasis NRF24L01. (Pratama, Akbar, & Bhawiyuga, 2017)	Menggunakan modul komunikasi nRF24L01 dalam membangun sebuah *node* sensor.	Pembuatan *node* sensor menggunakan mikrokontroler MSP430G2553 yang pengiriman datanya menggunakan modul komunikasi NRF24L01.	Peneliti menggunakan mikrokomputer arduino nano yang terhubung dengan modul komunikasi nRF24L01 untuk pengiriman data sensor pada *node* sensor yang dibuat.
2	Rancang Bangun IOT *Cloud Platform* Berbasis Protokol Komunikasi MQTT. (Habibi, Bhawiyuga, & Basuki, 2018)	Menggunakan protokol MQTT untuk pengiriman data pada skenario IoT yang memiliki keterbatasan komputasi.	Menerapkan protokol komunikasi MQTT pada pengiriman data dari *node* sensor menuju ke *cloud platform*.	Peneliti membuat sebuah *gateway* yang menerapkan protokol MQTT untuk pengiriman data dari *node* sensor menuju pusat data.
3	The IIOT Device to Cloud *Gateway* Design and Implementation based on Microcontroller	Membuat sebuah *gateway* yang menerapkan 2 protokol berbeda untuk	Membangun sebuah *gateway* menggunakan mikrokontroler MCU dengan menerapkan protokol	Membangun sebuah *gateway* menggunakan mikrokontroler raspberry pi yang mampu mentranslasikan

	for Real-time Monitoring and Control in Automation Systems. (Nuratch, 2017)	pengiriman data ke pusat data.	websocket dan Modbus-RTU protokol.	2 protokol berbeda yaitu RF24 dan MQTT untuk pengiriman data dari *node* sensor menuju ke pusat data.

2.2 Dasar Teori

2.2.1 *Internet of Things*

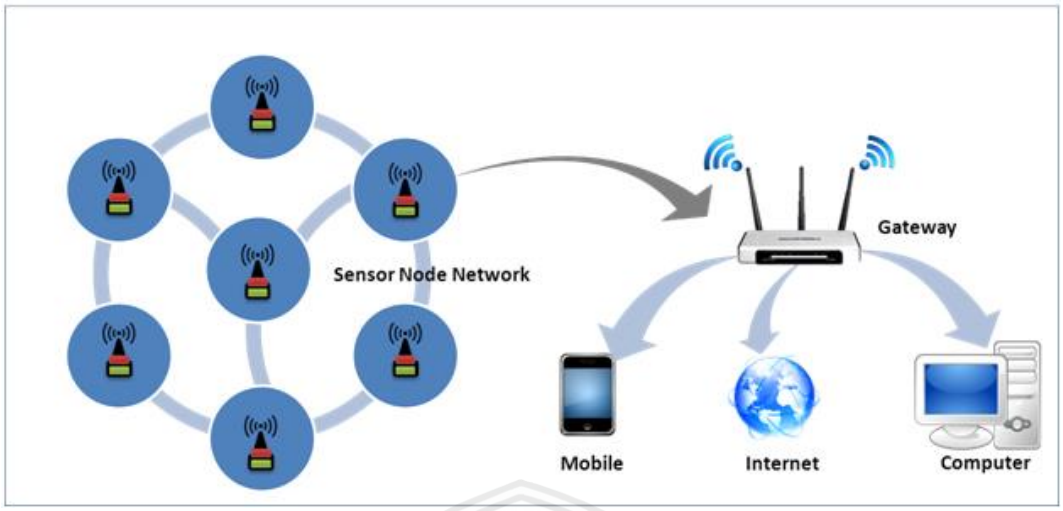
Internet of Things (IoT) adalah sebuah istilah yang digunakan untuk menjelaskan sebuah lingkungan dimana terdapat banyak objek ("*things*") yang dapat terkoneksi satu sama lain dan dapat berinteraksi dengan sendirinya (Cruz, Rodrigues, Al-Muhtadi, Korotaev, & Albuquerque, 2018). Melihat definisi dari IoT dapat disimpulkan bahwa IoT menggunakan prinsip dasar bahwa *thing* dapat berbicara dengan bahasa yang sama, menggunakan teknologi yang dapat digunakan untuk berkomunikasi di antara mereka.

Thing yang di maksud pada skenario IoT ialah sebuah perangkat yang dapat diidentifikasi secara unik dan terhubung dengan internet, contohnya *things* dapat berupa hal yang berbentuk fisikal mulai dari benda tidak bergerak sampai ke binatang hidup ataupun *things* dapat juga berasal dari lingkungan virtual. (Cruz, Rodrigues, Al-Muhtadi, Korotaev, & Albuquerque, 2018). Kebanyakan dari aplikasi IoT menggunakan protokol TCP atau UDP untuk melakukan *transport,* pada protokol aplikasi banyak digunakan XMPP, CoAP, DDS, MQTT dan AMQP (Yassein, Shatnawi, Aljwarneh, & Al-Hatmi, 2017).

2.2.2 *Wireless* Sensor *Network*

Wireless Sensor *Network* (WSN) adalah teknologi yang digunakan untuk melakukan pemantauan yang terdiri atas dua atau lebih *node* sensor yang dikordinasikan oleh sebuah sistem dengan menggunakan jaringan nirkabel (Pratama, Akbar, & Bhawiyuga, 2017). Setiap *node* umumnya memiliki kemampuan pengolahan data, memiliki memori, *transceiver,* sistem catu daya dan melibatkan sensor atau aktuator (Pratama, Akbar, & Bhawiyuga, 2017). Sensor berguna untuk mengkonversi sifat fisikal menjadi kuantitas numeric (Stanley, Lee, & Spanias, 2017).

Pada Gambar 2.1 menggambarkan arsitektur pada jaringan WSN. Jaringan ini terdiri dari banyak *node* sensor yang tersebar dan terhubung satu sama lainnya dalam sebuah jaringan nirkabel. Terdapat sebuah *gateway* yang berfungsi sebagai penghubung antara *node* sensor dengan *user*. *Gateway* sendiri dapat disebut dengan *sink* yang berarti pengumpul dari dari yang telah ditangkap oleh sensor. User pada gambar di bawah dapat berupa komputer ataupun ponsel yang akan menangani pemrosesan data yang telah didapatkan oleh *node* sensor.



Gambar 2.1 Arsitektur *Wireless Sensor Network*

Sumber: telcominded.wordpress.com (2018)

2.2.2.1 Sensor DHT11

Sensor DHT11 adalah sensor yang dapat digunakan untuk mengukur dua parameter lingkungan sekaligus yaitu suhu dan kelembaban udara. Sensor ini dapat dihubungkan dengan mikrokomputer arduino dengan menggunakan 4 pin yang terdapat pada sensor yaitu VCC, DATA, NC dan GND.

2.2.3 *Radio Frequency Identification*

Radio Frequecy Identification (RFID) adalah sistem yang mempunyai fungsi utama untuk identifikasi dan pelacakan objek, oleh karena itu objek wajib untuk memiliki identitas digital yang unik (Bade & Lamersdorf, 2011). RFID bekerja dengan membaca informasi elektronik yang terdapat pada *tags* yang terdapat pada objek dengan menggunakan sensor khusus.

Beberapa contoh penerapan teknologi RFID yang telah ada ialah pelacakan aset, manajemen gudang dan perkapalan serta kontrol manufaktur (Bade & Lamersdorf, 2011). Salah satu modul komunikasi yang telah menerapkan teknologi RFID adalah nRF24L01 dan nRF24L01+. Kedua perangkat tersebut cocok diterapkan untuk membangun komunikasi pada *low power node* sensor.

2.2.3.1 nRF24L01

NRF24L01 adalah sebuah modul komunikasi serial nirkabel yang digunakan pada pita frekuensi ISM (*Industrial, Scientific and Medical*) 2,4 GHz. Perangkat nRF24L01 ini didesain untuk aplikasi *ultra low power wireless* (Pratama, Akbar, & Bhawiyuga, 2017). Module nRF24L01 memiliki perangkat keras yang berupa *baseband logic Enhanced ShockBurst* dan *protocol accelerator* yang memungkinan untuk berkomunikasi dalam kecepatan tinggi. Selain itu, module ini juga memiliki fitur *true ULP solution*, yang berfungsi sebagai penghemat konsumsi daya sehingga hemat energi. Modul ini menggunakan antarmuka SPI (*Serial Parallel Interface*) untuk berkomunikasi dengan mikrokontroler. Modul NRF24L01 memiliki

kecepatan sampai 2Mbps dengan pilihan opsi *data rate* 250 Kbps, 1 Mbps, dan 2 Mbps. Konsumsi arus yang digunakan sangat rendah, hanya 9.0mA pada daya *output* -6dBm dan12.3mA dalam mode RX(Shobrina, Primananda, & Maulana, 2018). Gambar 2.2 menampilkan perangkat nRF24L01.

Gambar 2.2 Modul *Wireless* nRF24L01

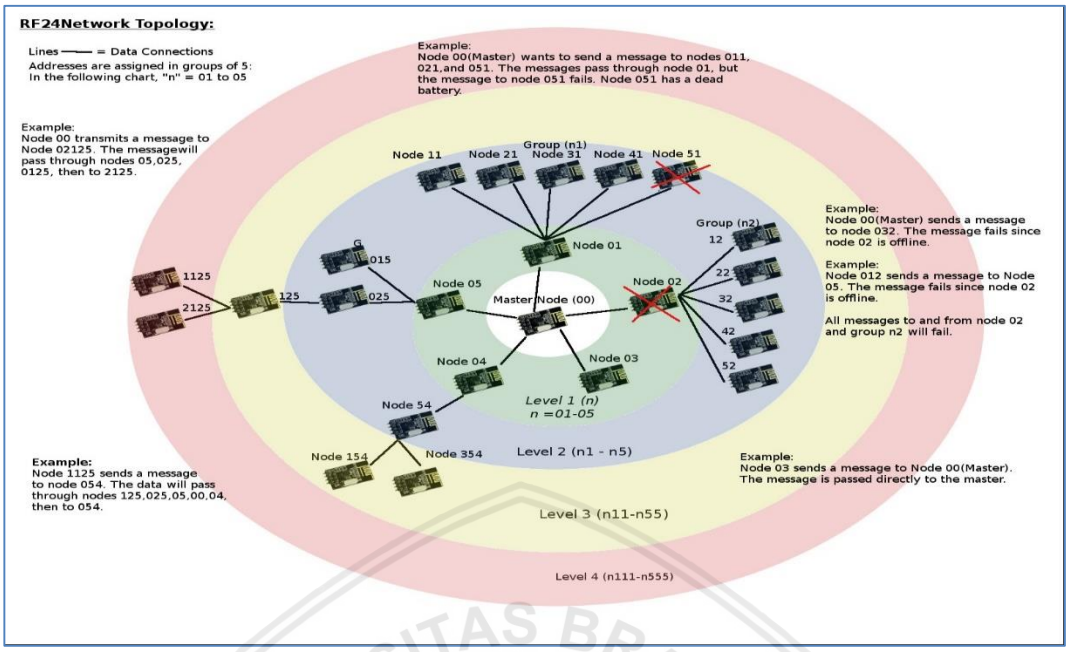
Sumber: www.hobbyandyou.com (2018)

Modul nRF24L01 memiliki 8 buah pin diantaranya VAC (3.3 Volt), GND (*Ground*), CE(*Chip Enable*), CSN (*Chip Select Not*), MOSI (*Master Output Slave Input*), MISO (*Master Input Slave Output*), SCK (SPI Clock), dan IRQ (*Interrupt Request*). Fitur lain yang ditawarkan dari modul ini yaitu jangkauan pengiriman 100 meter di tempat terbuka, penanganan paket data dan penanganan transaksi paket yang otomatis (Af'idah, Rochim, & Widianto, 2014). Tipikal modul nRF24 radio menggunakan 40-bit format alamat, membutuhkan 5 *byte* tempat penyimpanan per alamat dan komunikasi langsung antar radio terbatas hanya pada 6 *node* lain (tmrh20, 2015). Telah dikembangkan beberapa *library* yang dapat membantu dalam penggunaan perangkat nRF24L01 diantaranya *library* RF24, RF24Network dan RF24Mesh.

2.2.3.2 Protokol RF24Mesh

Protokol RF24Mesh adalah protokol komunikasi yang nantinya akan digunkan pada modul nRF24L01. Protokol ini bekerja pada lapisan *network* yang mengatur komunikasi pada *radio frequency* (RF) dengan panjang pita 2,4 GHz. Protokol RF24Mesh ini pada dasarnya adalah pengembangan dari protokol RF24Network yang memiliki kelebihan pada *automatic addressing* dan *dynamic configuration* pada sensor *wireless* (tmrh20, 2015).

Pada Gambar 2.3 menjelaskan komunikasi dan topologi pada RF24Network. Ketika transmisi terjadi pada suatu modul radio ke lainnya, penerima akan mengirimkan ack kepada si pengirim ketika transmisi berhasil terjadi. Radio pada jaringan ini terhubung pada alamat dan *pipes*. Setiap radio dapat terhubung dengan 6 alamat dan 6 *pipes*, oleh karena itu setiap radio dapat mempunyai 1 *parent pipes* dan 5 *child pipes*. Pada Gambar 2.3 *node master* berperan sebagai *gateway* dapat terhubung secara langsung dengan 4 *node* anak. Yang nantinya setiap *node* anak juga dapat terhubung secara langsung dengan 4 *node* lain.



Gambar 2.3 Topologi Pada RF24Network

Sumber: tmrh20.github.io (2018)

Contoh pengalamatan pada jaringan ini *master node* memiliki alamat 00, anak dari *master node* dapat memiliki alamat 01, 02, 03, 04, 05 dan anak dari 01 dapat memiliki alamat 011, 021, 031, 041, 051. *Routing* pada jaringan ini ditangani tanpa terihat oleh *user*. *User* hanya perlu membuat *header* yang berisi alamat *node* tujuan dan pada jaringan akan diteruskan menuju ke *node* tujuan. RF24Network menerapkan sebuah metode kompresi data untuk dapat menyimpan alamat hanya dengan menggunakan ukuran 2 *byte* (tmrh20, 2015).

2.2.4 Arduino

Arduino adalah sebuah platform elektronik *open-source* yang berbasis perangkat keras dan perangkat lunak yang mudah digunakan. Arduino pertama kali dikembangkan di Ivrea Interaction Design Institute sebagai alat *prototyping* yang mudah dan ditargetkan kepada pelajar tanpa latar belakang elektro dan pemrograman.

Produk Arduino mencakup aplikasi IoT, printer 3D, *embedded environtment* dan masih banyak lagi. Arduino memiliki kelebihan diantaranya sederhana, tidak mahal, *cross platform, open-source* dan *extensible software* dan *hardware* (Arduino, 2018).

2.2.4.1 Arduino Nano

Arduino Nano adalah sebuah perangkat mikrokontroler dari Arduino yang berukuran kecil, lengkap dan *breadboard-friendly board* berbasis Atmega328P(Arduino Nano 3.x) (Arduino, 2018). Memiliki fungsionalitas yang kurang lebih sama dengan Arduino Duemilanove namun paket yang berbeda dan dapat bekerja dengan menggunakan kabel USB mini (Arduino, 2018). Selain

dilengkapi dengan flash memori, mikrokontroller ATmega168 dan ATmega328 juga dilengkapi dengan SRAM dan EEPROM. SRAM dan EEPROM dapat digunakan untuk menyimpan data selama program utama bekerja. Besar SRAM untuk ATmega168 adalah 1 kb dan untuk ATmega328 adalah 2 kb sedangkan besar EEPROM untuk ATmega168 adalah 512 b dan untuk ATmega328 adalah 1 kb.

Arduino Nano mempunyai 14 pin digital yang dapat digunakan sebagai pin input atau output. Arduino Nano juga dilengkapi dengan 8 buah pin analog. Gambar 2.4 menampilkan bentuk dari perangkat Arduino Nano. Arduino Nano dapat menggunakan catudaya langsung dari mini-USB port atau menggunakan catudaya luar yang dapat diberikan pada pin30 (+) dan pin29 (-) untuk tegangan kerja 7 – 12 V atau pin 28(+) dan pin 29(-) untuk tegangan 5V.

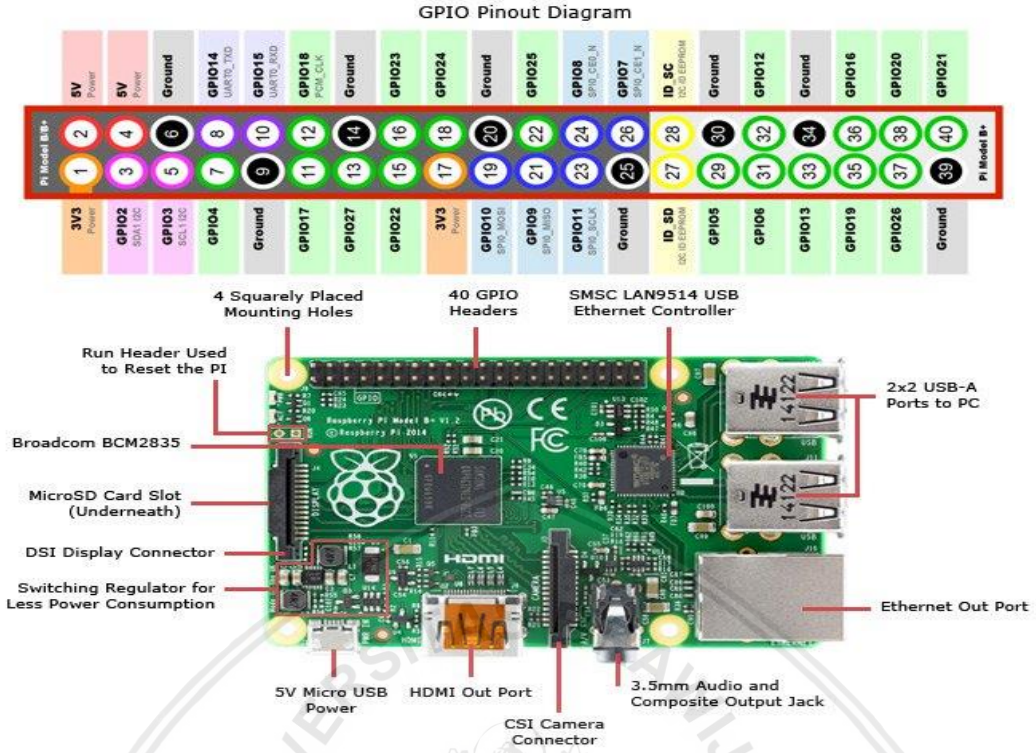
Gambar 2.4 Arduino Nano

Sumber: www.arduino.cc (2018)

2.2.5 Raspberry Pi

Raspberry Pi adalah komputer seukuran kartu kredit denga biaya rendah yang dapat dihubungkan dengan tv atau monitor dan menggunakan *keyboard* dan *mouse* standar (Raspberry Pi, 2018). Raspberry Pi memiliki kemampuan untuk berinteraksi dengan dunia luar dan telah banyak digunakan dalam pembuatan proyek digital, mulai dari mesin music sampai stasiun cuaca, dapat juga digunakan untuk pemrosesan kata, memainkan *video* dengan kualitas gambar yang tinggi hingga bermain gim (Raspberry Pi, 2018). Raspberry Pi 3 menggunakan pin header general-purpose-input-output (GPIO). Terdapat 40 pin GPIO pada raspberry. Gambar 2.5 menampilkan komponen komponen yang ada pada raspberry pi dan juga detail dari pinout GPIO.

