# PENERAPAN MEKANISME TRANSMISI DATA ECG BERBASIS TEKNOLOGI LORA (*LONG RANGE*)

**SKRIPSI**

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Fadila Rafi Alifiandi
NIM: 155150200111306


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

# PENGESAHAN

**PENERAPAN MEKANISME TRANSMISI DATA ECG BERBASIS TEKNOLOGI LORA (*LONG RANGE*)**

**SKRIPSI**

**Diajukan untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer**

**Disusun Oleh :**
**Fadila Rafi Alifiandi**
**NIM: 155150200111306**

**Skripsi ini telah diuji dan dinyatakan lulus pada**
**8 Januari 2020**
**Telah diperiksa dan disetujui oleh:**

| Dosen Pembimbing I | Dosen Pembimbing 2 |
|---|---|
| Eko Sakti Pramukantoro, S.Kom, M.Kom | Rakhmadhany Primananda, S.T, M.Kom |
| NIK: 201102 860805 1 001 | NIK: 201609 860406 1 001 |

**Mengetahui**
**Ketua Jurusan Teknik Informatika**

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

# PERNYATAAN ORISINALITAS

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

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

Malang, 03 Februari 2020


Fadila Rafi Alifiandi

NIM: 155150200111306

# PRAKATA

Puji syukur penulis panjatkan ke hadirat Allah SWT atas pertolongan dan nikmat sehat serta akal yang dilimpahkan-Nya sehingga penulis dapat menyelesaikan laporan skripsi dengan judul "Penerapan Mekanisme Transmisi Data ECG Berbasis Teknologi LoRa (*Long Range*)". Shalawat serta salam semoga selalu tercurahkan kepada Nabi Muhammad SAW beserta keluarga dan para sahabatnya hingga pada umatnya sampai akhir zaman.

Laporan skripsi ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya. Dalam proses penyusunan skripsi ini, penulis mendapatkan banyak sekali bantuan, bimbingan, serta dukungan dari berbagai pihak. Sehingga dalam kesempatan ini, penulis juga bermaksud untuk menyampaikan rasa terima kasih kepada:

1. Allah SWT yang telah melancarkan dan selalu memberi kemudahan selama proses penulisan skripsi ini.
2. Kedua orang tua, adik, dan segenap keluarga besar saya yang selalu memberikan dukungan dan doa selama saya menempuh pendidikan di Universitas Brawijaya.
3. Bapak Eko Sakti Pramukantoro, S.Kom, M.Kom. selaku dosen pembimbing I skripsi yang telah membimbing dan memberikan masukan serta waktunya agar penulis dapat menyelesaikan skripsi ini.
4. Bapak Rakhmadhany Primananda, S.T, M.Kom. selaku dosen pembimbing II yang juga telah membimbing dan memberikan masukan serta waktunya agar penulis dapat menyelesaikan skripsi ini.
5. Bapak Wayan Firdaus Mahmudy , S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer.
6. Bapak Tri Astoto Kurniawan , S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika.
7. Bapak Agus Wahyu Widodo, S.T., M.Cs. selaku Ketua Program Studi Teknik Informatika.
8. Teman-teman sepermainan yang tidak dapat saya sebutkan namanya satu per satu yang telah menemani dan saling mendukung semasa perkuliahan di Universitas Brawijaya, termasuk pada saat pengerjaan skripsi ini.
9. Teman-teman bimbingan skripsi pak Eko yang selalu saling membantu, saling mendukung, dan saling berbagi informasi dalam pengerjaan skripsi ini.
10. Teman-teman Teknik Informatika angkatan 2015 yang selalu saling mendukung dan berbagi informasi selama masa perkuliahan.
11. Seluruh civitas akademik Fakultas Ilmu Komputer yang selalu memberikan bantuan selama masa perkuliahan.

Penulis menyadari bahwa masih banyak sekali kekurangan dalam skripsi yang disusun penulis sehingga penulis sangat mengharapkan kritik dan saran yang membangun dari pembaca. Semoga skripsi ini dapat bermanfaat bagi pembaca dan dapat memberikan inspirasi pada penelitian berikutnya.