Raspberry Pi memiliki dua model: model A dan model B.. Perbedaan model A dan B terletak pada modul penyimpanan yang digunakan. Model A menggunakan penyimpanan sebesar 256 MB dan penyimpanan model B sebesar 512 MB. Selain itu, model B sudah dilengkapi dengan port Ethernet (untuk LAN) yang tidak terdapat di model A. Desain Raspberry Pi didasarkan pada SoC (system-on-a-chip) Broadcom BCM2835, yang telah menanamkan prosesor ARM1176JZF-S dengan 700 MHz, GPU VideoCore IV, dan RAM sebesar 256 MB (model B). Penyimpanan data tidak didesain untuk menggunakan cakram keras atau solid-state drive, melainkan mengandalkan kartu penyimpanan tipe SD untuk menjalankan sistem dan sebagai media penyimpanan jangka panjang.



Gambar 2.5 Perangkat Raspberry Pi 3

Sumber: www.jameco.com (2018)

2.2.6 *Gateway*

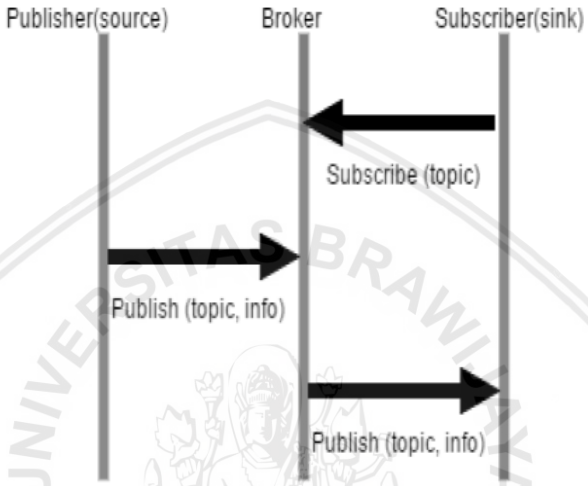
Gateway adalah sebuah perangkat yang berperan sebagai penjembatan antara 2 program komputer atau sistem. Dalam skenario *gateway* IoT, *gateway* dapat menjembatani antara sensor dengan sebuah pusat data dengan perantara sebuah mikrokontroler. Karenanya *gateway* yang diimplementasikan pada mikrokontroler berperan sebagai konversi protokol dan sistem keamanan (Nuratch, 2017). *Gateway* akan meneruskan parameter yang ditangkap oleh sensor menuju ke pusat data, dan juga *gateway* akan meneruskan *request* yang dikirimkan pusat data menuju perangkat *node* sensor (Nuratch, 2017).

2.2.7 *Message Queuing Telemetry Transport*

Message Queuing Telemetry Transport (MQTT) adalah sebuah protokol pertukaran pesan yang dikenalkan oleh Andy Stanford-Clark dari IBM dan Arlen Nipper pada tahun 1999 (Yassein, Shatnawi, Aljwarneh, & Al-Hatmi, 2017). Tujuan dari protokol MQTT adalah untuk penerapan pada perangkat yang memiliki kapabilitas memori dan kekuatan pemrosesan yang terbatas, oleh karena itu MQTT dianggap sebagai protokol koneksi yang paling menguntungkan pada M2M (*machine to machine*) dan IoT (Yassein, Shatnawi, Aljwarneh, & Al-Hatmi, 2017).

MQTT menggunakan model *client server* dan berbasis pada pola komunikasi *publish*/*subscribe* (Yassein, Shatnawi, Aljwarneh, & Al-Hatmi, 2017). Selain itu dalam MQTT juga terdapat sebuah peran yang bernama *broker*. *Broker* memiliki tanggung jawab untuk memproses komunikasi antar client dan mendistribusikan

pesan berdasarkan topik yang diinginkan oleh *client*. *Client* dalam MQTT dapat berupa *subscriber* maupun *publisher* tergantung dari tugasnya, jika bertugas membuat konten maka menjadi *publisher* dan jika meminta konten maka menjadi *subscriber*. Gambar 2.6 menunjukkan proses *publish/subscribe* yang ada pada MQTT. Seperti yang terlihat dalam Gambar 2.6 *publisher* akan membuat konten / isi(dalam gambar tersebut info) sesuai dengan topic tertentu dan *subscriber* akan meminta sebuah konten dengan mengirimkan topik yang diinginkan. Sedangkan *broker* hanya akan meneruskan konten dengan topic yang sesuai dengan permintaan *subscriber* dari *publisher*.



Gambar 2.6 Proses *Publish/Subscribe* MQTT

Sumber : (Yassein, Shatnawi, Aljwarneh, & Al-Hatmi, 2017)

2.2.8 Parameter Pengujian

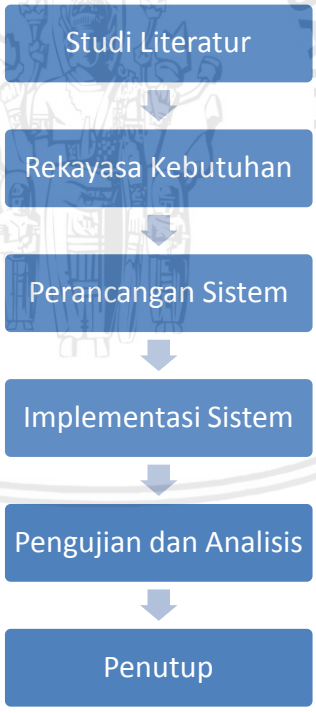
Parameter yang diuji pada penelitian ini adalah *successfull rate* pada perangkat *gateway*. *Successfull rate* adalah tingkat keberhasilan perangkat *gateway* dalam meneruskan data dari *node* sensor menuju ke *server*. Parameter ini akan dihitung ketika perangkat *gateway* menerima data dari *node* sensor lalu perangkat *gateway* menerukan datanya menuju ke *server*. Persamaan pada parameter pengujian *successfull rate* yaitu sebagai berikut :

$$successfull\ rate = \left(\frac{total\ paket\ dikirim - total\ paket\ gagal}{total\ paket\ dikirim} \times 100\%\right)$$

BAB 3 METODOLOGI

Dalam bab ini akan menjelaskan sistematika dalam penulisan skripsi ini, dari tahap pertama sampai keseluruhan penelitian. Proses awal dari penelitian ini adalah mencari literatur yang sesuai dengan topik penelitian. Analisis Kebutuhan untuk mencari kebutuhan kebutuhan apa saja yang dibutuhkan untuk membangun sistem. Perancangan sistem adalah proses untuk memberikan gambaran umum tentang rancangan dari sistem yang akan di buat. Implementasi sistem adalah tahap dimana sistem dibuat berdasarkan rancangan yang telah ada sebelumnya. Pengujian dan analisis adalah tahap dimana sistem yang telah selesai diimplemetasikan untuk di uji dan dianalisis kinerjanya, diharapkan dari tahap ini juga dapat di ambil kesimpulan untuk tahap berikutnya. Kesimpulan adalah hasil akhir yang didapatkan dari awal hingga akhir penelitian.

Alur penelitian yang diterapkan dalam implementasi perangkat *gateway* untuk pengiriman data sensor dari lapangan ke pusat data pada jaringan *wireless sensor network* berbasis perangkat nRF24L01 akan di gambarkan secara singkat pada Gambar 3.1 dalam bentuk diagram alir metodologi penelitan.



Gambar 3.1 Diagram Alur Metodologi Penelitian

3.1 Studi Literatur

Sebagai penunjang dalam penulisan skripsi, dilakukan studi literatur yang bertujuan untuk mencari dasar teori dan kajian pustaka. Dasar teori didapatkan dari beberapa jurnal, buku, dokumentasi *project* yang memiliki relevansi dengan judul dan topik penelitian ini. Kajian pustaka yang digunakan adalah beberapa penelitian terdahulu yang memiliki kesamaan dengan topik dan judul penelitian ini. Beberapa studi literatur yang berkaitan dengan penelitian ini yaitu :

1. *Internet of Things* (IoT)

   Pada bagian ini menjelaskan tentang konsep dari *internet of things* dan menjelaskan komponen yang membentuk *internet of things.* Bagian ini juga menyebutkan beberapa protokol yang banyak digunakan pada pengaplikasian *internet of things*.

2. Wireless Sensor Network

   Pada bagian ini menjelaskan tentang konsep *wireless sensor network* dan komponen yang ada dalam *wireless sensor network*. Pada sub bagian ini dijelaskan salah satu jenis sensor yang akan digunakan pada penelitian ini.

3. *Radio Frequency Identification* (RFID)

   Pada bagian ini menjelaskan tentang konsep *Radio Frequency Identification* (RFID) dan beberapa contoh penerapan konsep RFID. Pada sub bagian ini mendeskripsikan perangkat modul wireless yang menggunakan radio frekuensi dan protokol yang digunakannya dalam pertukaran data.

4. Arduino

   Pada bagian ini menjelaskan tentang perangkat mikrokontroler arduino, pengaplikasian perangkat arduino yang telah ada dan kelebihan dari perangkat arduino. Pada sub bagian ini menjelaskan perangkat lunak yang digunakan untuk memprogram perangkat arduino dan spesifikasi salah satu produk arduino yang akan digunakan pada penelitian ini.

5. Raspberry Pi

   Pada bagian ini menjelaskan tentang perangkat mikrokomputer raspberry pi yang akan digunakan sebagai *gateway* pada penelitian ini dan beberapa pengaplikasian yang dapat dilakukan oleh perangkat raspberry pi.

6. *Gateway*

   Pada bagian ini akan menjelaskan tentang konsep *gateway* dan fungsi perangkat *gateway* pada sebuah sistem.

7. Message Queuing Telemetry Transport (MQTT)

   Pada bagian ini menjelaskan tentang konsep protokol MQTT, komponen yang ada pada MQTT dan tugas dari komponen yang ada dalam protokol MQTT.

3.2 Rekayasa Kebutuhan

Rekayasa kebutuhan dari penelitian ini dilakukan untuk menentukan kebutuhan yang diperlukan untuk berjalannya penelitian ini. Dengan adanya rekayasa kebutuhan diharapkan dapat menjadi acuan pada tahap perancangan dan implementasi agar kedua tahap tersebut dapat berjalan dengan lebih terarah dan sistematis.

3.2.1 Analisis Kebutuhan Sistem

Pada analisis kebutuhan sistem akan membahas tentang kebutuhan apa saja yang harus terpenuhi pada sistem agar dapat berjalan sesuai dengan tujuan awal pembuatan sistem.

3.2.2 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras akan membahas tentang kebutuhan perangkat keras yang harus terpenuhi untuk dapat berjalannya sistem. Terdapat 3 perangkat keras yang dibutuhkan oleh sistem yaitu *node* sensor, *gateway* dan *server*.

3.2.3 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak akan membahas tentang kebutuhan perangkat lunak yang harus terpenuhi untuk dapat berjalannya sistem. Dibutuhkan perangkat lunak untuk mengoperasikan perangkat perangkat keras yang telah ditentukan sebelumnya. Sistem operasi, *text editor* dan aplikasi ssh merupakan perangkat lunak yang dibutuhkan dalam perancangan serta implementasi sistem.

3.3 Perancangan Sistem

Pada perancangan sistem akan membahas perancangan apa saja yang dilakukan untuk membangun sistem berdasarkan rekayasa kebutuhan yang telah dilakukan sebelumnya agar sistem dapat berjalan sesuai dengan tujuan. Perancangan sistem terdiri dari perancangan topologi jaringan, perancangan data, perancangan protokol, perancangan *node* sensor, perancangan *gateway*, perancangan *server*, perancangan parameter pengujian dan perancangan skenario pengujian.

3.3.1 Perancangan Topologi Jaringan

Perancangan topologi jaringan dibuat untuk memberikan gambaran secara umum topologi yang akan dibuat pada sistem.

3.3.2 Perancangan Data

Perancangan data dilakukan untuk menjelaskan data yang dibuat dan alur data yang berjalan pada sistem. Perancangan data akan meliputi data pada pengiriman sensor dan pengiriman *server*.

3.3.3 Perancangan Protokol

Perancangan protokol dilakukan untuk menjelaskan protokol yang digunakan pada sistem. Perancangan protokol akan memuat tentang protokol RF24Mesh dan protokol MQTT.

3.3.4 Perancangan *Node* Sensor

Perancangan *node* sensor dilakukan untuk menentukan komponen komponen yang membangun *node* sensor pada sistem yang dibuat. Perancangan *node* sensor juga akan menjelaskan alur kerja pada *node* sensor.

3.3.5 Perancangan *Gateway*

Perancangan *gateway* dilakukan untuk menetukan komponen yang menyusun perangkat *gateway* pada sistem yang dibuat. Perancangan *gateway* juga akan membahas alur kerja pada perangkat *gateway*.

3.3.6 Perancangan *Server*

Perancangan *server* dilakukan untuk menentukan komponen yang menyusun perangkat *server* pada sistem yang dibuat. Perancangan *server* juga akan membahas alur kerja pada perangkat *server*.

3.3.7 Perancangan Parameter Pengujian

Perancangan parameter pengujian dilakukan untuk menentukan parameter yang akan digunakan dalam menilai kinerja dari perangkat *gateway* yang telah dibangun. Dalam perancangan parameter pengujian juga akan ditentukan variasi dari parameter yang diuji.

3.3.8 Perancangan Skenario Pengujian

Perancangan skenario pengujian dilakukan untuk menentukan skenario skenario yang akan dilakukan pada pengujian sistem sesuai dengan parameter pengujian yang telah ditentukan sebelumnya.

3.4 Implementasi

Pada tahap ini akan dilakukan pemasangan dan konfigurasi pada beberapa komponen pembangun sistem. Implementasi dibuat berdasarkan rancangan yang telah di buat pada perancangan sistem. Pada tahap implementasi terdapat 3 komponen yang akan di implementasikan yaitu implementasi *node* sensor, implementasi perangkat *gateway* dan implementasi *server*.

3.4.1 Implementasi *Node* Sensor

Implementasi *node* sensor dilakukan dengan mengimplementasikan protokol dan membangun seluruh komponen yang telah disebutkan pada perancangan. Menghubungkan seluruh komponen dengan kabel jumper sesuai dengan petunjuk pin *layout*. Selanjutnya mengimplementasikan protokol pada kode program untuk *node* sensor. Protokol yang diimplementasikan pada *node* sensor adalah protokol RF24Mesh.

3.4.2 Implementasi *Gateway*

Implementasi perangkat *gateway* dilakukan dengan membangun perangkat keras yang telah disebutkan pada perancangan dan menghubungkan seluruh komponen perangkat kerasnya. Lalu mengimplementasikan protokol dalam kode program pada perangkat *gateway*. Protokol yang diimplementasikan pada *node* sensor adalah protokol RF24Mesh dan MQTT.

3.4.3 Implementasi *Server*

Implementasi *server* dilakukan dengan membangun perangkat *server* yang telah dijelaskan pada perancangan. Pengimplementasian protokol pada *server* akan dilakukan dalam kode program yang dibuat. Protokol yang diimplementasikan pada *node* sensor adalah protokol MQTT.

3.5 Pengujian dan Analisis

Pada tahap pengujian dan analisis akan dilakukan pengujian unit yang telah disiapkan pada tahap implementasi. Tahap pengujian akan dibagi menjadi pengujian fungsional dan pengujian performa sesuai dengan skenario dan parameter yang telah ditentukan. Pada bagian analisis akan dianalisa data yang didapatkan dari hasil pengujian untuk melihat apakah sistem bekerja sesuai dengan yang telah diharapkan.

3.6 Penutup

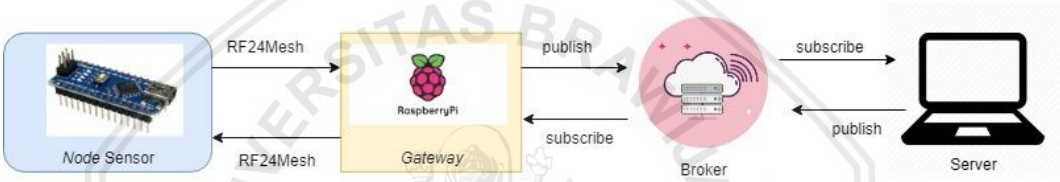
Pada bagian penutup akan disajikan kesimpulan hasil analisa pada tahap-tahap sebelumnya. Lalu akan ditarik kesimpulan atas kinerja dari rancangan yang telah dibuat oleh penulis dari sisi kelebihan maupun kekurangan yang muncul pada sistem yang telah dibuat penulis. Penulis juga akan menyertakan saran penyempurnaan yang dapat dilakukan pada penelitian selanjutnya.

BAB 4 REKAYASA KEBUTUHAN

Analisis kebutuhan dibutuhkan dalam pengembangan sistem agar sistem dapat lebih terarah dan sistematis. Pada bab ini akan dibahas daftar kebutuhan sistem meliputi kebutuhan perangkat lunak, kebutuhan perangkat keras, kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan ini digambarkan dalam bentuk tabel ataupun gambaran umum sistem.

4.1 Deskripsi Umum Sistem

Secara umum tujuan utama dari sistem yang dikembangkan adalah untuk pengimplementasian *gateway* sebagai penjembatan komunikasi antara *node* sensor dengan *server*. Komunikasi disini ialah pengiriman data dari *node* sensor menuju ke *server* dan pengiriman perintah dari *server* menuju ke *node* sensor. Penggambaran deskripsi umum sistem dapat dilihat pada Gambar 4.1.



Gambar 4.1 Deskripsi Umum Sistem

Seperti pada Gambar 4.1 Deskripsi Umum Sistem pada penelitian ini dibangun sebuah sistem sebagai lingkungan pengujian dari *gateway* yang dikembangkan. Sistem terdiri atas dari sebuah *node* sensor yang mengirimkan data melalui protokol RF24Mesh ke *gateway* untuk diteruskan. Selanjutnya *gateway* akan meneruskan data menuju ke *server* melalui protokol MQTT dengan melakukan *publish* kepada *broker* dan pada *server* akan melakukan *subscribe* pada *broker* untuk mendapatkan data. Setelah data diterima dari *broker server* akan menampilkan data tersebut. Hal yang sama terjadi pada pengiriman perintah dari *server* menuju ke *node* sensor.

4.2 Kebutuhan Sistem

Kebutuhan sistem bertujuan untuk mengetahui kebutuhan yang diperlukan dalam membangun sistem dan mengimplementasikan *gateway* sebagai perantara komunikasi antara *node* sensor dengan *server*. Kebutuhan ini dapat mempermudah proses perancangan dan implementasi. Kebutuhan terbagi menjadi dua yakni kebutuhan fungsional dan kebutuhan non-fungsional.