Malang, 03 Februari 2020

Fadila Rafi Alifiandi

fadilarafia@student.ub.ac.id

# ABSTRAK

**Fadila Rafi Alifiandi, Penerapan Mekanisme Transmisi Data ECG Berbasis LoRa (*Long Range*)**

**Pembimbing: Eko Sakti Pramukantoro, S.Kom, M.Kom dan Rakhmadhany Primananda, S.T, M.Kom**

Pada tahun 2016, tercatat penyakit kardiovaskular masih menjadi penyebab kematian nomor 1 paling banyak di dunia. Hal tersebut terjadi karena deteksi penyakit kardiovaskular yang sering terlambat. Saat ini, telah dikembangkan arsitektur bernama *Internet of Things* (IoT) yang memungkinkan terhubungnya benda apapun dengan siapapun kapanpun dan dimanapun. Sehingga IoT dinilai dapat menjadi solusi untuk masalah tersebut apabila diterapkan pada sistem pemantauan *electrocardiogram* (ECG). Saat ini, juga telah dikembangkan sistem pemantauan ECG dengan arsitektur IoT. Sistem tersebut terdiri dari *node* sensor ECG yang menggunakan WiFi untuk mengirimkan data yang diperoleh menuju ke penyimpanan data pada *cloud*. Namun, WiFi dinilai kurang tepat untuk digunakan pada pengaplikasian IoT karena besarnya *frame overhead* yang digunakan dan kebutuhan konsumsi dayanya yang tinggi. Oleh karena itu, dalam penelitian ini dirancang sebuah mekanisme transmisi data ECG berbasis teknologi LoRa. LoRa merupakan salah satu dari media pengiriman data yang memiliki *frame overhead* yang kecil hanya terdiri dari *preamble*, *header* (opsional), dan CRC (opsional) dan hemat daya karena tergolong protokol *Low-Power Wide-Area Network* (LPWAN) sehingga dinilai tepat untuk digunakan pada lingkungan IoT. Namun untuk pengiriman data menuju ke *cloud*, diperlukan juga sebuah *gateway* karena LoRa menggunakan protokol *non-IP-based* yang membuatnya tidak dapat digunakan pada koneksi ke internet. Oleh karena itu, dirancang juga sebuah LoRa *gateway* untuk meneruskan data dari *node* sensor ke *cloud.* Penelitian ini menghasilkan sebuah mekanisme dimana *node* sensor mampu merekam dan mengirimkan data ECG ke LoRa *gateway* dan LoRa *gateway* mampu meneruskan data ECG yang diterima ke aplikasi pada *cloud*. Namun, penerapan mekanisme tersebut menghasilkan rata-rata *end-to-end delay* 25,5 detik untuk pengiriman hasil satu siklus perekaman data ECG dari *node* sensor ke aplikasi. Sehingga, mekanisme pengiriman data ECG yang diterapkan pada penelitian ini masih belum memenuhi salah satu kebutuhan pada pengiriman data ECG untuk keperluan pemantauan yang bersifat *non-critical*, yaitu *delay* pengiriman maksimum 4 detik.

Kata kunci: LoRa, ECG, *Internet of Things*

# ABSTRACT

**Fadila Rafi Alifiandi, LoRa-based ECG Data Transmission Mechanism Application**

**Supervisors: Eko Sakti Pramukantoro, S.Kom, M.Kom dan Rakhmadhany Primananda, S.T, M.Kom**

*In 2016, cardiovascular disease was the number 1 cause of death in the world. It was happened because cardiovascular disease was often detected late. At this moment, an architecture called Internet of Things (IoT) has been developed to allows humans and things to be connected anytime and anywhere. Therefore, IoT can be considered as a solution to that problem if it is applied on electrocardiogram (ECG) monitoring system. At this moment, an ECG monitoring system with IoT architecture has been developed too. That system consists of ECG sensor nodes which uses WiFi to transmit the sensor data to the cloud as data storagae. However, WiFi is considered unsuitable to be used in IoT applications because of its large overhead and high power consumption. Therefore, a mechanism for sending ECG data using LoRa technology is designed in this research. LoRa is one of the data transmission media that has a small frame overhead consisting only of preamble, header (optional), and CRC (optional) and low power consumption because it is classified as the Low-Power Wide-Area Network (LPWAN) protocol. That features make LoRa considered suitable to be used in an IoT environment. But to send the data to the cloud, a gateway is needed because LoRa uses a non-IP-based protocol that makes it can't be used in internet connection. Therefore, a LoRa gateway that forwards data from sensor nodes to the cloud is also designed. This research produces a mechanism where sensor nodes are able to record and send ECG data to the LoRa gateway and LoRa gateway is able to forward received ECG data to the application on the cloud. But, the applied ECG data transmission mechanism in this research resulted 25,5 seconds end-to-end delay. So, it didn't fulfill one of the requirements in ECG data transmission for non-critical monitoring uses, the unfulfilled requirement is 4 seconds maximum delay.*

Keywords: *LoRa, ECG, Internet of Things*

# DAFTAR ISI

PENGESAHAN ..... ii
PERNYATAAN ORISINALITAS ..... iii
PRAKATA ..... iv
ABSTRAK ..... vi
ABSTRACT ..... vii
DAFTAR ISI ..... viii
DAFTAR TABEL ..... xi
DAFTAR GAMBAR ..... xiii
BAB 1 PENDAHULUAN ..... 1
1.1 Latar Belakang ..... 1
1.2 Rumusan Masalah ..... 2
1.3 Tujuan ..... 3
1.4 Manfaat ..... 3
1.5 Batasan Masalah ..... 3
1.6 Sistematika Pembahasan ..... 4
BAB 2 LANDASAN KEPUSTAKAAN ..... 5
2.1 Kajian Pustaka ..... 5
2.2 Dasar Teori ..... 7
2.2.1 Internet of Things (IoT) ..... 7
2.2.2 *Node* Sensor ..... 9
2.2.3 *Gateway* pada IoT ..... 9
2.2.4 LoRa ..... 10
2.2.5 *Electrocardiogram* ..... 10
2.2.6 WEMOS LOLIN D32 (ESP32) ..... 12
2.2.7 Raspberry Pi 3 Model B+ ..... 14
BAB 3 METODOLOGI PENELITIAN ..... 16
3.1 Studi Literatur ..... 16
3.2 Perancangan ..... 17
3.2.1 Deskripsi Umum Sistem ..... 17
3.2.2 Lingkungan Penelitian ..... 17