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan-kebutuhan yang harus dipenuhi dan proses-proses yang dapat dilakukan oleh sistem. Kebutuhan fungsional sistem dalam penelitian ini dapat dilihat pada Tabel 4.1 berikut ini :

Tabel 4.1 Kebutuhan Fungsional.

No.	Kebutuhan Fungsional
1	*Node* sensor dapat mengambil data sensor.
2	*Node* sensor dapat mengirim data sensor menuju *gateway*.
3	Perangkat *gateway* dapat menerima data sensor yang dikirimkan oleh *node* sensor.
4	Perangkat *gateway* dapat meneruskan data sensor menuju ke *server*.
5	*Server* dapat menerima data sensor dari perangkat *gateway*.
6	*Server* dapat menerima data *input* perintah dan mengirimkan data menuju perangkat *gateway*.
7	Perangkat *gateway* dapat menerima data dari *server*.
8	Perangkat *gateway* dapat meneruskan data menuju ke *node* sensor.
9	*Node* sensor dapat menerima data dari perangkat *gateway*.
10	*Node* sensor menjalankan aktuator sesuai data yang diterima dari perangkat *gateway*.

4.2.2 Kebutuhan Non-fungsional

Kebutuhan non-fungsional adalah kebutuhan yang menitiberatkan pada perilaku dan batasan fungsi pada sistem. Dalam penelitian ini kebutuhan non-fungsional terdiri dari kebutuhan perangkat keras dan kebutuhan perangkat lunak.

4.2.2.1 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang dibutuhkan dalam pengembangan *gateway* dan sistem dalam penelitian ini dapat di lihat pada Tabel 4.2 kebutuhan perangkat keras yaitu :

Tabel 4.2 Kebutuhan Perangkat Keras.

Perangkat Keras	Keterangan
Arduino Nano	Perangkat ini merupakan mikrokontroler yang akan digunakan sebagai *slave node.*
Raspberry Pi 3	Perangkat ini merupakan mikrokontroler yang akan digunakan sebagai *gateway* sekaligus *master node.*
Modul sensor *DHT 11*	Perangkat ini adalah modul sensor untuk mengambil data suhu dan tingkat kelembaban udara.
Modul *nRF24L01*	Perangkat ini adalah modul komunikasi yang dapat menggunakan protokol *RF24Mesh.*

	Perangkat ini juga yang digunakan untuk komunikasi antara *node* sensor dengan *gateway*.
Kabel jumper	Kabel yang dibutuhkan untuk menyabungkan modul pada perangkat mikrokontroler.
Light-emitting diode (LED)	Sebuah lampu berukuran kecil yang digunakan sebagai aktuator pada *node* sensor.
Laptop	Perangkat ini akan digunakan untuk menjalankan *host* dari *server* yang berupa mesin virtual.

4.2.2.2 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang dibutuhkan dalam pengembangan *gateway* dan sistem dalam penelitian ini dapat di lihat pada Tabel 4.3 kebutuhan perangkat lunak yaitu :

Tabel 4.3 Kebutuhan Perangkat Lunak.

Perangkat Lunak	Keterangan
Arduino IDE	Perangkat lunak ini merupakan sebuah *IDE* yang dipergunakan untuk memprogram mikrokontroler *Arduino Nano.*
PuTTy	Perangkat lunak ini diperlukan untuk mengakses mesin *Raspberry Pi 3* dan *server* secara *remote* dengan menggunakan *SSH.*
GNU nano	Perangkat lunak ini merupakan perangkat lunak *text editor* berbasis *command line interface* yang digunakan untuk menulis kode program pada perangkat Raspberry Pi dan *server*.
Raspbian Stretch	Perangkat lunak ini adalah sistem operasi berbasis unix yang ada untuk mengoperasikan *Raspberry Pi 3*.
Virtualbox	Perangkat lunak ini akan dipergunakan sebagai *host server*.
Ubuntu *Server* 18.04	Perangkat lunak ini akan digunakan sebagai sistem operasi pada *server*.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

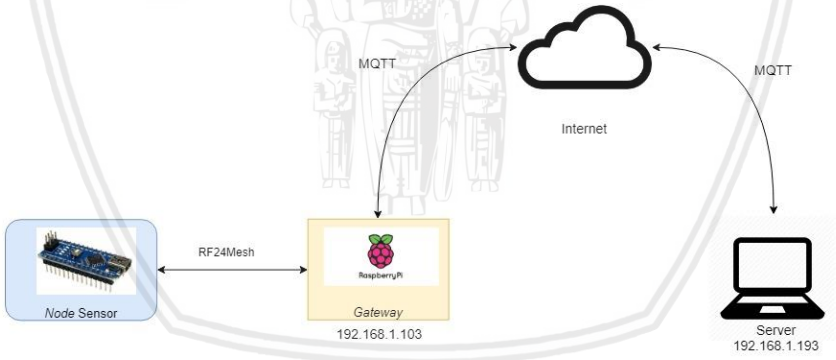
Perancangan dan implementasi merupakan tahap merancang dan mengimplementasikan sistem yang telah dideskripsikan kebutuhannya pada bab rekayasa kebutuhan.

5.1 Perancangan

Perancangan merupakan tahap dimana komponen komponen yang dibutuhkan sistem dirancang. Tahap ini ada agar tahap implementasi menjadi lebih mudah dan terarah. Tahap ini akan menentukan berjalan tidaknya tahap implementasi sistem. Perancangan memuat tentang perancangan topologi jaringan, perancangan data, perancangan protokol, perancangan *node* sensor, perancangan *gateway*, perancangan *server*, perancangan parameter pengujian dan perancangan skenario pengujian.

5.1.1 Perancangan Topologi Jaringan

Perancangan topologi jaringan akan menjelaskan tentang topologi yang digunakan pada sistem yang dibangun. Topologi jaringan ini juga dapat memberikan gambaran umum dari sistem yang dibuat. Untuk gambaran mengenai topologi jaringan dapat dilihat pada Gambar 5.1 perancangan topologi jaringan sistem.



Gambar 5.1 Perancangan Topologi Jaringan

Gambar 5.1 menujukkan topologi jaringan yang membangun sistem. Pada topologi jaringan ini terdapat tiga komponen perangkat keras yang ada pada sistem. Terdapat sebuah *node* sensor yang terhubung dengan *gateway* dengan protokol RF24Mesh yang menggunakan frekuensi radio untuk keperluan komunikasi. Perangkat *gateway* memiliki fungsi menjembatani komunikasi antara *node* sensor dan *server*. Perangkat *gateway* berkomunikasi dengan *server* melalui koneksi internet dengan menggunakan protokol MQTT yang memiliki model *publish-subscribe*. Alamat dari perangkat *gateway* telah dikonfigurasi agar statis pada 192.168.1.103. Sama halnya dengan *gateway*, pada sisi *server* untuk terhubung dengan *gateway* diperlukan koneksi internet dengan protokol yang

digunakan adalah MQTT. Alamat pada *server* juga telah diatur menjadi statis pada 192.168.1.193. Untuk komunikasi radio antara perangkat nRF24L01 pada node sensor dan *gateway* dikonfigurasikan dengan *node id* 1 untuk *node* sensor dan 0 untuk *gateway*.

5.1.2 Perancangan Data

Pada perancangan data akan menjelaskan tentang perancangan data yang dipergunakan pada pertukaran data antar komponen pada jaringan yang dibangun. Perancangan data akan dibagi berdasarkan komunikasi antar komponen pada sistem.

Pada komunikasi antara *node* sensor dengan *gateway* tipe data yang digunakan adalah *struct*. *Struct* berisi kumpulan variabel yang berada dalam satu nama yang sama dan memiliki kaitan satu sama lain. Isi dari data *struct* sendiri adalah variabel *integer* hum, temp, packet_number dan *node*_id . Variabel hum dan temp berisi nilai kelembaban dan suhu udara yang ditangkap pada perangkat sensor. Variabel packet number berisi nomor paket yang berhasil dikirim dari *node* sensor menuju ke *gateway*. *Node* id berisi nomor dari *node* pengirim, bernilai satu untuk *node* sensor dan bernilai nol untuk *gateway*. Untuk ukuran data pada pengiriman ini disesuaikan dengan *length* dari data yang dikirimkan pada *node* sensor.

Pada komunikasi antara *gateway* dengan *server* tipe data yang digunakan adalah json. Data json memiliki struktur yang tidak berbeda jauh dengan *struct* yaitu dengan menggunakan pasangan *key* dan *value*. Data json yang dikirimkan ke sever memiliki nilai yang sama dengan yang diterima oleh *gateway* dari *node* sensor yaitu hum, temp, packet_number dan *node*_id. Data yang dikirimkan bernilai sama karena *gateway* hanya meneruskan data menuju ke *server*, hal yang dilakukan oleh *gateway* hanya merubah bentuk tipe data *struct* menjadi json.

Pada komunikasi antara *server* menuju ke *node* sensor hanya memiliki perbedaan pada nilai data yang dikirimkan yaitu berupa nilai *integer* inputan pada sisi *server*. Tipe data yang digunakan untuk komunikasi antar komponen dari *server* menuju ke *node* sensor menggunakan tipe data yang sama yaitu json dan *struct*.

5.1.3 Perancangan Protokol

Perancangan protokol akan menjelaskan protokol yang digunakan pada komunikasi antara *node* sensor dengan *server* dan sebaliknya antara *server* dengan *node* sensor. Terdapat dua protokol yang digunakan pada sistem yang dibangun yaitu protokol RF24Mesh dan MQTT.

Protokol RF24Mesh adalah protokol yang diterapkan pada komunikasi antara *node* sensor dengan perangkat *gateway*. Protokol ini diterapkan dengan menggunakan bantuan *library* RF24Mesh yang telah ada. Hal yang perlu ditentukan oleh perangkat untuk menggunakan protokol ini adalah *node id* dan *message type*. Pada sistem yang dibangun *gateway* berperan sebagai *master node* memiliki *node id* 0. Sedangkan *node* sensor berperan sebagai *slave node* yang

dapat memiliki *node id* dalam kisaran 1-255. Ditetapkan pada *node* sensor menggunakan *node id* 1. *Message type* yang digunakan adalah 'M'. Untuk pengalamatan dan *routing* pada protokol akan diatur oleh *library* secara otomatis. Kode program untuk protokol RF24Mesh ditulis dengan menggunakan bahasa C++.

Protokol MQTT adalah protokol yang diterapkan pada komunikasi antara *gateway* dengan *server*. Perancangan protokol akan dibantu dengan menggunakan *library* yang sudah ada yaitu Paho MQTT. Protokol akan dirancang dengan menetapkan beberapa parameter seperti alamat *broker*, *port*, *username*, *password,* topik dan *quality of service* (qos). *Broker* yang digunakan pada sistem ini adalah *broker online* dengan alamat m15.cloudmqtt.com. *Port* yang digunakan adalah 13562. *Username* dan *password* yang digunakan adalah "dtmmbckv" dan "fe4kBI26no44". Topik yang digunakan oleh *gateway* untuk *publish* ke *broker* adalah "*node*/1" dan topik yang digunakan oleh *server* untuk *subscribe* ke *broker* adalah "*node*/#". Topik yang digunakan oleh *server* untuk *publish* ke *broker* adalah "cloud" dan topik yang digunakan oleh *gateway* untuk *subscribe* ke *broker* adalah "cloud". Qos yang digunakan adalah qos 1 yang memiliki kepastian sampainya data dengan kemungkinan terjadinya duplikasi data. Kode program untuk protokol MQTT ditulis dengan menggunakan bahasa C++.

5.1.4 Perancangan *Node* Sensor

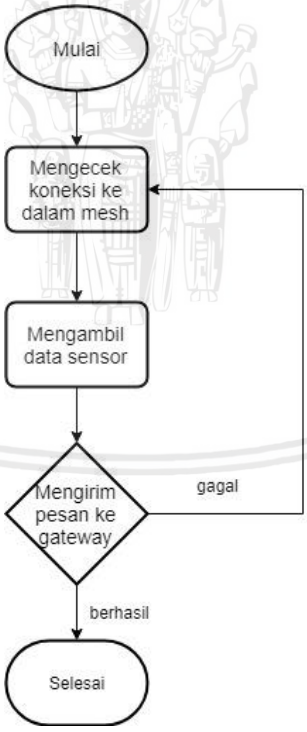
Pada perancangan *node* sensor akan menjelaskan hal hal yang perlu dilakukan untuk membuat komponen *node* sensor yang dapat berjalan sesuai dengan tujuan. *Node* sensor akan menggunakan mikrokontroler Arduino Nano. Pada *node* sensor dihubungkan dengan 2 buah modul yaitu modul komunikasi nRF24L01 dan sensor DHT11. Kedua modul ini dihubungkan dengan Arduino menggunakan kabel jumper *female-female* pada pin pin yang telah ada. Untuk deskripsi *pinout* pada *node* sensor dapat dilihat pada Tabel 5.1.

Tabel 5.1 Susunan Pinout *Node* Sensor

Arduino Nano	nRF24L01	Sensor DHT11	LED
GND	GND	GND	Cathode (-)
3.3V	VCC	-	-
5V	-	VCC	-
D13	SCK	-	-
D12	MISO	-	-
D11	MOSI	-	-
D8	CE	-	-
D7	CSN	-	-
D2	-	D0	-

D4	-	-	A*node* (+)

Tabel 5.1 merupakan tabel konfigurasi *pinout* untuk *node* sensor yang terdiri dari empat buah perangkat yaitu mikrokontroler Arduino Nano, modul komunikasi nRF24L01, sensor DHT dan LED. Arduino nano sendiri mendapatkan daya dengan menghubungkan kabel *usb mini* dengan sumber tegangan dari usb perangkat lain (contoh laptop/komputer) atau langsung dihubungkan dengan stopkontak menggunakan *power adapter*. Pada nRF24L01 tegangan positif (VCC) dihubungkan dengan pin Arduino 3,3V. Dan tegangan negatif (GND/ground) dihubungkan dengan sesama GND pada Arduino. Pada pin *Master In Slave Out* (MISO) dihubungkan dengan pin Digital 12 (D12). Pada pin *Master Out Slave in* (MOSI) terhubung dengan pin Digital 11 (D11). Pin *Serial Clock* (SCK) dihubungkan dengan pin Digital 13(D13). Maksud dari 3 pin tersebut adalah MISO (Master In Slave Out) & MOSI (Master Out Slave In) adalah jalur data untuk komunikasi antara *Master* (*programmer* / *downloader*) dan *Slave* (mikrokontroler). MISO merupakan jalur yang digunakan *download* untuk menerima dan MOSI adalah jalur *downloader* mengirim data. Kedua jalur ini adalah jalur yang digunakan *downloader* dan mikrokontroler untuk berkomunikasi. Untuk menghindari kesalahan dalam berkomunikasi memanfaatkan jalur SCK dalam sinkronisasi.

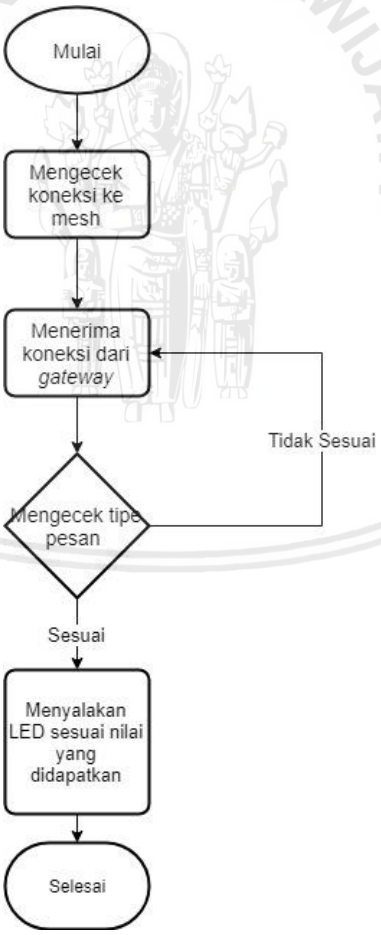


Gambar 5.2 Diagram Alir Pengiriman *Node* Sensor Menuju *Gateway*

Pada Sensor DHT11 VCC menggunakan 5V dan GND terhubung dengan sesama GND. Untuk *digital* output mengambil data menggunkan D2, D2 ini nantinya yang akan diprogram pada kode program. Pada *LED* tagangan negatif terhubung dengan GND dan tegangan positif terhubung dengan Digital 4 (D4). Pada kode program *input* yang akan dimainkan menggunakan pin D4. Pin katode (+) juga

perlu terhubung pada sebuah *resistor* sebelum terhubung pada pin D4 untuk mengatur tegangan. Jika dilihat pada Tabel 5.1 pada Arduino dibutuhkan 3 buah pin GND untuk 3 buah perangkat yang terhubung. Arduino nano sendiri hanya menyediakan 2 pin GND, untuk itu diperlukan sebuah *project board* jika ingin menghubungkan 3 perangkat secara langsung pada mikrokontroler Arduino nano. Untuk kode program karena menggunakan Arduino maka berbasis bahasa C++. Kode program dapat ditulis menggunakan Arduino IDE, perangkat lunak yang berguna untuk memprogram produk keluaran Arduino.

Alur kerja pada *node* sensor terbagi menjadi alur kerja pengiriman data dari *node* sensor menuju ke *gateway* dan alur kerja penerimaan data dari *gateway*. Pada Gambar 5.2 menjelaskan secara singkat alur kerja *node* sensor pada pengiriman data menuju ke *server*. Dimulai dengan melakukan pengecekan terhadap koneksi ke mesh lalu mengambil data sensor pada perangkat sensor dan mengirimkannya menuju *gateway* jika berhasil. Jika pengiriman gagal akan melakukan cek ulang pada koneksi dalam mesh. Pada Gambar 5.3 akan dijelaskan alur kerja *node* sensor ketika menerima data dari *gateway* dalam bentuk diagram alir.



Gambar 5.3 Diagram Alir Menerima Data *Input* Dari *Gateway*

Pada Gambar 5.3 menjelaskan secara singkat alur kerja *node* sensor ketika menerima data *input* yang dikirimkan dari *gateway*. Dimulai dengan melakukan

pengecekan koneksi ke *mesh*. Proses selanjutnya berjalan ketika terdapat data yang masuk pada nRF24L01 dari *gateway*. Akan dilihat apakah tipe pesan yang masuk sesuai. Jika tidak maka akan kembali menunggu koneksi yang masuk lagi dari *gateway*. Setelah itu akan menjalankan aktuator yaitu LED sesuai dengan nilai yang masuk, menyala ketika bernilai 1 dan mati ketika bernilai 0.

5.1.5 Perancangan *Gateway*

Pada penelitian ini perancangan *gateway* merupakan komponen yang menjadi fokus utama. *Gateway* disini akan bertugas untuk meneruskan data dari *node* sensor menuju ke *server* dengan kata lain penjembatan dari komponen satu ke komponen yang lain. Jika *gateway* tidak dapat berjalan sesuai tujuan maka data dari sensor tidak dapat dikirim begitu pula perintah yang masuk pada sisi *server* tidak dapat disampaikan ke *node* sensor. *Gateway* akan dirancang dengan menggunakan mikrokontroler Raspberry Pi 3. Pemilihan raspberry pi dikarenakan perlunya perangkat yang memiliki *adapter wireless* bawaan yang akan digunakan dalam pengkonversian protokol dari RF24Mesh menjadi MQTT. Pada perangkat *gateway* modul komunikasi yang digunakan adalah nRF24L01 yang menggunakan protokol RF24Mesh. Modul dan protokol ini akan digunakan pada komunikasi antara *gateway* dengan *node* sensor. Modul komunikasi nRF24L01 dihubungkan dengan perangkat *gateway* menggunakan kabel jumper *female-female.* Untuk komunikasi antara *gateway* dengan *server* menggunakan protokol MQTT dengan model *publish/subscribe*. Dimana data yang diperoleh *server* sebagai *subscriber* akan tergantung topik yang dibutuhkan oleh *server*. Untuk deskripsi lebih lanjut pada *pinout* yang ada pada perangkat *gateway* dapat dilihat pada Tabel 5.2.

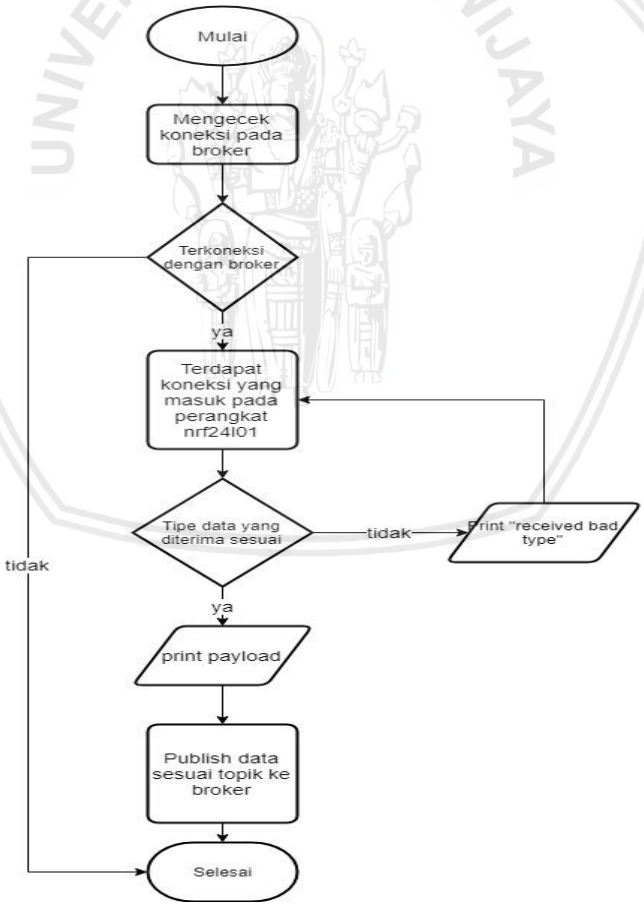
Tabel 5.2 Susunan *Pinout* Perangkat *Gateway*

Raspberry Pi 3	nRF24L01
GND	GND
3.3V	VCC
GPIO11	SCK
GPIO9	MISO
GPIO10	MOSI
GPIO22	CE
GPIO8	CSN

Seperti terlihat pada Tabel 5.2 terdapat 7 buah pin yang digunakan pada nRF24L01 untuk terhubung pada mikrokontroler raspberry pi. Pada raspberry pi, pin yang ada disebut sebagai *General Purpose Input Output* (GPIO). GPIO memiliki fungsi untuk mengalirkan data ataupun tegangan listrik di mana pin tersebut bisa digunakan sebagai media penyalur input atau output, input dan output-nya dapat berupa data atau tegangan listrik. Pada raspberry pi sumber daya dapat

dihubungkan dengan menggunakan kabel *usb* atau dengan *power adapter* yang langsung terhubung pada stopkontak sama halnya dengan Arduino. Pada modul nRF24L01 tegangan negatif (GND) terhubung dengan pin GND pada raspberry pi. Tegangan positif (VCC) juga terhubung pada tegangan VCC pada raspberry pi. Untuk SCK, MISO, MOSI digunakan sebagai jalur komunikasi antara *master* (*programmer* / *downloader*) dan *slave* (mikrokontroler). Raspbian Stretch digunakan sebagai sistem operasi untuk menjalan kan raspberry pi. Untuk kode program pada *gateway* menggunakan bahasa C++ yang dapat ditulis dengan GNU nano yang merupakan text editor bawaan pada sistem operasi unix. Penggunaan bahasa C++ juga untuk memudahkan menuliskan kode program karena pada beberapa *library* yang digunakan *gateway* untuk menulis kode program berbasis C++.

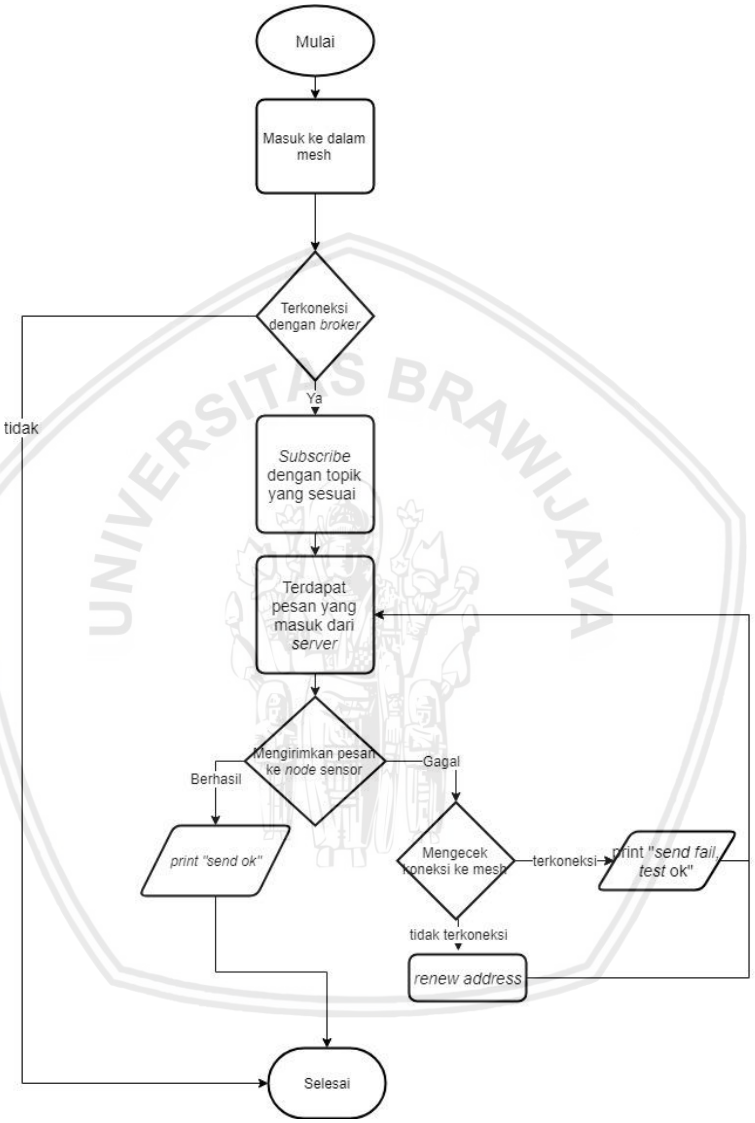
Alur kerja pada *gateway* terbagi menjadi alur kerja pengiriman data dari *node* sensor menuju ke *server* dan alur kerja pengiriman data dari *server* menuju ke *node* sensor. Pada Gambar 5.3 akan dijelaskan pengiriman data dari *node* sensor menuju ke *server* dalam bentuk diagram alir.



Gambar 5.4 Diagram Alir Pengiriman Data Sensor Menuju *Server*

Pada Gambar 5.4 menjelaskan secara singkat alur kerja *gateway* pada pengiriman data menuju ke *server*. Dimulai dengan melakukan pengecekan

terhadap koneksi ke *broker* jika tidak maka selesai dan jika iya proses akan dilanjutkan. Proses selanjutnya berjalan ketika terdapat data yang masuk pada nRF24L01 dari *node* sensor. Akan dilihat apakah tipe pesan yang masuk sesuai. Jika iya *payload* akan ditampilkan dan jika tida akan menampilkan pesan "*received bad type*". Setelah proses *print payload* maka data yang masuk akan dilakukan *publish* ke *broker* dengan topik yang sesuai.



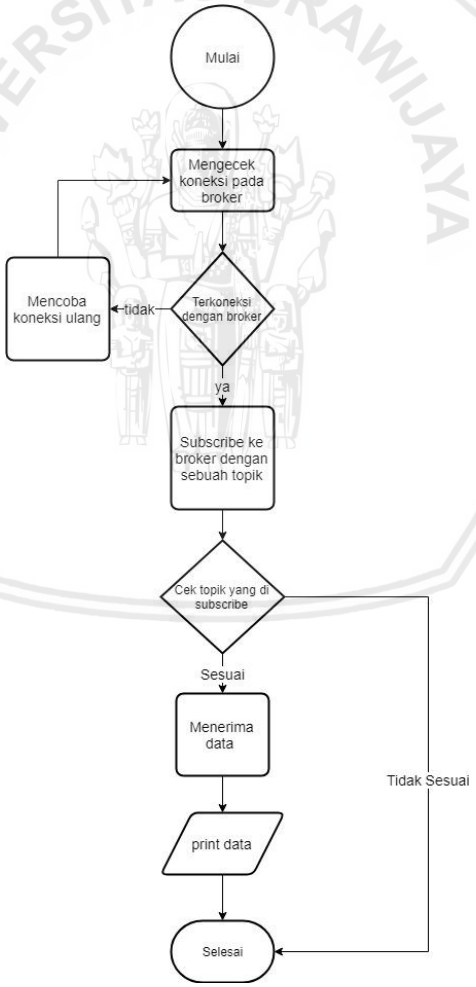
Gambar 5.5 Diagram Alir Pengiriman Data *Input* Menuju *Node* Sensor

Pada Gambar 5.5 menjelaskan secara singkat alur kerja dari *gateway* pada saat pengiriman data *input* menuju ke *node* sensor. Dimulai dengan masuk ke dalam jaringan *mesh* lalu melakukan pengecekan pada *broker* untuk melihat koneksi sudah terhubung atau tidak. Jika tidak maka proses akan selesai. Jika iya *gateway* akan melakukan *subscribe* pada *broker* dengan topik yang sesuai dan menunggu adanya pesan yang masuk dari *server*. Setelah terdapat pesan yang masuk dari *server* pesan akan dikirimkan menuju *node* sensor. Jika gagal akan melakukan cek koneksi ke dalam *mesh*. Jika terhubung ke dalam *mesh* akan menampilkan pesan "*send fail, test ok*" dan jika tidak akan mencoba *renewing address* ke dalam *mesh*.

Jika pesan berhasil dikirim akan menampilkan pesan "*send ok*" dan proses akan selesai.

5.1.6 Perancangan *Server*

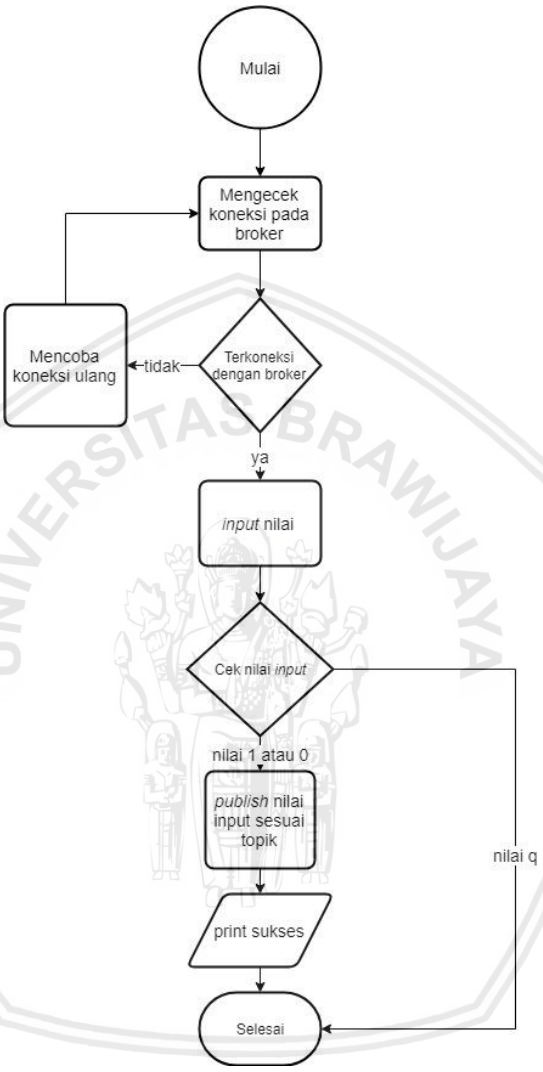
Perancangan *server* akan membahas tahapan untuk merancang *server* pada sistem yang dibangun. *Server* yang digunakan pada penelitian ini berbentuk *virtual machine* (VM) yang akan dibuat dengan aplikasi Virtualbox. Untuk membangun perangkat *server* dibutuhkan sebuah laptop dengan RAM sebesar 1Gb untuk menjalankan *server*. *Server* menggunakan sistem operasi Ubuntu *Server* 18.04 untuk menjalankan seluruh proses yang dibutuhkan pada *server*. Pada *server* penulisan kode program menggunakan bahasa C++. *Network adapter* pada VM diatur menggunakan *bridge* untuk terhubung dengan internet dan *host only* untuk dapat terhubung dengan *host* secara lokal. Alur kerja pada *server* terbagi menjadi alur kerja penerimaan data dari *gateway* dan alur kerja pada pengiriman data dari *server* menuju ke *gateway*. Pada gambar 5.6 akan dijelaskan alur kerja penerimaan data dari *gateway* dalam bentuk diagram alir.



Gambar 5.6 Diagram Alir Menerima Data Sensor Pada *Server*

Pada Gambar 5.6 menjelaskan secara singkat alur kerja dari *server* pada saat menerima data dari *gateway*. Dimulai dengan melakukan pengecekan pada *broker*

untuk melihat koneksi sudah terhubung atau tidak. Jika tidak maka mencoba melakukan koneksi ulang ke *broker*. Jika iya *server* akan melakukan *subscribe* pada *broker* dengan sebuah topik tertentu. Jika topik tidak sesuai makan proses selesai dan *server* tidak mendapatkan data. Jika topik yang di*subscribe* sesuai maka *server* akan menerima data sensor lalu menampilkannya pada layar monitor *server*.



Gambar 5.7 Diagram Alir Mengirim Data Input Ke *Gateway*

Pada Gambar 5.7 menjelaskan secara singkat alur kerja dari *server* pada saat pengiriman data menuju ke *gateway*. Dimulai dengan melakukan pengecekan pada *broker* untuk melihat koneksi sudah terhubung atau tidak. Jika tidak maka mencoba melakukan koneksi ulang ke *broker*. Jika iya *server* akan menunggu *input* nilai. Setelah terdapat *input* nilai *server* akan melakukan pengecekan terhadap nilai, jika input bernilai 'q' maka proses akan selesai. Jika *input* memiliki nilai '1' atau '0' maka *server* akan melakukan *publish* dengan topik "cloud" dan konten berupa nilai *input*. Setelah berhasil melakukan *publish* akan menampilkan pesan sukses dan proses selesai.

5.1.7 Perancangan Parameter Pengujian

Perancangan parameter pengujian dilakukan untuk menentukan parameter yang diperlukan dalam pengujian sistem. Parameter ini diperlukan sebagai tolak ukur tercapainya kebutuhan ataupun performa dari sistem yang dibangun. Pada kebutuhan fungsional sistem, parameter ini berupa hasil yang diharapkan pada pengujian dengan skenario yang telah dibuat. Pada pengujian performa parameter ini berupa tingkat keberhasilan (*successfull rate*) dari pengiriman yang berhasil dilakukan oleh *gateway* pada pengiriman data dari *node* sensor menuju ke *server*. Parameter pengujian performa juga akan diuji terhadap tiga parameter lain yaitu jeda waktu, jarak dan ukuruan paket. Pemilihan jarak dilakukan dengan melihat penelitian terdahulu (Pratama, Akbar, & Bhawiyuga, 2017) yang menggunakan jarak maksimal 30 meter. Untuk pemilihan parameter ukuran paket mengacu pada dokumentasi (tmrh20, 2015) yang menyebutkan *default* data maksimum yang dapat dikirimkan oleh nRF24L01 adalah 120 *byte*. Pemilihan jeda waktu dilakukan untuk melihat performansi dari *gateway* jika terdapat perbedaan waktu pengiriman pada setiap paketnya.

5.1.8 Perancangan Skenario Pengujian

Perancangan skenario pengujian diperlukan agar pengujian yang dilakukan lebih sistematis. Perancangan pengujian berisi perancangan pengujian *fungsional* dan *non-fungsional*. Pada pengujian *fungsional* akan dibuat sebuah skenario untuk setiap kebutuhan fungsional yang telah dituliskan pada bab rekayasa kebutuhan. Untuk pengujian *non-fungsional* akan dibuat skenario untuk melihat performa atau keandalan dari sistem yang dibangun.

5.1.8.1 Pengujian *Fungsional*

Perancangan pengujian fungsional sistem dilakukan berdasarkan kebutuhan sistem yang telah dijelaskan pada bab rekayasa kebutuhan. Pengujian fungsional sistem membahas 3 komponen yang berkerja yaitu *node* sensor, *gateway* dan *server*. Semua pengujian dilakukan dengan kondisi tidak adanya penghalang antara *node* sensor dan *gateway*. Lokasi pengujian fungsional ditentukan oleh peneliti dengan memperhatikan latar belakang yang telah ditulis. Lokasi pengujian berada di lahan pertanian yang terbuka dan tidak terdapat penghalang diantara *node* sensor dengan *gateway*. Pengujian fungsional dinyatakan valid jika hasil skenario yang diharapkan dan hasil pengujian bernilai sama. Detail pengujian fungsional dapat dilihat pada tabel 5.3.