3.2.3 Perancangan Arsitektur Jaringan ..... 17
3.2.4 Perancangan *Node* Sensor ..... 17
3.2.5 Perancangan LoRa *Gateway* ..... 18
3.2.6 Perancangan Pengujian ..... 18
3.3 Implementasi ..... 18
3.4 Pengujian dan Pembahasan ..... 18
3.5 Penarikan Kesimpulan ..... 19
BAB 4 PERANCANGAN ..... 20
4.1 Deskripsi Umum Sistem ..... 20
4.2 Lingkungan Penelitian ..... 20
4.2.1 Lingkungan Perangkat Keras ..... 21
4.2.2 Lingkungan Perangkat Lunak ..... 21
4.3 Perancangan Arsitektur Jaringan ..... 22
4.4 Perancangan *Node* Sensor ..... 23
4.4.1 Perancangan Perangkat Keras *Node* Sensor ..... 23
4.4.2 Perancangan Fungsionalitas *Node* Sensor ..... 25
4.5 Perancangan LoRa *Gateway* ..... 26
4.5.1 Perancangan Perangkat Keras LoRa *Gateway* ..... 26
4.5.2 Perancangan Arsitektur LoRa *gateway* ..... 27
4.5.3 Perancangan Fungsionalitas LoRa *Gateway* ..... 28
4.6 Perancangan Pengujian ..... 31
4.6.1 Perancangan Pengujian Fungsionalitas Sistem ..... 31
4.6.2 Perancangan Pengujian Kinerja ..... 33
BAB 5 IMPLEMENTASI ..... 36
5.1 Implementasi *Node* Sensor ..... 36
5.1.1 Implementasi Fungsi Perekaman Data ECG dari Pengguna ..... 37
5.1.2 Implementasi Fungsi Pengiriman Data ECG ke LoRa *Gateway* ..... 38
5.2 Implementasi LoRa *gateway* ..... 39
5.2.1 Implementasi Fungsi Autentikasi *Token* ke Aplikasi ..... 41
5.2.2 Implementasi Fungsi Penerimaan Paket Data ECG ..... 42

5.2.3 Implementasi Fungsi Penerusan Data ECG ke Aplikasi ..... 42
BAB 6 PENGUJIAN DAN PEMBAHASAN ..... 44
6.1 Pengujian Fungsionalitas Sistem ..... 44
6.1.1 Pengujian Kemampuan *Node* Sensor Merekam Data ECG dari Pengguna ..... 44
6.1.2 Pengujian Kemampuan *Node* Sensor Membungkus Data ECG dengan Topik dan ID Sensor dalam Format JSON ..... 46
6.1.3 Pengujian Kemampuan *Node* Sensor Mengirim Data ECG Terformat ke LoRa *Gateway* Melalui LoRa ..... 48
6.1.4 Pengujian Kemampuan LoRa *Gateway* Menerima Data ECG Terformat yang Dikirim *Node* Sensor Melalui LoRa ..... 50
6.1.5 Pengujian Kemampuan LoRa *Gateway* Mengautentikasikan *Token*-nya ke Aplikasi ..... 51
6.1.6 Pengujian Kemampuan LoRa *Gateway* Meneruskan Data ECG Beserta ID Sensor Sesuai dengan Topiknya Menuju ke Aplikasi ..... 54
6.2 Pengujian Kinerja Sistem ..... 57
6.2.1 Pengujian *Delay* Pengiriman Data ECG dari *Node* Sensor ke LoRa *Gateway* Melalui LoRa ..... 57
6.2.2 Pengujian *End-to-End Delay* Pengiriman Data ECG dari *Node* Sensor Sampai ke Aplikasi pada *Cloud* ..... 59
6.2.3 Pengujian *End-to-End Delay* Pengiriman Hasil Satu Siklus Perekaman Data ECG dari *Node* Sensor sampai ke Aplikasi ..... 61
6.2.4 Pengujian *Packet Delivery Ratio* yang Dikirimkan oleh *Node* Sensor melalui LoRa ..... 62
BAB 7 KESIMPULAN DAN SARAN ..... 66
7.1 Kesimpulan ..... 66
7.2 Saran ..... 67
DAFTAR REFERENSI ..... 69