Tabel 5.3 Perancangan Skenario Pengujian Fungsional

No	Kode	Fungsi	Skenario Pengujian
1	FR-01	*Node* sensor dapat mengambil data sensor.	1. *Node* sensor keadaan menyala 2. Perangkat sensor terhubung dengan *node* sensor

			3. Mengambil data sensor menggunakan perangkat sensor 4. Menampilkan hasil data sensor yang ditangkap.
2	FR-02	*Node* sensor dapat mengirim data sensor menuju *gateway*.	1. *Node* sensor keadaan menyala 2. Modul nRF24L01 keadaan menyala 3. *Node* sensor telah mengambil data dari perangkat sensor 4. *Node* sensor mengirim data menuju perangkat *gateway*.
3	FR-03	Perangkat *gateway* dapat menerima data sensor yang dikirimkan oleh *node* sensor.	1. *Gateway* dalam keadaan menyala 2. Modul nRF24L01 dalam keadaanya menyala 3. *Gateway* menerima data dari *node* sensor.
4	FR-04	Perangkat *gateway* dapat meneruskan data sensor menuju ke *server*.	1. *Gateway* dalam keadaan menyala 2. Modul nRF24L01 dalam keadaan menyala 3. *Gateway* telah menerima data dari *node* sensor 4. *Gateway* meneruskan data menuju *broker* dengan *publish* topik 5. *Broker* meneruskan ke *server* jika telah melakukan *subscribe* ke topik yang sesuai.
5	FR-05	*Server* dapat menerima data sensor dari perangkat *gateway*.	1. *Server* keadaan menyala 2. *Server* mendapatkan data dengan melakukan *subscribe* sesuai topik 3. *Server* menampilkan data yang didapatkan.
6	FR-06	*Server* dapat menerima data *input* perintah dan mengirimkan data menuju perangkat *gateway*.	1. *Server* keadaan menyala 2. *Server* mendapat nilai *input* 3. *Gateway* keadaan menyala 4. *Server* mengirimkan data *input* dengan *publish* ke *broker*

			5. *Broker* meneruskan data ke *gateway* jika telah melakukan *subscribe* topik yang sesuai.
7	FR-07	Perangkat *gateway* dapat menerima data dari *server*.	1. Perangkat *gateway* dalam keadaan menyala 2. Perangkat *gateway* telah melakukan *subscribe* topik yang sesuai 3. *Gateway* menampilkan data input dari *broker.*
8	FR-08	Perangkat *gateway* dapat meneruskan data menuju ke *node* sensor.	1. Perangkat *gateway* dalam keadaan menyala 2. Modul komunikasi nRF24L01 dalam keadaan menyala 3. *Node* sensor dalam keadaan menyala 4. *Gateway* mengirimkan data menuju *node* sensor.
9	FR-09	*Node* sensor dapat menerima data dari perangkat *gateway*.	1. *Node* sensor dalam keadaan menyala 2. Modul komunikasi nRF24L01 dalam keadaan menyala 3. *Node* sensor menampilkan data *input* dari *gateway*.
10	FR-10	*Node* sensor menjalankan aktuator sesuai data yang diterima dari perangkat *gateway*.	1. LED telah terpasang dalam kondisi mati 2. *Node* sensor telah menerima data dari *gateway* 3. LED menyala sesuai dengan nilai input yang didapatkan.

5.1.8.2 Pengujian Kinerja

Parameter yang di uji pada pengujian performa ini adalah *successfull rate*. Parameter ini akan menguji tingkat keberhasilan dari *gateway* dalam meneruskan data dari *node* sensor menuju ke *server*. Lokasi pengujian berada di lahan pertanian yang memiliki panjang atau lebar minimal 30 meter yang terbuka dan tidak terdapat penghalang diantara *node* sensor dengan *gateway*. Parameter *successfull rate* akan diuji dengan 3 parameter yaitu jarak, jeda waktu dan ukuran

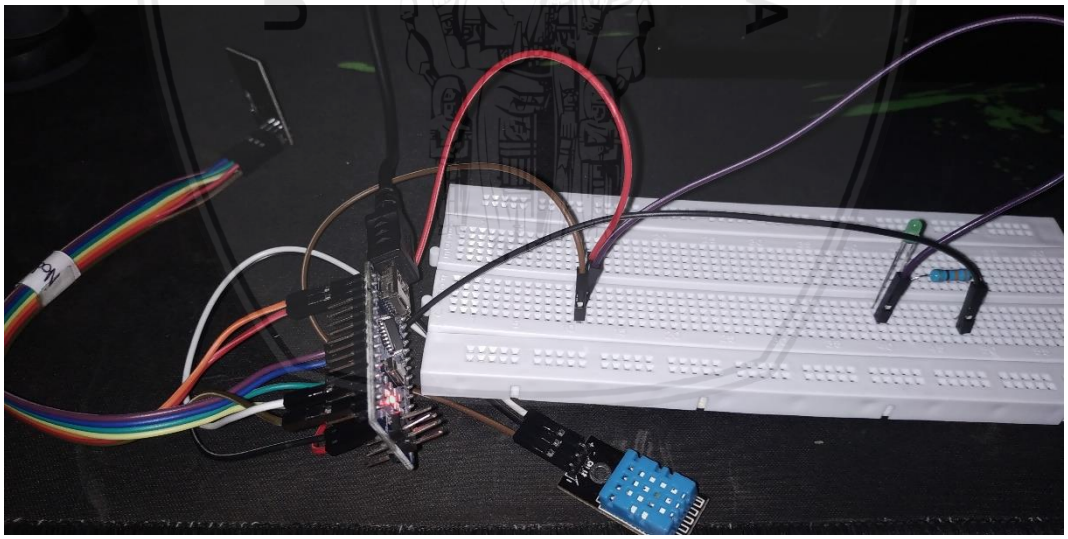
paket. Pembagian skenario pengujian performa dibagi menjadi tiga yaitu pengujian performa dengan jeda waktu 0,5 detik, 1 detik dan 5 detik. Pada setiap skenario jeda waktu terdapat variasi jarak yang akan diuji yaitu 20 meter, 25 meter dan 30 meter. Variasi ukuran paket yang akan diuji yaitu 50 *Byte*, 100 *Byte*, 150 *Byte*.

5.2 Implementasi

Berangkat pada perancangan yang telah dilakukan, langkah berikutnya adalah melakukan implementasi. Dimulai dari implementasi *node* sensor lalu selanjutnya implementasi *gateway*.

5.2.1 Implementasi *Node* Sensor

Implementasi *node* sensor dalam sistem ini menggunakan empat buah perangkat keras yaitu mikrokomputer Arduino Nano, LED, sensor dht11 dan modul *nRF24L01* sebagai modul komunikasi dengan *gateway.* Perangkat keras sensor, dan modul *nRF24L01* yang telah disiapkan, di hubungkan dengan kabel *jumper* sesuai dengan pinout yang telah dijelaskan pada bab perancangan. Setelah perangkat keras saling terhubung selanjutnya memberi daya pada Arduino Nano agar dapat diaktifkan. Arduino Nano dapat disambungkan ke laptop untuk mendapatkan sumber daya listrik dengan kabel *mini* usb. Gambar *node* sensor dapat dilihat pada Gambar 5.8 di bawah ini.



Gambar 5.8 *Node* Sensor

Setelah perangkat penyusun *node* sensor berhasil diimplemetasikan. Selanjutnya adalah menambahkan *library* yang dibutuhkan untuk memudahkan menjalankan kode program.

Tabel 5.4 Library Pada *Node* Sensor

```
1  #include "RF24.h"
2  #include "RF24Network.h"
3  #include "RF24Mesh.h"
4  #include <Adafruit_Sensor.h>
```

```
5  #include <DHT.h>
6  #include <DHT_U.h>
```

Include library RF24.h, RF24Network.h, RF24Mesh.h digunakan untuk memudahkan menjalankaan fungsi perangkat modul nRF24L01 sebagai media komunikasi. Selanjutnya *library* Adafruit_Sensor.h, DHT.h, DHT U.h digunakan untuk memudahkan menjalankan fungsi perangkat dht11. Sesudah menyiapkan *library* yang dibutuhkan, pembuatan utama kode program dapat dilakukan. Kode program dituliskan menggunakan bahasa pemrograman C++.

Kode Program 1 : Mengambil Nilai Sensor

```
1  #define DHTPIN 2
2  #define DHTTYPE DHT11
3  DHT_Unified dht(DHTPIN, DHTTYPE);
4
5  payload_t getDHT(payload_t dat) {
6    sensors_event_t event;
7    dht.temperature().getEvent(&event);
8    if (isnan(event.temperature)) {
9      Serial.println("Error reading temperature!");
10   }
11   else {
12     Serial.println("Data sensor receive :  ");
13     Serial.print("Temperature: ");
14     Serial.print(event.temperature);
15     Serial.println(" *C");
16     dat.temp = event.temperature;
17   }
18   // Get humidity event and print its value.
19   dht.humidity().getEvent(&event);
20   if (isnan(event.relative_humidity)) {
21     Serial.println("Error reading humidity!");
22   }
23   else {
24     Serial.print("Humidity: ");
25     Serial.print(event.relative_humidity);
26     Serial.println("%");
27     dat.hum2 = event.relative_humidity;
28     dat.node_id = mesh.getNodeID();
29     Serial.print("Node Id: ");
30     Serial.println(dat.node_id);
31   }
32   Serial.print("Packet Number : ");
33   Serial.println(dat.packet_number);
34   return dat;
35 }
```

Pada Kode Program 1 menampilkan kode program yang digunakan untuk mengambil nilai data sensor yang telah terpasang pada *node* sensor dengan menggunakan fungsi *getEvent()* pada sensor dht11. Untuk setiap masing masing variabel yaitu suhu pada baris 7 -17 dan suhu udara memanggil fungsi *getEvent()* pada baris 18-30. Sebelumnya juga perlu dikonfigurasikan pin dari sensor dht11 yang digunakan pada baris 1-3. Nilai yang didapatkan akan disimpan pada *struct* dengan nama payload_t.

Kode Program 2 : Konfigurasi *Node* Sensor

```
1  #define nodeID 1
```

```
2   const int LED = 4;
3   RF24 radio(7, 8);
4   RF24Network network(radio);
5   RF24Mesh mesh(radio, network);
6
7   void setup() {
8     pinMode(LED, OUTPUT);
9     c = 1;
10    dht.begin();
11    Serial.begin(115200);
12    // Set the nodeID manually
13    mesh.setNodeID(nodeID);
14    // Connect to the mesh
15    Serial.println(F("Connecting to the mesh..."));
16    mesh.begin();
    }
```

Pada Kode Program 2 menampilkan konfigurasi yang digunakan untuk LED pada baris 8 dan nRF24L01 pada *node* sensor pada baris 1-5. Selain itu juga terdapat *method* setup yang berisi fungsi untuk mulai menjalankan sensor dht11 pada baris 10 dan *mesh* pada baris 16.

Kode Program 3 : Mengirim dan Menerima Data

```
1   void loop() {
2    if (c<=100){
3
4     mesh.update();
5     // Send to the master node every second
6     if (millis() - displayTimer >= jeda) {
7       displayTimer = millis();
8       payload_t dat;
9       //dat.ms = displayTimer;
10      dat.packet_number = c;
11      dat = getDHT(dat);
12      c++;
13      Serial.print("counter packet: ");
14      Serial.println(c-1);
15      // Send an 'M' type message containing the current
16  millis()
17      if (!mesh.write(&dat, 'M', ukuran)) {
18
19        // If a write fails, check connectivity to the mesh
20  network
21        if ( ! mesh.checkConnection() ) {
22          //refresh the network address
23          Serial.println("Renewing Address");
24          Serial.println();
25          mesh.renewAddress();
26        } else {
27          Serial.println("Send fail, Test OK\n");
28        }
29      } else {
30        Serial.print("Send OK: \n");
31        Serial.println();
32      }
33    }
34  while (network.available()) {
35      RF24NetworkHeader header;
36      network.peek(header);
```

```
37      int payload;
38      switch(header.type){
39        case 'M': network.read(header, &payload,
40  sizeof(payload));
41
42  Serial.println("=========================================
43  =INCOMING PACKET============================");
44              Serial.print("Received packet : ");
45              Serial.print(payload);
46              Serial.print(" from node id : ");
47              Serial.println(header.from_node);
48              Serial.print("header id : ");
49              Serial.println(header.id);
50
51              if(payload == 1){
52                digitalWrite(LED, HIGH);
53                // turn the LED on (HIGH is the voltage
54  level)
55              }
56              else if (payload == 0){
57                digitalWrite(LED, LOW);     // turn the LED
58  off by making the voltage LOW
59              }
60              break;
61         default: network.read(header,0,0);
62              Serial.println(header.type);
63              break;
64      }
65    }
66   }
67   else {}
68  }
```

Kode Program 3 menampilkan kode program yang digunakan pada *node* sensor untuk mengirim dan menerima data. Untuk pengiriman data pada baris 6-33, setiap jeda waktu yang ditentukan *node* sensor akan menangkap nilai dari kelembaban dan suhu udara. Data yang ditangkap akan dimasukkan pada *struct*. Setelah berhasil mengambil nilai data sensor data akan dikirimkan dengan menggunakan fungsi *mesh.write*() pada baris 17 sesuai dengan tipe pesan dan ukuran dari *payload* yang diinginkan. Kode program untuk menerima data pada baris 34-68, setiap terdapat koneksi yang masuk akan ditangkap dengan menggunakan fungsi *network.available()* pada baris 34. Setelah terdapat data yang masuk akan dilakukan pengecekan tipe pesan pada baris 38-40 dan jika benar aktuator akan berjalan sesuai dengan isi dari pesan yang diterima seperti pada baris 51-59.

5.2.2 Implementasi *Gateway*

Implementasi perangkat *gateway* dalam sistem ini menggunakan dua buah perangkat keras yaitu mikrokomputer Raspberry Pi, dan modul *nRF24L01* sebagai modul komunikasi dengan *node* sensor. Modul *nRF24L01* yang telah disiapkan, di hubungkan dengan kabel *jumper* sesuai dengan pinout yang telah dijelaskan pada bab perancangan. Setelah perangkat keras saling terhubung selanjutnya memberi daya pada Raspberry Pi agar dapat diaktifkan. Raspberry Pi dapat disambungkan

ke laptop untuk mendapatkan sumber daya listrik dengan kabel *micro* usb. Kode program yang dituliskan pada perangkat *gateway* menggunakan bahasa pemrograman C++. Gambar perangkat *gateway* dapat dilihat pada Gambar 5.9 di bawah ini.

Gambar 5.9 Perangkat *Gateway*

Setelah perangkat penyusun *gateway* berhasil diimplemetasikan. Selanjutnya adalah menambahkan *library* yang dibutuhkan untuk memudahkan menjalankan kode program.

Tabel 5.5 *Library* Pada *Gateway*

```
1  #include "RF24Mesh/RF24Mesh.h"
2  #include <RF24/RF24.h>
3  #include <RF24Network/RF24Network.h>
4  #include <nlohmann/json.hpp>
5  #include <iostream>
6  #include "MQTTClient.h"
7  #include <thread>
8  #include <chrono>
9  #include "mqtt/async_client.h"
```

Library RF24.h, RF24Network.h, RF24Mesh.h digunakan untuk memudahkan menjalankaan fungsi perangkat modul nRF24L01 sebagai media komunikasi. *Library* MQTTClient.h dan async_client.h digunakan untuk memudahkan menjalankan fungsi untuk pengiriman data dengan protokol mqtt metode *publish-subscribe*. *Library* json.hpp digunakan untuk memudahkan pembuatan data dalam bentuk json. Sesudah menyiapkan *library* yang dibutuhkan, pembuatan utama kode program dapat dilakukan.

```
Kode Program 4 : Menerima Data
1  class callback : public virtual mqtt::callback
2  {
3  public:
4        void connection_lost(const string& cause) override {
5              cout << "\nConnection lost" << endl;
```

```
6               if (!cause.empty())
7                       cout << "\tcause: " << cause << endl;
8       }
9
10      void delivery_complete(mqtt::delivery_token_ptr tok)
11  override {
12              cout << "\tDelivery complete "  << endl;
13      }
14
15      void message_arrived(mqtt::const_message_ptr msg)
16  override {
17                  auto data = msg->get_payload();
18                  json data2 = json::parse(data);
19                  int input = data2.value("input", -1);
20                  payload_c payload2;
21                  std::cout << "\n==========INCOMING
22  PACKET==========" << std::endl;
23                  std::cout << "Message arrived " << std::endl;
24                  std::cout << "payload receive : " << input <<
25  std::endl;
26                  if(input == 1){
27                      payload2.pesan = 1;
28                  } else if (input == 0){
29                      payload2.pesan = 0;
30                  }
31
32                  uint16_t addr = mesh.getAddress(1);
33                  cout << "Sending data to Node 1 with address
34  "  << addr << "\n";
35                  if(!mesh.write(&payload2, 'M',
36  sizeof(payload2), sensor_node)){
37                      if(!mesh.checkConnection() ){
38                      printf("Renewing Address\n");
39                      mesh.renewAddress();
40                      }else{
41                      printf("Send fail, Test OK\n");
42                      }
43                  } else {
44                    printf("Send OK \n");
45                  }
46                    //Sending an message
47              //}
48      }
49  };
50
```

Pada Kode Program 4 menampilkan kode program yang digunakan untuk menerima data. Dilakukan pemanggilan kelas *callback* pada baris 1 untuk dapat menggunakan beberapa *method*. Pada baris 4-8 *method connection_lost* dipanggil ketika koneksi menuju broker terputus. Pada baris 11-14 *method delivery_complete* dipanggil jika pesan telah berhasil dikirim. Pada baris 16-50 *method message_arrived* dipanggil jika terdapat pesan yang masuk dari *server* lalu pada baris 36-46 pesan yang diterima akan dikirim menggunakan fungsi *mesh.write()* menuju ke *node* sensor.