# DAFTAR TABEL

Tabel 2.1 Kajian pustaka ..... 5
Tabel 2.2 Spesifikasi WEMOS LOLIN D32 ..... 12
Tabel 2.3 Spesifikasi ESP32-WROOM-32 ..... 13
Tabel 2.4 Spesifikasi Raspberry Pi 3 model B+ ..... 15
Tabel 4.1 Lingkungan perangkat keras ..... 21
Tabel 4.2 Lingkungan perangkat lunak ..... 21
Tabel 4.3 *Pinout* ESP32 dengan AD8232 ..... 24
Tabel 4.4 *Pinout* ESP32 dengan RFM95w ..... 24
Tabel 4.5 *Pinout* Raspberry Pi dengan RFM95w ..... 26
Tabel 4.6 Skenario pengujian fungsionalitas sistem ..... 31
Tabel 4.7 Skenario pengujian kinerja ..... 34
Tabel 5.1 *Pseudocode node* sensor ..... 37
Tabel 5.2 *Pseudocode* fungsi rekamData ..... 37
Tabel 5.3 *Pseudocode* fungsi kirimData ..... 38
Tabel 5.4 *Pseudocode* LoRa *gateway* ..... 39
Tabel 5.5 *Pseudocode* fungsi login ..... 41
Tabel 5.6 *Pseudocode* fungsi getData ..... 42
Tabel 5.7 *Pseudocode* fungsi fwdToApps ..... 42
Tabel 6.1 Pengujian kemampuan *node* sensor merekam data ECG dari pengguna ..... 44
Tabel 6.2 Pengujian kemampuan *node* sensor membungkus data ECG dengan topik dan id sensor dalam format JSON ..... 46
Tabel 6.3 Pengujian kemampuan *node* sensor mengirim data ECG terformat ke LoRa *gateway* melalui LoRa ..... 48
Tabel 6.4 Pengujian kemampuan LoRa *gateway* menerima data ECG terformat yang dikirim *node* sensor melalui LoRa ..... 51
Tabel 6.5 Pengujian kemampuan LoRa *gateway* mengautentikasikan *token*-nya ke aplikasi ..... 52
Tabel 6.6 Pengujian kemampuan LoRa *gateway* meneruskan data ECG beserta id sensor sesuai dengan topiknya menuju ke aplikasi ..... 54
Tabel 6.7 Pengujian *delay* pengiriman data ECG dari *node* sensor ke LoRa *gateway* melalui LoRa ..... 57

Tabel 6.8 Hasil pengujian *delay* pengiriman data melalui LoRa..... 58
Tabel 6.9 Pengujian *end-to-end delay* pengiriman data ECG dari *node* sensor sampai ke aplikasi *cloud*..... 59
Tabel 6.10 Hasil pengujian *end-to-end delay*..... 60
Tabel 6.11 Pengujian *end-to-end delay* pengiriman hasil satu siklus perekaman data ECG dari *node* sensor sampai ke aplikasi ..... 62
Tabel 6.12 Hasil pengujian *end-to-end delay* pengiriman hasil satu siklus perekaman data ECG ..... 62
Tabel 6.13 Pengujian *Packet Delivery Ratio* yang dikirimkan *node* sensor melalui LoRa..... 63
Tabel 6.14 Hasil pengujian *Packet Delivery Ratio* ..... 63

# DAFTAR GAMBAR


Gambar 2.1 Arsitektur IoT ..... 8
Gambar 2.2 Arsitektur *node* sensor ..... 9
Gambar 2.3 Contoh grafik ECG ..... 10
Gambar 2.4 Pola dasar aktifitas listrik pada ECG ..... 11
Gambar 2.5 WEMOS LOLIN D32 ..... 12
Gambar 2.6 Raspberry Pi 3 model b+ ..... 14
Gambar 3.1 Metodologi penelitian ..... 16
Gambar 4.1 Gambaran umum sistem ..... 20
Gambar 4.2 Arsitektur jaringan pada sistem ..... 22
Gambar 4.3 *Layout* pin ESP32 ..... 23
Gambar 4.4 *Layout* pin RFM95w ..... 23
Gambar 4.5 *Layout* pin AD8232 ..... 24
Gambar 4.6 Diagram alir *node* sensor ..... 25
Gambar 4.7 Format pembungkusan data dalam JSON ..... 26
Gambar 4.8 *Layout* pin Raspberry Pi ..... 27
Gambar 4.9 Arsitektur LoRa *gateway* ..... 27
Gambar 4.10 *Sequence diagram* LoRa *gateway* ..... 28
Gambar 4.11 *Flow diagram* fungsionalitas LoRa *gateway* ..... 29
Gambar 4.12 Fungsionalitas pemrosesan paket LoRa pada *thread* ..... 30
Gambar 5.1 Implementasi perangkat keras *node* sensor ..... 36
Gambar 5.2 Implementasi perangkat keras LoRa *gateway* ..... 39
Gambar 6.1 Pemasangan elektroda pada badan pengguna ..... 45
Gambar 6.2 Tampilan *node* sensor pada laptop ..... 45
Gambar 6.3 Tampilan *node* sensor saat perekaman data dilakukan ..... 45
Gambar 6.4 Isi *file* dataECG.txt ..... 46
Gambar 6.5 Tampilan *node* sensor pada laptop ..... 47
Gambar 6.6 *File* yang berisi data hasil rekam ECG pengguna ..... 47
Gambar 6.7 Hasil pembungkusan data ECG pengguna ..... 48
Gambar 6.8 Tampilan *node* sensor pada laptop ..... 49
Gambar 6.9 Tampilan LoRa *gateway* pada SSH laptop ..... 49

Gambar 6.10 Data JSON yang berisi data ECG, topik, dan id sensor ..... 49
Gambar 6.11 Tampilan pada LoRa *gateway* saat program penerimaan berjalan 50
Gambar 6.12 Tampilan pada *node* sensor saat program pengiriman berjalan .... 50
Gambar 6.13 Data yang diterima pada LoRa *gateway* ..... 50
Gambar 6.14 Tampilan LoRa *gateway* pada SSH laptop ..... 52
Gambar 6.15 *Token* dan *secret key* pada LoRa *gateway* ..... 53
Gambar 6.16 *Token* dan *secret key* pada aplikasi ..... 53
Gambar 6.17 Tampilan pada LoRa *gateway* saat autentikasi dilakukan ..... 53
Gambar 6.18 Tampilan pesan berhasil dan *token* terautentikasi ..... 54
Gambar 6.19 Tampilan LoRa *gateway* pada tampilan SSH laptop ..... 55
Gambar 6.20 Data dan *token* yang akan dikirimkan oleh LoRa *gateway* ..... 55
Gambar 6.21 Pesan bahwa data berhasil diteruskan ke aplikasi ..... 56
Gambar 6.22 Tampilan iotapps.belajardisini.com pada topik 'ecg' ..... 56
Gambar 6.23 Tampilan detil data pada iotapps.belajardisini.com ..... 57
Gambar 6.24 Grafik *delay* pengiriman data dari *node* sensor ke LoRa *gateway* .. 59
Gambar 6.25 Grafik *end-to-end delay* dari *node* sensor ke aplikasi ..... 61
Gambar 6.26 Grafik *Packet Delivery Ratio* (PDR) ..... 64

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Pada tahun 2016, tercatat penyakit kardiovaskular masih menjadi penyebab kematian nomor 1 paling banyak di dunia. Penyebab tingginya kematian akibat penyakit kardiovaskular adalah deteksi yang sering terlambat terutama di negara berkembang (World Heart Federation, 2016). Hal tersebut terjadi karena masih kurangnya tenaga ahli pada negara-negara berkembang. Pendeteksian penyakit kardiovaskular dapat dilakukan dengan pemeriksaan *electrocardiogram* (ECG). ECG merupakan sebuah pemeriksaan yang mendeteksi aktifitas kelistrikan di dalam jantung dengan menggunakan beberapa sensor yang ditempelkan pada kulit di sekitar dada pasien (Curry et al., 2018). Saat ini, ECG masih sering diperiksa secara konvensional dimana pemeriksaan dilakukan secara langsung di rumah sakit oleh tenaga ahli. Hal ini tentu menimbulkan risiko keterlambatan penanganan apabila sewaktu-waktu timbul gejala penyakit kardiovaskular pada pasien di saat tenaga ahli sedang tidak tersedia. Oleh karena itu, pasien membutuhkan teknologi yang dapat digunakan untuk memeriksa kondisi kardiovaskularnya dimana hasil dari pemeriksaan tersebut dapat dipantau kapanpun dan darimanapun dengan mudah. Saat ini, telah dikembangkan arsitektur bernama *Internet of Things* (IoT) yang memungkinkan terhubungnya benda apapun dengan siapapun kapanpun dan dimanapun (Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, 2014). Sehingga, *Internet of Things* (IoT) dinilai dapat memenuhi kebutuhan tersebut (Nogueira dan Carnaz, 2019).

Secara umum, persamaan yang menggambarkan lingkungan IoT adalah "lingkungan IoT = Internet + *Wireless Sensor Network* (WSN)" (Ali, Ali dan Badawy, 2015). WSN merupakan jaringan yang terdiri dari beberapa *node* sensor data yang memiliki infrastruktur komunikasi untuk memantau kejadian-kejadian atau kondisi dari suatu objek. Satu *node* sensor pada WSN terdiri dari sebuah antarmuka sensor, unit pemrosesan, *transceiver*, dan sumber daya listrik. *Node* sensor pada penelitian ini digunakan untuk merekam data ECG pada pasien. Sedangkan internet merupakan layanan yang digunakan untuk mengakses *platform* yang dapat menyediakan layanan penyimpanan data dan pengaksesan data darimana saja kapan saja, yaitu *cloud*.