Kode Program 5 : Mengirim Data

```
1   int main(int argc, char** argv) {
2
```

```
3     // Set the nodeID to 0 for the master node
4     mesh.setNodeID(master_node);
5
6     string addressMQTT  = (argc > 1) ? string(argv[1]) :
7   DFLT_SERVER_ADDRESS,
8           clientID     = (argc > 2) ? string(argv[2]) :
9   DFLT_CLIENT_ID;
10
11    c = 0;
12    printf("start\n");
13    mqtt::async_client client(addressMQTT, clientID);
14    // connect to mesh
15    mesh.begin();
16
17    mqtt::connect_options connOpts;
18    connOpts.set_keep_alive_interval(60);
19    connOpts.set_user_name(user);
20    connOpts.set_password(pass);
21    connOpts.set_clean_session(true);
22
23    callback cb;
24    client.set_callback(cb);
25
26    radio.printDetails();
27    printf("\n");
28    cout << "Initializing for server '" << addressMQTT <<
29  "'..." << endl;
30    cout << "Connecting..." << endl;
31    mqtt::token_ptr conntok = client.connect(connOpts);
32    cout << "Waiting for the connection..." << endl;
33    conntok->wait();
34    cout << "  ...OK" << endl;
35    client.subscribe(TOPIC2,QOS);
36
37    mqtt::delivery_token_ptr pubtok;
38    mqtt::message_ptr pubmsg;
39    printf("\n");
40
41  while(1)
42  {
43    // Call network.update as usual to keep the network
44  updated
45    mesh.update();
46    // In addition, keep the 'DHCP service' running on the
47  master node so addresses will be assigned to the sensor
48  nodes
49    mesh.DHCP();
50    // Check for incoming data from the sensors
51    while(network.available()){
52      RF24NetworkHeader header;
53      network.peek(header);
54      payload_t dat;
55      json j;
56
57      switch(header.type){
58        // Display the incoming millis() values from the
59  sensor nodes
60        case 'M': network.read(header,&dat,sizeof(dat));
61              c++;
```

```
62              j["temp"] = dat.temp;
63          j["hum"] = dat.hum2;
64              j["packet_number"] = dat.packet_number;
65              j["node_id"] = dat.node_id;
66
67              cout << "\nReceive packet from sensor node.
68  Details packets : \n"
69              << "Temperature : " << dat.temp << "\n"
70                  << "Kelembaban : " << dat.hum2 << "\n"
71              << "Packet number : " << dat.packet_number  <<
72  "\n"
73                  << "Counter : "<< c << "\n"
74              << "From node id : " << dat.node_id << "\n";
75
76             cout << "\nSending message to server ... " <<
77  endl;
78             pubmsg = mqtt::make_message(TOPIC, j.dump());
79             pubmsg->set_qos(QOS);
80             client.publish(pubmsg)->wait_for(TIMEOUT);
81             cout << "  ...OK" << endl ;
82             break;
83       default:  network.read(header,0,0);
84                 printf("Rcv bad type %d from
85  0%o\n",header.type,header.from_node);
86                 break;
87     }
88   }
89  delay(2);
90   }
91  return 0;
92  }
```

Pada Kode Program 5 menampilkan kode program yang digunakan untuk mengirimkan data ke *server*. Seperti yang terlihat pada baris 1 kode program ini masuk pada kelas *method main*. Pada baris 4 menentukan nomer *node* dari perangkat *gateway*. Pada baris 13 menginisialisasi *client* dengan parameter alamat dari broker dan ID dari *gateway*. Kode pada baris 15 untuk mulai masuk ke dalam *mesh* dengan fungsi *mesh.begin().* Pada baris ke 35 melakukan *subscribe* ke *broker* dengan topik dan qos yang telah ditentukan yaitu topik "*cloud*" dan qos 1. *Subscribe* dilakukan agar dapat menerima data dari *server*. Pada baris 51 melakukan cek jika terdapat koneksi yang masuk dari *node* sensor melalui perangkat nRF24L01. Baris 57-60 melakukan pengecekan tipe dari pesan yang masuk. Jika tipe pesan yang masuk bernilai "M" maka data akan dirubah kedalam tipe data *json* seperti pada baris 62-65. Pada baris 78-79 pembuatan pesan dengan menggunakan fungsi *make_message* dengan parameter topik dan qos yaitu topik "sensor/1" dan qos 1. Untuk pengiriman menuju ke *server* menggunkan fungsi *publish* dari *client* mqtt yang telah dibuat seperti pada baris 80.

5.2.3 Implementasi *Server*

Implementasi *server* menjelaskan hal apa saja yang perlu dilakukan untuk menjalankan perangkat *server*. Perangkat *server* menggunakan sebuah laptop sebagai *host*. Pada laptop dilakukan instalasi dan konfigurasi pada *virtualbox* untuk membuat sebuah *virtual machine* yang akan diisi dengan sistem operasi Ubuntu

Server 18.04. Nantinya sistem operasi ini yang akan dipergunakan untuk menjalankan *server*.

Tabel 5.6 *Library* Pada *Server*

```
1  #include "RF24Mesh/RF24Mesh.h"
2  #include <RF24/RF24.h>
3  #include <RF24Network/RF24Network.h>
4  #include <nlohmann/json.hpp>
5  #include <iostream>
6  #include "MQTTClient.h"
7  #include <thread>
8  #include <chrono>
9  #include "mqtt/async_client.h"
```

Pada tabel 5.8 menunjukkan *library* yang digunakan pada *server*. *Library* MQTTClient.h dan async_client.h digunakan untuk memudahkan menjalankan fungsi untuk pengiriman data dengan protokol mqtt metode *publish-subscribe*. *Library* json.hpp digunakan untuk memudahkan pembuatan data dalam bentuk json. Sesudah menyiapkan *library* yang dibutuhkan, pembuatan utama kode program dapat dilakukan. Kode program ditulis dengan menggunakan bahasa pemrograman C++.

Kode Program 6 : Menerima Data

```
1  class callback : public virtual mqtt::callback,
2                                  public virtual
3  mqtt::iaction_listener
4
5  {
6        // Counter for the number of connection retries
7        int nretry_;
8        mqtt::async_client& cli_;
9        mqtt::connect_options& connOpts_;
10       action_listener subListener_;
11
12       void reconnect() {
13
14       std::this_thread::sleep_for(std::chrono::millisecond
15 s(2500));
16             try {
17                   cli_.connect(connOpts_, nullptr, *this);
18             }
19             catch (const mqtt::exception& exc) {
20                   std::cerr << "Error: " << exc.what() <<
21 std::endl;
22                   exit(1);
23             }
24       }
25
26       // Re-connection failure
27       void on_failure(const mqtt::token& tok) override {
28             std::cout << "Connection failed" << std::endl;
29             if (++nretry_ > N_RETRY_ATTEMPTS)
30                   exit(1);
31             reconnect();
32       }
33
34       // Re-connection success
```

```
35      void on_success(const mqtt::token& tok) override {
36            std::cout << "\nConnection success" <<
37  std::endl;
38            std::cout << "\nSubscribing to topic '" <<
39  TOPIC << "'\n"
40                  << "\tfor client " << CLIENT_ID
41                  << " using QoS" << QOS << "\n"
42                  << "\nPress Q<Enter> to quit\n" <<
43  std::endl
44                  << "Press 1<Enter> to turn on the light
45  in sensor node\n" << std::endl
46                  << "Press 0<Enter> to turn off the light
47  in sensor node\n" << std::endl;
48
49            cli_.subscribe(TOPIC, QOS, nullptr,
50  subListener_);
51      }
52      // This will initiate the attempt to manually
53  reconnect.
54      void connection_lost(const std::string& cause)
55  override {
56            std::cout << "\nConnection lost" << std::endl;
57            if (!cause.empty())
58                  std::cout << "\tcause: " << cause <<
59  std::endl;
60
61            std::cout << "Reconnecting..." << std::endl;
62            nretry_ = 0;
63            reconnect();
64      }
65
66      // Callback for when a message arrives.
67      void message_arrived(mqtt::const_message_ptr msg)
68  override {
69            counter++;
70            std::cout << "Message arrived" << std::endl;
71            std::cout << "\ttopic: '" << msg->get_topic()
72  << "'" << std::endl;
73            auto data = msg->get_payload();
74            json data2 = json::parse(data);
75            packet_number = data2.value("packet_number", -
76  1);
77            temp = data2.value("temp", -1);
78            hum = data2.value("hum", -1);
79            node_id = data2.value("node_id", -1);
80
81            std::cout << "\tpayload : '{\"counter\": " <<
82  counter  << ", \"packet_number\": " << packet_number  <<"
83  \"temperature \" : " << temp
84                                          << ", \"node_id\":
85  "<< node_id << ", \"humidity\": "<< hum
86                                    << ", \"jeda\": "<< jeda <<
87  ", \"ukuran \": " << ukuran  << ", \"jarak\": " << jarak
88  << "}'.\n";
89
90            std::ofstream myfile;
91            myfile.open (file_name, std::ios::out |
92  std::ios::app);
93
```

```
94              myfile << "payload : '{\"counter\": " <<
95  counter  << ", \"packet_number\": " << packet_number  << "
96  \"temperature \" : " << temp
97                                                  << ", \"node_id\":
98  "<< node_id << ", \"humidity\": "<< hum
99                                  << ", \"jeda\": "<< jeda <<
100 ", \"ukuran \": " << ukuran  << ", \"jarak\": " << jarak
101 << "}'.\n";
102             if(counter==100){
103             myfile << "\n";
104             }
105             myfile.close();
106             std::cout << "\n";
107       }
108
109       void delivery_complete(mqtt::delivery_token_ptr
110 token) override {
111       std::cout << "Delivery complete..." << std::endl;
112       }
```

Pada Kode Program 6 menampilkan kode program yang digunakan pada saat menerima data dari *gateway*. Pada baris 1 memanggil kelas *callback* yang berisi *method* yang akan digunakan. Pada baris 12-24 adalah *method reconnect* yang akan dipanggil apabila koneksi dengan *broker* terputus. Pada baris 27-33 adalah *method on_failure* yang akan dipanggil ketika koneksi ulang gagal. Pada baris 36-52 adalah *method on_success* dipanggil ketika koneksi dengan *broker* dapat terhubung. Pada baris 50 akan dilakukan *subscribe* dengan topik dan qos yang telah ditentukan yaitu topik"sensor/#" dan qos 1. Pada baris 55-65 adalah *method connection_lost* yang akan dipanggil ketika koneksi dengan broker terputus dan akan mencoba melakukan koneksi ulang dengan broker seperti pada bari 64 *reconnect()*. Baris 68-107 adalah *method message_arrived* yang dipanggil ketika terdapat pesan yang masuk. Pada baris 75-80 data dengan bentuk *json* diambil nilainya. Baris 82-89 akan menampilkan nilai yang telah diambil. Pada baris 91-106 menuliskan nilai yang didapatkan menjadi bentuk .txt. Baris 109-112 adalah *method delivery_complete* yang akan dipanggil ketika pesan berhasil dikirimkan.

Kode Program 7 : Mengirim Data

```
1   int main(int argc, char* argv[]){
2         mqtt::connect_options connOpts;
3         connOpts.set_keep_alive_interval(20);
4         connOpts.set_clean_session(true);
5         connOpts.set_user_name(user);
6         connOpts.set_password(pass);
7         counter = 0;
8         json j;
9         mqtt::async_client client(SERVER_ADDRESS,
10  CLIENT_ID);
11        callback cb(client, connOpts);
12        client.set_callback(cb);
13        mqtt::message_ptr pubmsg;
14        // Start the connection.
15        // When completed, the callback will subscribe to
16  topic.
17
18        try {
19
```

```
20              std::cout << "Connecting to the MQTT
21    server..." << std::flush;
22              client.connect(connOpts, nullptr, cb);
23        }
24        catch (const mqtt::exception&) {
25              std::cerr << "\nERROR: Unable to connect to
26    MQTT server: '"
27                    << SERVER_ADDRESS << "'" << std::endl;
28              return 1;
29        }
30
31        auto input = ' ';
32        while (input != 'q'){
33          input = std::tolower(std::cin.get());
34              if ((input == '1')){
35                //payload2 = input;
36                int input2 = 1;
37                j["input"] = input2;
38                pubmsg = mqtt::make_message(TOPIC2,
39    j.dump());
40                pubmsg->set_qos(QOS);
41                client.publish(pubmsg)->wait_for(TIMEOUT);
42                std::cout << "  ...OK" << std::endl;
43              }
44              else if (input == '0'){
45                int input2 = 0;
46                j["input"] = input2;
47                pubmsg = mqtt::make_message(TOPIC2,
48    j.dump());
49                pubmsg->set_qos(QOS);
50                client.publish(pubmsg)->wait_for(TIMEOUT);
51                std::cout << "  ...OK" << std::endl;
52              }
53        }
54        //;
55        // Disconnect
56
57        try {
58              std::cout << "\nDisconnecting from the MQTT
59    server..." << std::flush;
60              client.disconnect()->wait();
61              std::cout << "OK" << std::endl;
62        }
63        catch (const mqtt::exception& exc) {
64              std::cerr << exc.what() << std::endl;
65              return 1;
66        }
67        return 0;
68  }
```

Pada Kode Program 7 menampilkan kode program yang digunakan untuk mengirim data menuju ke *gateway*. Pada baris 2-6 melakukan konfigurasi untuk protokol mqtt yang akan digunakan. Baris 9 menginsialisasi *client* dengan nama *client*. Baris 21 *client* menjalankan fungsi *connect* agar terkoneksi pada broker. Baris 30-45 proses *input* untuk nilai yang digunakan untuk menjalankan aktuator pada *node* sensor. Data *input* yang didapatkan akan dirubah menjadi tipe data *json* sebelum dikirimkan. Pada baris 37-39 membuat pesan sesuai dengan topik dan

qos yang telah ditentukan. Pada baris 40 melakukan *publish* untuk mengirimkan data *input* menuju ke *gateway*. Baris 57-67 adalah opsi memutuskan koneksi dengan *broker* jika nilai input bernilai ‘q’.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab pengujian akan menjelaskan tentang uji sistem berdasarkan implementasi dari sistem. Tahapan pengujian dilakukan untuk mengetahui cara bekerjanya sistem sesuai dengan yang telah dituliskan pada bab rekayasa kebutuhan tentang kebutuhan fungsional dan non-fungsional.

6.1 Pengujian Kebutuhan Fungsional

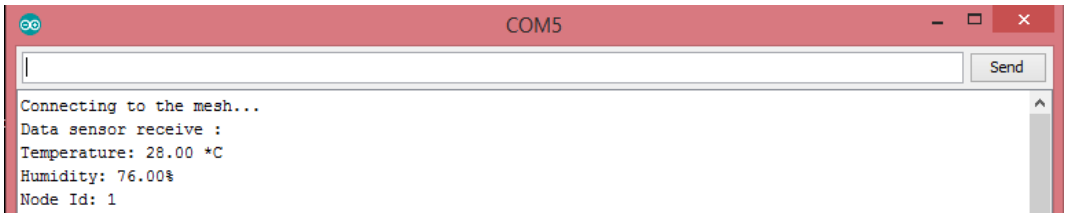
Pengujian kebutuhan fungsional dilakukan agar dapat mengetahui apakah fungsi yang dituliskan hasil implementasi dengan perancangan telah sesuai atau tidak. Pengujian fungsional dilakukan secara *black-box* yang mana pengujian *black-box* merupakan pengujian yang dilakukan hanya untuk mengamati hasil akhir dari eksekusi program tersebut. Untuk skenario pada pengujian kebutuhan fungsional dilakukan dengan mengikuti skenario yang telah di buat pada bab perancangan pengujian.

6.1.1 Pengujian Kode FR-01

Kasus uji proses pengambilan data sensor ditunjukkan pada Tabel 6.1.

Tabel 6.1 Pengujian Kode FR-01

Kode	FR-01
Fungsi	*Node* sensor dapat mengambil data sensor
Tujuan Pengujian	Menguji proses pengambilan data sensor
Skenario Pengujian	1. *Node* sensor keadaan menyala 2. Perangkat sensor terhubung dengan *node* sensor 3. Mengambil data sensor menggunakan perangkat sensor 4. Menampilkan hasil data sensor yang ditangkap.
Hasil yang Diharapkan	Data sensor berhasil diambil oleh *node* sensor
Hasil Pengujian	Data sensor berhasil di ambil



Gambar 6.1 Hasil Dari Pengujian FR-01

Pada Gambar 6.1 menunjukkan tampilan pada *serial* monitor Arduino yang diambil pada *node* sensor. Sensor yang digunakan DH11 mengambil data berupa kelembaban dan suhu udara. Mikrokontroler yang digunanakan untuk menjalankan *node* sensor menggunkan Arduino Nano. Hasil yang ditampilkan menujukkan bahwa *node* sensor berhasil mengambil data sensor. Pada pengujian ini memperlihatkan bahwa sensor DHT11 dapat diimplementasikan pada *node* sensor untuk menangkat data lingkungan yaitu kelembaban dan suhu udara.

6.1.2 Pengujian Kode FR-02

Kasus uji proses pengiriman data sensor ditunjukkan pada Tabel 6.2.

Tabel 6.2 Pengujian Kode FR-02

Kode	FR-02
Fungsi	*Node* sensor dapat mengirim data sensor menuju *gateway*.
Tujuan Pengujian	Menguji proses pengiriman data sensor
Skenario Pengujian	1. *Node* sensor keadaan menyala 2. Modul nRF24L01 keadaan menyala 3. *Node* sensor telah mengambil data dari perangkat sensor 4. *Node* sensor mengirim data menuju perangkat *gateway*.
Hasil yang Diharapkan	Data sensor berhasil dikirim oleh *node* sensor
Hasil Pengujian	Data sensor dikirim di ambil

Gambar 6.2 Hasil Dari Pengujian FR-02

Pada gambar 6.2 menunjukkan tampilan pada *serial* monitor Arduino yang diambil pada *node* sensor. Sensor yang digunakan DH11 telah berhasil mengambil data berupa kelembaban dan suhu udara. *Node* id merupakan nomer dari *node* pengirim, yang mana nilai 1 menunjukkan *node* sensor sebagai *sender* dan nilai 0 menunjukkan *gateway* sebagai *receiver*. Hasil yang ditampilkan *send ok* : 1 menujukkan bahwa *node* sensor berhasil mengirimkan data sensor. Pengiriman data dari *node* sensor menuju *gateway* menggunakan bentuk data *struct.* Pada pengujian ini memperlihatkan bahwa pengiriman data sensor dari *node* sensor

menuju *gateway* dapat dilakukan dengan menggunakan modul komunikasi nRF24L01 dan protokol RF24Mesh.

6.1.3 Pengujian Kode FR-03

Kasus uji proses menerima data sensor ditunjukkan pada Tabel 6.3.

Tabel 6.3 Pengujian Kode FR-03

Kode	FR-03
Fungsi	Perangkat *gateway* dapat menerima data sensor yang dikirimkan oleh *node* sensor.
Tujuan Pengujian	Menguji penerimaan data sensor pada *gateway*
Skenario Pengujian	1. *Gateway* dalam keadaan menyala 2. Modul nRF24L01 dalam keadaanya menyala 3. *Gateway* menerima data dari *node* sensor.
Hasil yang Diharapkan	Data sensor berhasil diterima oleh *gateway*
Hasil Pengujian	*Gateway* berhasil menerima data sensor

```
Initializing for server 'tcp://m15.cloudmqtt.com:13562'...
Connecting...
Waiting for the connection...
  ...OK


Receive packet from sensor node. Details packets :
Temp : 28
Humidity : 76
Packet number : 1
From node id : 1
Counter : 1
```

Gambar 6.3 Hasil Dari Pengujian FR-03

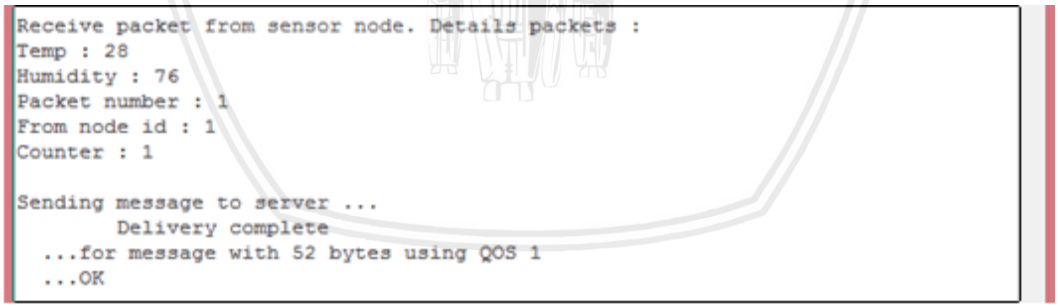
Pada Gambar 6.3 menunjukkan tampilan pada terminal Raspberry Pi yang diambil pada perangkat *gateway*. Mikrokontroler yang digunanakan untuk menjalankan *gateway* menggunkan Raspberry Pi. Hasil yang ditampilkan menunjukkan bahwa perangkat *gateway* berhasil menerima data sensor yang dikirimkan oleh *node* sensor. Temp dan humidity menunjukan nilai suhu dan kelembaban dari tanah. Sedangkan packet number menunjukkan nomer paket yang dikirimkan. *Node* id menujukkan asal dari data tersebut dimana nilai 1 menujukkan perangkat sensor. *Counter* menujukkan urutan data yang masuk pada *gateway*. Pada pengujian ini memperlihatkan bahwa komunikasi antara *node* sensor menuju *gateway* dapat dilakukan dengan menggunakan modul komunikasi nRF24L01 dan protokol RF24Mesh.