Salah satu penelitian yang menerapkan arsitektur IoT pada pemantauan ECG adalah penelitian sebelumnya yang dilakukan oleh Yang et al. (2016). Sistem pada penelitian tersebut menggunakan *node* pemantau dengan tiga elektroda dan media pengiriman WiFi untuk mengirimkan data yang diperoleh menuju ke *cloud* dimana data ECG akan disimpan dan dianalisis nantinya. Pada penelitian tersebut juga diimplementasikan protokol MQTT untuk pengiriman data dari *node* sensor menuju ke *cloud* dan protokol HTTP untuk pengaksesan data dari *cloud*. Namun, menurut Salman dan Jain (2017), media transmisi WiFi dinilai kurang tepat untuk digunakan pada pengaplikasian IoT karena besarnya *frame overhead* yang digunakan dan kebutuhan konsumsi dayanya yang tinggi. Sehingga diperlukan

teknologi media transmisi yang lebih tepat untuk menerapkan arsitektur IoT pada pemantauan ECG.

Saat ini, telah dikembangkan teknologi yang tergolong baru yang sudah mengatasi keterbatasan-keterbatasan tersebut. Teknologi tersebut adalah LoRa (*Long Range*). LoRa merupakan salah satu dari media pengiriman data yang memiliki *frame overhead* yang kecil hanya terdiri dari *preamble*, *header* (opsional), dan CRC (opsional) dan hemat daya karena tergolong protokol *Low-Power Wide-Area Network* (LPWAN) (Yi et al., 2016). Teknologi LoRa juga tahan terhadap adanya interferensi dari sinyal lain dan juga tahan terhadap *multipath*/*fading* (Sonnenberg, 2019; Semtech, 2015). Teknologi ini merupakan teknologi yang dikembangkan untuk mendukung komunikasi jarak jauh yang hemat daya yang diklaim oleh Semtech (2018) dapat menjangkau perangkat yang sampai dengan 30 mil atau 48 km jauhnya.

Apabila dibandingkan, WiFi menggunakan *frame overhead* berukuran 34 bytes dan mengonsumsi daya sebesar 400 mA saat transmisi dan 20 mA saat *standby*, sedangkan LoRa menggunakan *frame overhead* yang hanya berukuran 2-7 bytes dan mengonsumsi daya sebesar 28-44 mA saat transmisi dan 1,4 mA saat *standby* (Sarkar dan Member, 2011; Islam, Islam dan Almutairi, 2019; Yi et al., 2016). Berdasarkan perbandingan tersebut, LoRa merupakan teknologi yang dinilai lebih tepat untuk digunakan sebagai media pengiriman pada pengaplikasian arsitektur IoT dibandingkan dengan WiFi. Namun untuk pengiriman data menuju ke aplikasi *cloud*, dibutuhkan sebuah *gateway* karena LoRa menggunakan protokol *non-IP-based* yang membuatnya tidak dapat digunakan untuk koneksi ke internet. Oleh karena itu, dirancang juga sebuah LoRa *gateway* yang berfungsi untuk menerima data dari *node* sensor dan meneruskannya sampai ke *cloud*. LoRa *gateway* terdiri dari antarmuka LoRa untuk menerima data ECG dari *node* sensor dan antarmuka internet untuk meneruskan data yang diterima ke aplikasi pada *cloud*.

Berdasarkan latar belakang yang telah dipaparkan sebelumnya pada subbab ini, dirancang sebuah mekanisme transmisi data ECG dari *node* sensor yang berupa ESP32 ke LoRa *gateway* yang berupa Raspberry Pi dan selanjutnya diteruskan ke aplikasi. Aplikasi yang digunakan adalah aplikasi pada *cloud* yang sudah dikembangkan dalam penelitian (Pramukantoro et al., 2017). Media yang digunakan untuk transmisi data dari *node* sensor ke LoRa *gateway* adalah LoRa, sedangkan media yang digunakan untuk transmisi data dari LoRa *gateway* ke aplikasi adalah WiFi. Data yang dikirimkan terdiri dari topik, ID dari *node* sensor pengirim dan data hasil rekaman ECG. Penelitian ini dilakukan untuk mendukung layanan pemantauan ECG pengguna kapanpun yang hasilnya dapat diakses darimanapun dengan penggunaan sumber daya yang hemat sehingga dapat membantu mengurangi angka kematian yang diakibatkan oleh terlambatnya mendeteksi penyakit kardiovaskular.

## 1.2 Rumusan Masalah

Dari latar belakang yang telah dikemukakan, dapat dirumuskan tiga permasalahan sebagai berikut:

1. Bagaimanakah format data ECG yang ditransmisikan oleh *node* sensor?
2. Bagaimanakah mekanisme transmisi data ECG dari *node* sensor ke LoRa *gateway* via media transmisi LoRa lalu ke aplikasi pada *cloud*?
3. Bagaimanakah kinerja dari transmisi data ECG dari *node* sensor ke LoRa *gateway* via media transmisi LoRa lalu ke aplikasi pada *cloud*?

## 1.3 Tujuan

Tujuan dari penelitian ini adalah menerapkan mekanisme transmisi data ECG dari *node* sensor ke LoRa *gateway* menggunakan LoRa, yang dinilai lebih tepat dibanding penggunaan WiFi pada penelitian sebelumnya, lalu diteruskan ke aplikasi pada *cloud* dengan tetap menggunakan WiFi.

## 1.4 Manfaat

Penelitian ini merupakan penelitian yang difokuskan pada transmisi data ECG dari node sensor ke LoRa *gateway* menggunakan media transmisi LoRa. Sehingga penelitian ini termasuk kedalam pengembangan IoT di sektor kesehatan. Diharapkan penelitian ini dapat menjadi satu dari banyak langkah kedepan dalam pengembangan arsitektur IoT di sektor kesehatan dan dapat menjadi dasar atau hanya sekedar menjadi gambaran untuk penelitian-penelitian selanjutnya yang terkait. Penelitian ini juga diharapkan dapat menjadi pembanding dari penelitian-penelitian lainnya yang berhubungan dengan transmisi data pada IoT di sektor kesehatan baik menggunakan teknologi yang baru atau teknologi yang sudah dibuat. Terlebih lagi, diharapkan penelitian ini dapat memberikan dampak positif untuk dunia kesehatan.

## 1.5 Batasan Masalah

Agar penelitian ini dapat dilakukan dengan ruang lingkup yang jelas dan spesifik, perlu didefinisikan batasan masalah sebagai berikut:

1. Mikrokontroler yang digunakan pada *node* sensor adalah ESP32 dan perangkat yang digunakan sebagai LoRa *gateway* adalah Raspberry Pi.
2. Data yang dikumpulkan dan ditransmisikan oleh *node* sensor adalah data ECG.
3. Transmisi data dari *node* sensor ke LoRa *gateway* menggunakan media transmisi LoRa. Sedangkan, transmisi data dari LoRa *gateway* ke aplikasi menggunakan media transmisi WiFi.
4. Pemrograman pada *node* sensor dan LoRa *gateway* menggunakan bahasa pemrograman Python.
5. Pengujian yang dilakukan hanya difokuskan pada transmisi data yang melibatkan media transmisi LoRa karena fokus dari penelitian ini adalah penggunaan LoRa.
6. Dikarenakan fokus penelitian ini hanyalah pada transmisi datanya, maka aspek keamanan tidak dibahas di dalam penelitian ini.

## 1.6 Sistematika Pembahasan

Struktur penulisan skripsi ini adalah sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini terdiri dari latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan dari penelitian ini.

**BAB II LANDASAN KEPUSTAKAAN**

Bab ini terdiri dari kajian pustaka yang berisi penelitian-penelitian yang terkait yang telah dilakukan sebelumnya sebagai penguat untuk berjalannya penelitian ini dan dasar teori yang berisi teori/konsep yang relevan dengan penelitian ini.

**BAB III METODOLOGI PENELITIAN**

Bab ini berisi metode sistematik untuk menyelesaikan masalah pada penelitian ini dan juga logika dibalik mengapa metode tersebut yang dipilih untuk menyelesaikan masalah.

**BAB IV PERANCANGAN**

Bab ini berisi seluruh perancangan sistem agar dapat berjalan dengan seharusnya. Perancangan tersebut terdiri dari deskripsi umum sistem, lingkungan penelitian, perancangan *node-node* pada sistem, dan perancangan pengujian..

**BAB V IMPLEMENTASI**

Bab ini berisi implementasi sistem yang dilakukan dengan mengacu pada perancangan.

**BAB VI PENGUJIAN DAN PEMBAHASAN**

Bab ini berisi pengujian-pengujian yang dilakukan dan pembahasan dari hasil pengujian tersebut untuk mengetahui bagaimana kinerja dari hasil implementasi sistem.

**BAB VII KESIMPULAN DAN SARAN**

Bab ini terdiri dari kesimpulan dari penelitian yang telah dilakukan dan saran yang dapat dijadikan dasar untuk melakukan penelitian selanjutnya.

# BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menguraikan hasil dari studi literatur yang dilakukan berupa kajian pustaka yang menjelaskan secara umum penelitian-penelitian terdahulu yang berhubungan dengan topik penelitian yang diangkat dan menunjukkan persamaan dan perbedaan skripsi tersebut terhadap penelitian terdahulu yang dituliskan serta dasar teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian.

## 2.1 Kajian Pustaka

Subbab ini menguraikan penelitian-penelitian terdahulu yang berkaitan dengan penelitian yang dilakukan. Rangkuman persamaan dan perbedaan dari penelitian-penelitian tersebut tercantum pada Tabel 2.1.