6.1.4 Pengujian Kode FR-04

Kasus uji proses meneruskan data sensor menuju ke *server* ditunjukkan pada Tabel 6.4

Tabel 6.4 Pengujian Kode FR-04

Kode	FR-04
Fungsi	Perangkat *gateway* dapat meneruskan data sensor menuju ke *server*.
Tujuan Pengujian	Menguji pengiriman data sensor dari *gateway* menuju ke *server*.
Skenario Pengujian	1. *Gateway* dalam keadaan menyala 2. Modul nRF24L01 dalam keadaan menyala 3. *Gateway* telah menerima data dari *node* sensor 4. *Gateway* meneruskan data menuju *broker* dengan *publish* topik 5. *Broker* meneruskan ke *server* jika telah melakukan *subscribe* ke topik yang sesuai.
Hasil yang Diharapkan	Data sensor berhasil diteruskan oleh *gateway*
Hasil Pengujian	*Gateway* berhasil meneruskan data sensor



Gambar 6.4 Hasil Dari Pengujian FR-04

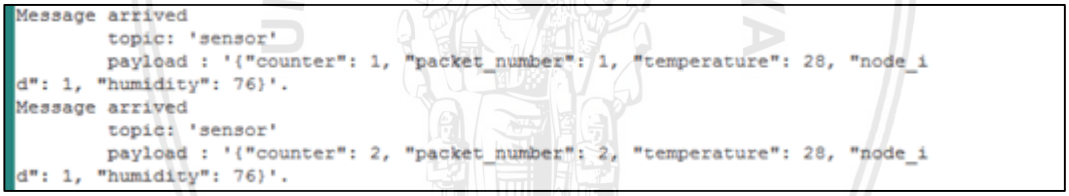
Pada Gambar 6.4 menunjukkan tampilan pada terminal Raspberry Pi yang diambil pada perangkat *gateway*. Data yang berhasil diterima akan dikirimkan ke *server* dengan menggunakan protokol MQTT. Komunikasi protokol MQTT menggunakan metode *publish-subscribe,* dimana *gateway* akan melakukan *publish* data ke *server* yang dikirim melalui *broker* dengan topik "sensor/1" dengan qos 1. Hasil yang ditampilkan pada Gambar 6.4 menunjukkan bahwa *gateway* berhasil meneruskan data sensor ke *broker*. Bentuk data yang diteruskan pada tahap ini berupa json yang sebelumnya berupa bentuk data *struct* yang telah diubah. Pada pengujian ini memperlihatkan bahwa pengiriman data sensor dari *gateway* menuju *server* dapat dilakukan dengan menggunakan protokol MQTT.

6.1.5 Pengujian Kode FR-05

Kasus uji proses menerima data input pada *server* dan mengirimkan menuju *gateway* ditunjukkan pada Tabel 6.5.

Tabel 6.5 Pengujian Kode FR-05

Kode	FR-05
Fungsi	*Server* dapat menerima data sensor dari perangkat *gateway*.
Tujuan Pengujian	Menguji *server* dapat menerima data dari perangkat *gateway*.
Skenario Pengujian	1. *Server* keadaan menyala 2. *Server* mendapatkan data dengan melakukan *subscribe* sesuai topik 3. *Server* menampilkan data yang didapatkan
Hasil yang Diharapkan	Data sensor berhasil diterima oleh *server*
Hasil Pengujian	*Server* berhasil menerima data sensor



Gambar 6.5 Hasil Dari Pengujian FR-05

Pada Gambar 6.5 menunjukkan tampilan pada terminal *server*. Data dari *gateway* dikirimkan ke *server* melalui *broker. Broker* kemudian meneruskan data ke *server* sesuai dengan topik yang telah di *subscribe* oleh *server*. Hasil yang ditampilkan menunjukkan bahwa *server* berhasil menerima data dari *node* sensor yang dikirim melalui *gateway* dengan topik *nodes/#*. Data yang ditampilkan pada *server* berupa *counter* pada *server*, nomer paket yang masuk, *node* id dari sensor, data suhu dan kelembaban tanah. Pada pengujian ini memperlihatkan bahwa komunikasi antara *gateway* menuju *server* dapat dilakukan dengan menggunakan protokol MQTT.

6.1.6 Pengujian Kode FR-06

Kasus uji proses meneruskan data input menuju ke *gateway* ditunjukkan pada Tabel 6.6.

Tabel 6.6 Pengujian Kode FR-06

Kode	FR-06

Fungsi	*Server* dapat menerima data *input* perintah dan mengirimkan data menuju perangkat *gateway*.
Tujuan Pengujian	Menguji *server* dapat menerima input data dan mengirimkan pada perangkat *gateway*.
Skenario Pengujian	1. *Server* keadaan menyala 2. *Server* mendapat nilai *input* 3. *Gateway* keadaan menyala 4. *Server* mengirimkan data *input* dengan *publish* ke *broker* 5. *Broker* meneruskan data ke *gateway* jika telah melakukan *subscribe* topik yang sesuai.
Hasil yang Diharapkan	Data input berhasil di dapatkan dan dikirimkan menuju *gateway*
Hasil Pengujian	Data input berhasil didapatkan dan dikirimkan ke *gateway*

```
Connecting to the MQTT server...
Connection success

Subscribing to topic 'sensor/#'
        for client cloudserver using QoS1

Press Q<Enter> to quit

Press 1<Enter> to turn on the light in sensor node

Press 0<Enter> to turn off the light in sensor node
```

Gambar 6.6 Tampilan Terminal *Server*

Pada Gambar 6.6 menunjukkan tampilan pada terminal *server*. Data yang diinputkan pada *server* selanjutnya akan dikirimkan pada *gateway* dengan protokol MQTT. Komunikasi protokol MQTT menggunakan metode *publish-subscribe,* dimana *server* akan melakukan *publish* data ke *gateway* yang dikirimkan melalui *broker* dengan topik “cloud” dengan qos 1.

```
0
Delivery complete...
   ...OK
```

Gambar 6.7 Hasil Dari Pengujian FR-06

Pada Gambar 6.7 menampilkan hasil data inputan berhasil diterima dan dikirimkan menuju ke *broker*. Terdapat pesan “*delivery complete*” yang

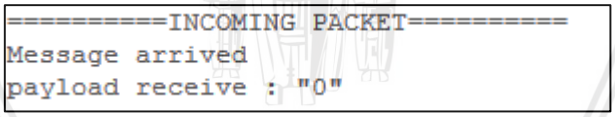
menujukkan data telah berhasil dikirimkan. Pada pengujian ini memperlihatkan bahwa pengiriman data *input* dari *server* menuju *gateway* dapat dilakukan dengan menggunakan protokol MQTT.

6.1.7 Pengujian Kode FR-07

Kasus uji proses *gateway* menerima data dari *server* ditunjukkan pada Tabel 6.7.

Tabel 6.7 Pengujian Kode FR-07

Kode	FR-07
Fungsi	Perangkat *gateway* dapat menerima data dari *server*.
Tujuan Pengujian	Menguji *gateway* dapat menerima data dari *server*.
Skenario Pengujian	1. Perangkat *gateway* dalam keadaan menyala 2. Perangkat *gateway* telah melakukan *subscribe* topik yang sesuai 3. *Gateway* menampilkan data input dari *broker*
Hasil yang Diharapkan	Data berhasil diterima oleh *gateway*
Hasil Pengujian	Data berhasil diterima oleh *gateway*



Gambar 6.8 Hasil Dari Pengujian FR-07

Pada Gambar 6.8 menunjukkan tampilan pada terminal Raspberry Pi yang diambil pada perangkat *gateway*. Hasil yang ditampilkan menunjukkan bahwa perangkat *gateway* berhasil menerima input dikirimkan oleh *server*. *Payload* yang diterima berisi nilai yang diinputkan pada *server*. Pada pengujian ini memperlihatkan bahwa komunikasi antara *server* menuju *gateway* dapat dilakukan dengan menggunakan protokol MQTT.

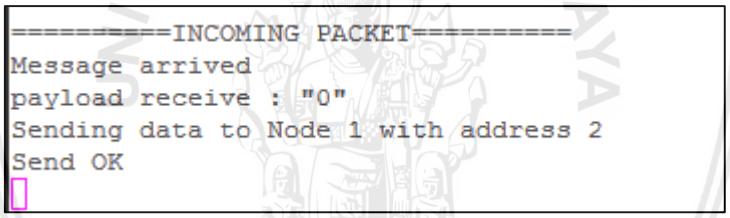
6.1.8 Pengujian Kode FR-08

Kasus uji proses meneruskan data input dari *gateway* menuju *node* sensor ditunjukkan pada Tabel 6.8.

Tabel 6.8 Pengujian Kode FR-08

Kode	FR-08

Fungsi	Perangkat *gateway* dapat meneruskan data menuju ke *node* sensor.
Tujuan Pengujian	Menguji *gateway* dapat meneruskan data menuju ke *node* sensor.
Skenario Pengujian	1. Perangkat *gateway* dalam keadaan menyala 2. Modul komunikasi nRF24L01 dalam keadaan menyala 3. *Node* sensor dalam keadaan menyala 4. *Gateway* mengirimkan data menuju *node* sensor.
Hasil yang Diharapkan	Data berhasil diteruskan oleh *gateway* ke *node* sensor
Hasil Pengujian	Data berhasil diteruskan oleh *gateway* ke *node* sensor



Gambar 6.9 Hasil Dari Pengujian FR-08

Pada Gambar 6.9 menunjukkan tampilan pada terminal Raspberry Pi yang diambil pada perangkat *gateway*. Data yang berhasil diterima akan dikirimkan ke *node* sensor menggunakan protokol RF24Mesh. *Gateway* akan mengirimkan data menuju *node* 1 yang memiliki alamat 2 yang tercatat pada perangkat *gateway*. Pesan "*send ok*" menunjukka data berhasil di kirim menuju ke *node* sensor. Pada pengujian ini memperlihatkan bahwa pengiriman data *input* dari *server* menuju *gateway* dapat dilakukan dengan menggunakan protokol RF24Mesh dan modul komunikasi nRF24L01.

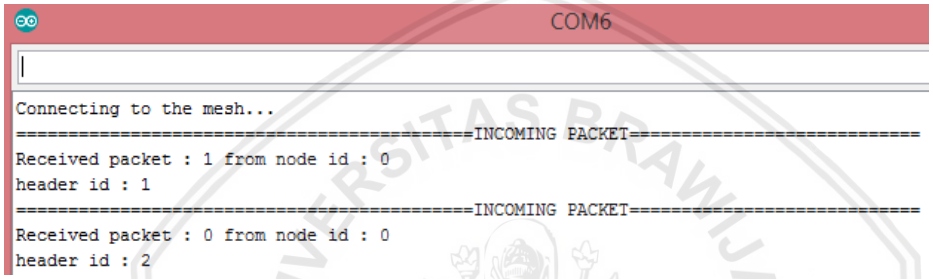
6.1.9 Pengujian Kode FR-09

Kasus uji proses *node* sensor menerima data dari *gateway* ditunjukkan pada Tabel 6.9.

Tabel 6.9 Pengujian Kode FR-09

Kode	FR-09
Fungsi	*Node* sensor dapat menerima data dari perangkat *gateway*.

Tujuan Pengujian	Menguji *node* sensor dapat menerima data dari *gateway*.
Skenario Pengujian	1. *Node* sensor dalam keadaan menyala 2. Modul komunikasi nRF24L01 dalam keadaan menyala 3. *Node* sensor menampilkan data *input* dari *gateway*.
Hasil yang Diharapkan	*Node* sensor dapat menerima data dari *gateway*
Hasil Pengujian	*Node* sensor menerima data dari *gateway*



Gambar 6.10 Hasil Dari Pengujian FR-09

Gambar 6.10 menunjukkan tampilan pada *serial* monitor Arduino yang diambil pada *node* sensor. Hasil yang ditampilkan menujukkan data yang dikirimkan oleh *gateway* dapat diterima oleh *node* sensor. *Received packet* menujukkan isi dari dari data yang dikirimkan dari sensor. *Node* id menunjukkan darimana packet berasal, nilai 0 menujukkan packet berasal dari *gateway*. *Header id* menujukkan nomer dari *header* yang dikirimkan. Pada pengujian ini memperlihatkan bahwa komunikasi antara *gateway* menuju *node* sensor dapat dilakukan dengan menggunakan protokol RF24Mesh dan modul komunikasi nRF24L01.

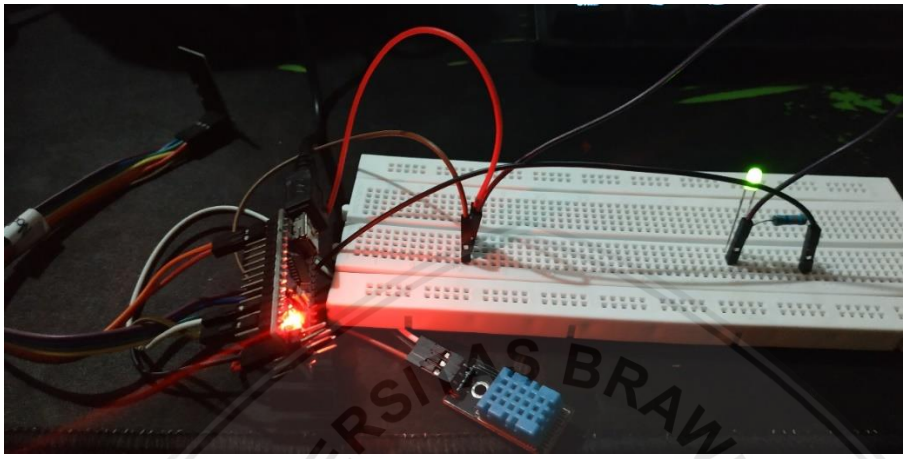
6.1.10 Pengujian Kode FR-10

Kasus uji proses menjalankan aktuator pada *node* sensor ditunjukkan pada Tabel 6.10.

Tabel 6.10 Pengujian Kode FR-10

Kode	FR-10
Fungsi	*Node* sensor menjalankan aktuator sesuai data yang diterima dari perangkat *gateway*.
Tujuan Pengujian	Menguji sensor dapat menjalankan aktuator
Skenario Pengujian	1. LED telah terpasang dalam kondisi mati 2. *Node* sensor telah menerima data dari *gateway*

	3. LED menyala sesuai dengan nilai input yang didapatkan.
Hasil yang Diharapkan	Lampu LED sebagai aktuator dapat menyala
Hasil Pengujian	Lampu LED sebagai aktuator menyala



Gambar 6.11 Hasil Dari Pengujian FR-10

Gambar 6.11 menujukkan hasil dari pengujian aktuator terhadap *node* sensor. Ketika data yang dikirimkan bernilai satu akan membuat lampu LED menjadi menyala berwana hijau. Pada pengujian ini memperlihatkan bahwa aktuator dapat bekerja sesuai dengan *input* yang dikirimkan oleh *server*.

6.2 Pengujian Performa

Pengujian performa dilakukan untuk menguji keandalan sistem yang telah dibuat. Pengujian performa berfokus pada parameter *successfull rate* dalam menerima dan meneruskan paket sampai ke tujuan. Pengujian dilakukan dengan menggunakan aplikasi putty untuk melakukan SSH pada *server* untuk melihat jumlah paket yang berhasil masuk . Skenario pengujian performa telah dijelaskan pada bab perancangan. Sehingga disini akan langsung ditunjukkan hasil yang diperoleh pengujian performa yang sudah dilakukan. Kebutuhan, batasan serta hasil yang diharapkan dalam pengujian performa dapat dilihat pada Tabel 6.11

Tabel 6.11 Pengujian Performa Sistem

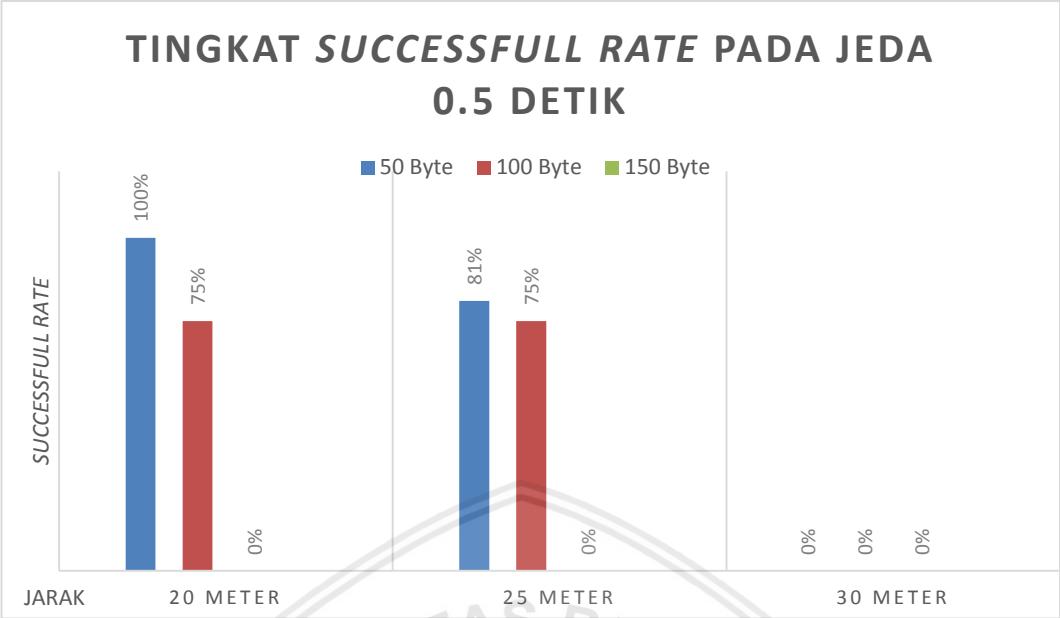
Parameter	Keterangan
Successfull Rate	Menggunakan *tools* putty untuk melakukan pengumpulan data dengan variasi jarak sebesar 20 meter, 25 meter, 30 meter dengan besar paket 50 *Byte,* 100 *Byte,* 150 *Byte* dan interval waktu 0,5 detik, 1 detik, 5 detik.

Pengujian ini diawali dengan melakukan ssh ke *server* dengan menggunakan aplikasi putty yang akan digunakan untuk pengambilan data. Tahap selanjutnya yaitu mempersiapkan perangkat yang akan digunakan seperti *node* sensor, *gateway*¸ dan *server*. Dibutuhkan beberapa laptop sebagai sumber daya untuk perangkat Raspberry Pi, Arduino nano dan *server* yang akan digunakan untuk pengujian. Pada setiap jarak, besar paket, dan interval yang jelaskan akan dilakukan proses pengirman data sebesar 100 paket untuk mendapatkan hasil yang akurat.

6.2.1 Pengujian Jeda 0,5 Detik

Pengujian jeda dilakukan dengan memeberikan jeda pada pengiriman setiap paket selama 0,5 detik. Pengujian dilakukan menggunakan variasi jarak yang telah ditentukan yaitu sebesar 20 meter, 25 meter, 30 meter. Pada setiap jarak terdapat variasi ukuran paket yang dikirimkan yaitu 50 *Byte,* 100 *Byte* dan 150 *Byte.* Pengujian ini dilakukan dengan pengiriman paket sebanyak 100. Pada setiap percobaan akan diperoleh tingkat *successful rate* dengan tingkat ukuran paket pada setiap variasi jarak ditunjukkan pada Gambar 6.12

Pada Gambar 6.12 menunjukkan hasil dari pengujian performa pada jeda waktu 0,5 detik. Didapatkan hasil 100% pengiriman berhasil pada jarak 20 meter dengan ukuran paket 50 *Byte* dan didapatkan hasil 75% pada jarak 20 meter dengan ukuran paket 100 *Byte*. Pada jarak 25 meter hasil pengujian yang didapatkan mengalami penurunan menjadi sebesar 81% dan 75% untuk masing masing ukuran paket 50 dan 100 *Byte*. Untuk pengujian dengan skenario ukuran paket 150 *Byte* dan jarak 30 meter semuanya memiliki hasil tingkat *successful rate* sebesar 0%. Tingkat *successful rate* paling baik dengan jeda pengiriman paket 0,5 detik didapatkan pada ukuran paket dan jarak 50 *Byte* dan 20 meter. Pengiriman pada jarak 30 meter tidak berhasil dilakukan karena *node* sensor berada di luar jarak jangkauan dari *gateway.* Pengiriman pada ukuran paket 150 *Byte* memiliki tingkat kesuksesan 0% atau gagal dilakukan karena ukuran paket yang dikirimkan melebihi batas maksimum paket yang dapat dikirimkan dengan nRF24L01.

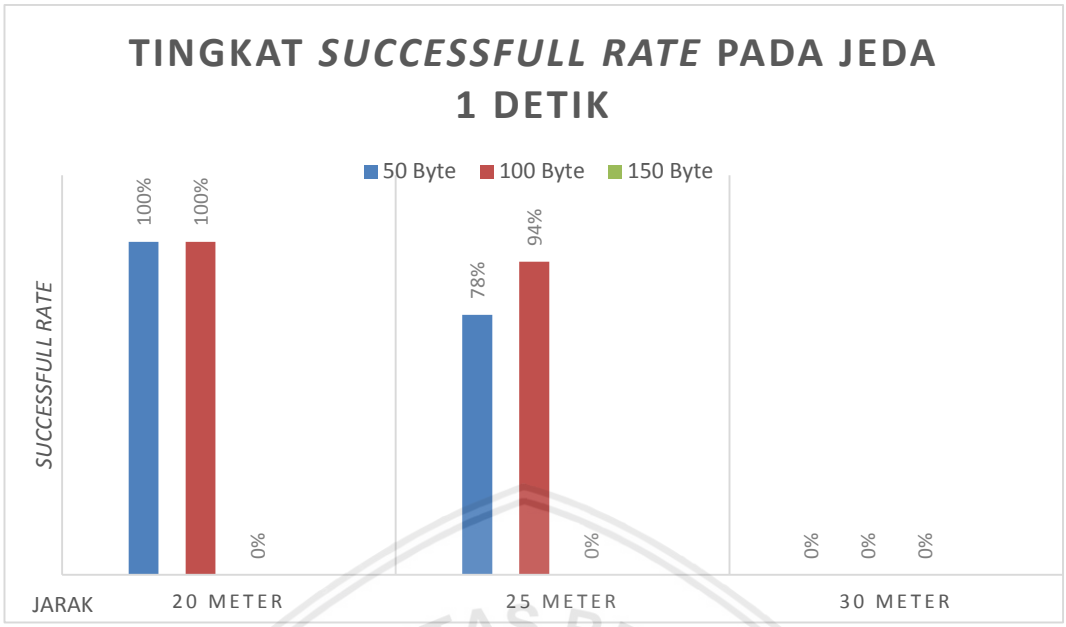


Gambar 6.12 Grafik Hasil Pengujian Dengan Jeda Waktu 0,5 Detik

6.2.2 Pengujian Jeda 1 Detik

Pengujian jeda dilakukan dengan memberikan jeda pada pengiriman setiap paket selama 1 detik. Pengujian dilakukan menggunakan variasi jarak yang telah ditentukan yaitu sebesar 20 meter, 25 meter, 30 meter. Pada setiap jarak terdapat variasi ukuran paket yang dikirimkan yaitu 50 *Byte,* 100 *Byte* dan 150 *Byte.* Pengujian ini dilakukan dengan pengiriman paket sebanyak 100. Pada setiap percobaan akan diperoleh tingkat *successful rate* dengan tingkat ukuran paket pada setiap variasi jarak ditunjukkan pada Gambar 6.13

Pada Gambar 6.13 menunjukkan hasil dari pengujian performa pada jeda waktu 1 detik. Didapatkan hasil 100% pengiriman berhasil pada jarak 20 meter dengan ukuran paket 50 *Byte* dan didapatkan hasil 100% pada jarak 20 meter dengan ukuran paket 100 *Byte*. Pada jarak 25 meter hasil pengujian yang didapatkan menurun menjadi sebesar 78% dan 94% untuk masing masing ukuran paket 50 dan 100 *Byte* Untuk pengujian dengan skenario ukuran paket 150 *Byte* dan jarak 30 meter semuanya memiliki hasil tingkat *successful rate* sebesar 0%. Tingkat *successful rate* paling baik dengan jeda pengiriman paket 1 detik didapatkan pada ukuran paket dan jarak 50 *Byte* dan 20 meter. Pengiriman pada jarak 30 meter tidak berhasil dilakukan karena *node* sensor berada di luar jarak jangkauan dari *gateway.* Pengiriman pada ukuran paket 150 *Byte* memiliki tingkat kesuksesan 0% atau gagal dilakukan karena ukuran paket yang dikirimkan melebihi batas maksimum paket yang dapat dikirimkan dengan nRF24L01.

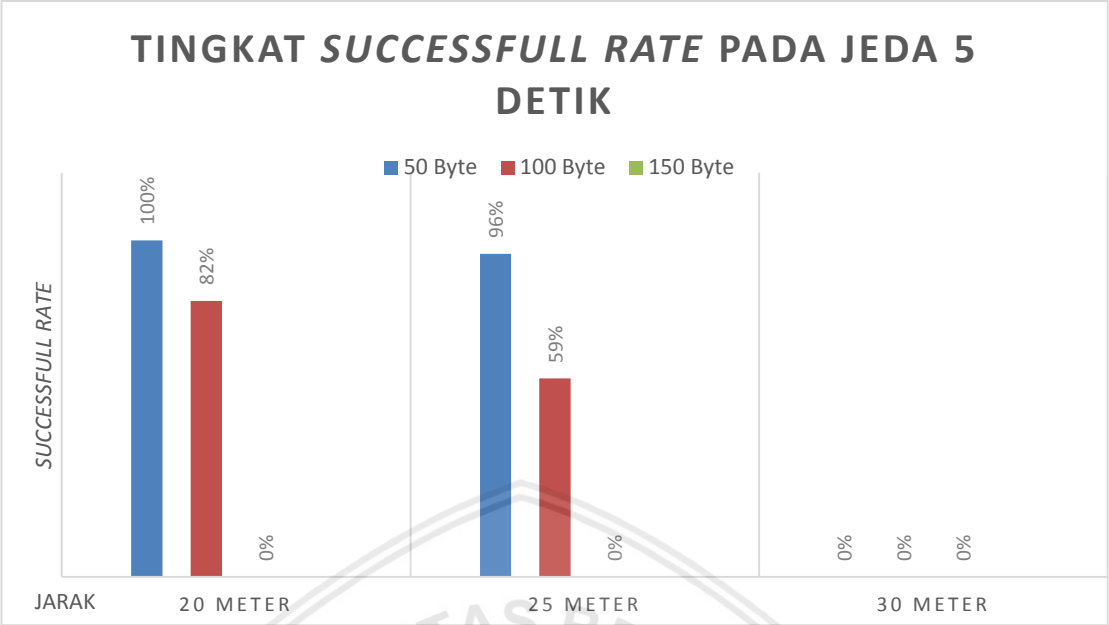


Gambar 6.13 Grafik Hasil Pengujian Dengan Jeda Waktu 1 Detik

6.2.3 Pengujian Jeda 5 Detik

Pengujian jeda dilakukan dengan memberikan jeda pada pengiriman setiap paket selama 5 detik. Pengujian dilakukan menggunakan variasi jarak yang telah ditentukan yaitu sebesar 20 meter, 25 meter, 30 meter. Pada setiap jarak terdapat variasi ukuran paket yang dikirimkan yaitu 50 *Byte,* 100 *Byte* dan 150 *Byte.* Pengujian ini dilakukan dengan pengiriman paket sebanyak 100. Pada setiap percobaan akan diperoleh tingkat *successful rate* dengan tingkat ukuran paket pada setiap variasi jarak ditunjukkan pada Gambar 6.14

Pada Gambar 6.14 menunjukkan hasil dari pengujian performa pada jeda waktu 5 detik. Didapatkan hasil 100% pengiriman berhasil pada jarak 20 meter dengan ukuran paket 50 *Byte* dan didapatkan hasil 82% pada jarak 20 meter dengan ukuran paket 100 *Byte*. Pada jarak 25 meter hasil pengujian yang didapatkan menurun menjadi sebesar 96% dan 59% untuk masing masing ukuran paket 50 dan 100 *Byte* Untuk pengujian dengan skenario ukuran paket 150 *Byte* dan jarak 30 meter semuanya memiliki hasil tingkat *successful rate* sebesar 0%. Tingkat *successful rate* paling baik dengan jeda pengiriman paket 5 detik didapatkan pada ukuran paket dan jarak 50 *Byte* dan 20 meter. Pengiriman pada jarak 30 meter tidak berhasil dilakukan karena *node* sensor berada di luar jarak jangkauan dari *gateway.* Pengiriman pada ukuran paket 150 *Byte* memiliki tingkat kesuksesan 0% atau gagal dilakukan karena ukuran paket yang dikirimkan melebihi batas maksimum paket yang dapat dikirimkan dengan nRF24L01.



Gambar 6.14 Grafik Hasil Pengujian Dengan Jeda Waktu 5 Detik

BAB 7 PENUTUP

Bab ini berisi kesimpulan yang ditarik dari bab analisis hasil pengujian yang menjawab rumusan masalah dan juga saran yang dapat digunakan untuk mengembangkan penelitian ini ke depannya.

7.1 Kesimpulan

Kesimpulan yang diperoleh berdasarkan hasil pada proses bab perancangan, implementasi, dan pengujian yang telah dilakukan adalah sebagai berikut:

1. Pengiriman data sensor menggunakan pengimplementasian perangkat modul komunikasi nRF24L01. Perangkat nRF24L01 dapat diimplementasikan dengan cara melakukan instalasi perangkat nRF24L01 pada mikrokontroler Arduino Nano. Pada mikrokontroler juga terpasang sensor DHT11 untuk mengambil data kelembaban dan suhu udara. Setelah seluruh komponen pada *node* sensor telah terpasang. *Node* sensor akan menghasilkan data sensor dan mengirimkan datanya menggunakan bentuk data *struct* menuju ke perangkat *gateway*. Komunikasi antara *node* sensor dengan *gateway* menggunakan protokol RF24Mesh dengan *library* yang telah ada. Hal yang perlu diperhatikan dalam menggunakan perangkat nRF24L01 dalam pengiriman data adalah menentukan *node id* dari perangkat *node* sensor dan *gateway*. *Node id* ini yang akan membedakan mana *sender* dan *receiver*.

2. *Gateway* diimplementasikan menggunakan mikrokomputer Raspberry Pi dan modul komunikasi nRF24L01. nRF24L01 menjembatani komunikasi nirkabel pada sistem yang telah dibuat. Sistem pada *node* sensor diimplementasikan dengan perangkat sensor mengambil data dari *node* sensor kemudian mengirimkan data menuju ke *gateway* dengan menggunakan modul komunikasi nRF24L01. Protokol yang digunakan adalah RF24Mesh. *Gateway* dapat menerima data dari *node* sensor dengan menggunakan modul komunikasi nRF24L01. Untuk meneruskan data dari perangkat *gateway* menuju ke *server* menggunakan protokol komunikasi MQTT. Pengiriman data dari *gateway* menuju ke server menggunakan data json.

3. Menurut hasil dari pengujian performa dengan parameter *successful rate* dari *gateway* yang dibangun dengan menggunakan modul nRF24L01 yang telah dilakukan sebelumnya dapat disimpulkan bahwa pada jarak 20 meter *gateway* memiliki performa yang lebih baik daripada jarak 25 meter. Performa yang didapatkan dengan jeda waktu yang paling baik pada nilai jeda 1 detik. Performa yang didapatkan dengan ukuran paket yang paling baik pada ukuran paket 50 *byte.* Pada ukuran paket 150 *byte* memiliki performa 0% pada

seluruh skenario pengujiannya. Jarak 30 meter juga memiliki performa 0% yang dengan kata lain tidak ada data yang berhasil diteruskan oleh *gateway* karena perangkat nRF24L01 tidak dapat menjangkau jarak 30 meter. Sama halnya dengan ukuran paket 150 byte, seluruh paket yang dikirimkan drop dikarenakan melebihi ukuran paket maksimum yang dapat dikirimkan.

7.2 Saran

Pada penelitian ini penulis merasa masih memiliki beberapa kekurangan untuk itu diperlukan perbaikan dan pengembangan di penelitian selanjutnya untuk terciptanya penelitian yang lebih baik. Beberapa hal di bawah ini adalah beberapa saran yang dapat diajukan oleh penulis untuk penelitian selanjutnya:

1. Pada penelitian ini perangkat nRF24L01 tidak menggunaka antena sehingga jarak yang dijangkau tidaklah terlalu jauh. Diharapkan kedepannya perangkat nRF24L01 yang digunakan menggunakan antena agar dapat jarak komunikasi dapat ditingkatkan dan melihat perbedaan performa dari perangkat dengan antena dan tidak dengan antena.
2. Pada penelitian ini jumlah *node* sensor yang digunakan hanya 1 buah. Diharapkan kedepannya dapat dilakukan penambahan jumlah *node* sensor agar dapat terbentuk sebuah jaringan Mesh yang seutuhnya.
3. Dalam penelitian ini *server* hanya menggunakan 1 buah perangkat *gateway*. Diharapkan pada penelitian kedepannya dapat ditambahkan jumlah *gateway* agar sistem dapat berjalan lebih kompleks.
4. Parameter yang digunakan pada pengujian penelitian ini adalah *successful rate*. Diharapkan pada penelitian berikutnya dapat menambahkan parameter lain seperti *throughtput* dan *delay*.

DAFTAR REFERENSI

Af'idah, D. I., Rochim, A. F., & Widianto, E. D. (2014). PERANCANGAN JARINGAN SENSOR NIRKABEL (JSN). *Jurnal Teknologi dan Sistem Komputer*, 267-276.

Arduino. (2018). *ARDUINO NANO*. Dipetik February 17, 2018, dari https://store.arduino.cc/usa/arduino-nano

Arduino. (2018). *GETTING STARTED | FOUNDATION > Introduction*. Dipetik February 17, 2018, dari https://www.arduino.cc/en/Guide/Environment

Arduino. (2018). *GETTING STARTED | FOUNDATION > Introduction*. Dipetik February 17, 2018, dari https://www.arduino.cc/en/Guide/Introduction#

Bade, D., & Lamersdorf, W. (2011). An Agent-Based Event Processing Middleware for Sensor Networks and RFID Systems. *The Computer Journal, 54*(3), 321 - 331.

Cruz, M. A., Rodrigues, J. J., Al-Muhtadi, J., Korotaev, V., & Albuquerque, V. H. (2018). A Reference Model for Internet of Things Middleware. *IEEE Internet of Things Journal, PP*(99), 1-1.

Habibi, M. W., Bhawiyuga, A., & Basuki, A. (2018). Rancang Bangun IOT Cloud Platform Berbasis Protokol Komunikasi MQTT. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, II*(2), 479-485.

Hanifah, S., Akbar, S. R., & Amron, K. (2018). Implementasi Quality of Service pada Protokol Message Queue Telemetry Transport – Sensor Network (MQTT-SN) Berbasis Arduino dan NRF24L01. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, II*(6), 2131-2140.

Mendez, G. R., Yunus, M. A., & Mukhopadhyay, S. C. (2012). A WiFi based Smart Wireless Sensor Network for an Agricultural Environment. (hal. 2640 - 2645). Graz: IEEE.

NordicSemiconduktor. (2014). *nRF24L01+*. Dipetik April 2, 2018, dari https://www.nordicsemi.com/kor/Products/2.4GHz-RF/nRF24L01P

Nuratch, S. (2017). The IIoT devices to cloud gateway design and implementation based on microcontroller for real-time monitoring and control in automation systems. *Industrial Electronics and Applications (ICIEA)*(12), 919-923.

Pratama, R. P., Akbar, S. R., & Bhawiyuga, A. (2017). Rancang Bangun Low Power Sensor Node Menggunakan MSP430 Berbasis NRF24L01. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 1*(3), 157-165.

Raspberry Pi. (2018). *WHAT IS A RASPBERRY PI?* Dipetik February 17, 2018, dari https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/

Shobrina, U. J., Primananda, R., & Maulana, R. (2018). sis Kinerja Pengiriman Data Modul Transceiver NRF24l01, Xbee dan Wifi ESP8266 Pada Wireless Sensor

Network. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 1510-1517.

Stanley, M., Lee, J., & Spanias, A. (2017). *Sensor Analysis for the Internet of Things* (1 ed.). Morgan & Claypool.

tmrh20. (2015). *RF24Mesh*. Dipetik December 7, 2018, dari http://tmrh20.github.io/RF24Mesh/

Yassein, M. B., Shatnawi, M. Q., Aljwarneh, S., & Al-Hatmi, R. (2017). Internet of Things: Survey and open issues of MQTT protocol. *International Conference on Engineering & MIS (ICEMIS)*.