**Tabel 2.1 Kajian pustaka**

| No | Nama Penulis, Tahun, Judul | Persamaan | Perbedaan | |
|---|---|---|---|---|
| | | | Sebelum | Sekarang |
| 1 | Yang, et al., 2016, "*An IoT-cloud Based Wearable ECG Monitoring System for Smart Healthcare*" | Penelitian menggunakan *cloud* sebagai penyimpanan datanya dan data yang direkam oleh sensor adalah data ECG | Media transmisi yang digunakan oleh *node* sensor adalah WiFi | Media transmisi yang digunakan oleh *node* sensor adalah LoRa |
| 2 | Pramukantoro et al., 2019, "*Bridging IoT Infrastructure and Cloud Application using Cellular-based Internet Gateway Device*" | Penelitian yang dilakukan menggunakan sebuah *internet gateway* untuk meneruskan data ke *cloud* | *Internet gateway* yang digunakan untuk meneruskan data ke *cloud* merupakan sebuah perangkat tersendiri | *Internet gateway* yang digunakan untuk meneruskan data ke *cloud* tergabung dengan LoRa *gateway* |
| 3 | Pramukantoro, et al., 2017, "*Topic Based IoT Data Storage Framework for Heterogeneous Sensor Data*" | Penelitian yang dilakukan menggunakan *cloud* sebagai penyimpanan datanya | Data yang disimpan adalah hasil *sensing* kamera dan sensor CO | Data yang disimpan adalah data ECG |
| 4 | Islam, et al., 2019, "*Monitoring of the Human Body Signal through the Internet of Things* | Penelitian tersebut menggunakan LoRa sebagai media transmisi | Data yang dikirimkan dari *node* sensor-nya adalah data yang sudah diproses | Data yang dikirimkan dari *node* sensor adalah data ECG *raw*. Dan fokus penelitian ini |

| No | Nama Penulis, Tahun, Judul | Persamaan | Perbedaan | |
|---|---|---|---|---|
| | | | Sebelum | Sekarang |
| | *(IoT) Based LoRa Wireless Network System*" | dan pemantauan kondisi jantung pasien merupakan salah satu fungsi yang disediakan oleh sistem yang dibuat pada penelitian tersebut. | menjadi informasi detak jantung per menit. Dan fokus penelitian tersebut lebih luas, yaitu pemantauan suhu tubuh, saturasi oksigen pada tubuh, denyut nadi, dan detak jantung pengguna dengan LoRa sebagai media transmisi datanya. | lebih spesifik kepada pemantauan data ECG pengguna dengan LoRa sebagai media transmisi datanya. |

Penelitian yang pertama adalah penelitian dengan judul "***An IoT-cloud Based Wearable ECG Monitoring System for Smart Healthcare***" oleh Yang et al. (2016). Pada penelitian tersebut, diajukan arsitektur sistem pemantauan ECG yang berbasis IoT *cloud.* Berdasarkan arsitekturnya, data ECG yang diperoleh dari badan pengguna akan ditransmisikan secara langsung ke IoT *cloud* dengan media transmisi WiFi. IoT *cloud* tersebut berfungsi untuk penyimpanan data ECG untuk analisis lebih lanjut dan menyediakan akses untuk pengguna, dengan mengimplementasikan *server* HTTP, *server* MQTT, dan *server* penyimpanan. Akses yang disediakan adalah pengaksesan menggunakan web. Namun, kelemahan dari sistem yang dibuat pada penelitian tersebut terletak pada penggunaan media transmisi WiFi yang memiliki *frame overhead* yang besar dan konsumsi daya yang tinggi. Untuk mengatasi kelemahan yang ditemukan pada penelitian tersebut, penelitian ini menggunakan media transmisi LoRa untuk pengiriman data dari *node* sensor ke *gateway* agar *node* sensor dapat lebih menghemat penggunaan daya.

Penelitian kedua adalah penelitian dengan judul "***Bridging IoT Infrastructure and Cloud Application using Cellular-based Internet Gateway Device***" oleh Pramukantoro, Luckies dan Bakhtiar (2019). Pada penelitian tersebut, dibangun sebuah *Internet Gateway Device* (IGD) yang menerima data dari *IoT middleware* yang menggunakan pendekatan *publish-subscribe* dan menggunakan GPRS untuk mendapatkan koneksi internet sehingga data dari *middleware* yang berada di intranet dapat dikirimkan menuju ke *cloud* yang berada di internet. IGD yang dibangun memiliki dua fungsi, yaitu mengambil data dari *middleware* dan mengirimkan data menuju ke *cloud*. IGD yang dibangun pada penelitian tersebut dijadikan acuan untuk mengimplementasikan fungsionalitas penerusan data ke aplikasi pada penelitian ini. Namun pada penelitian ini, konektifitas ke internet diperoleh melalui media transmisi WiFi.

Penelitian ketiga adalah penelitian dengan judul "***Topic Based IoT Data Storage Framework for Heterogeneous Sensor Data***" oleh Pramukantoro et al.

(2017). Pada penelitian tersebut, diimplementasikan sebuah *framework* untuk penyimpanan data IoT dengan fungsi IGD, sebuah *Web Service*, NoSQL, dan aplikasi IoT. *Framework* tersebut mampu menyimpan dan mengatur data terstruktur maupun tidak terstruktur sehingga dapat mengatasi beberapa tantangan dalam membangun IoT *data storage*, seperti pertumbuhan data yang pesat, tipe data yang heterogen, dan format data yang bermacam-macam. *Cloud* yang dikembangkan pada penelitian tersebut digunakan dalam penelitian ini.

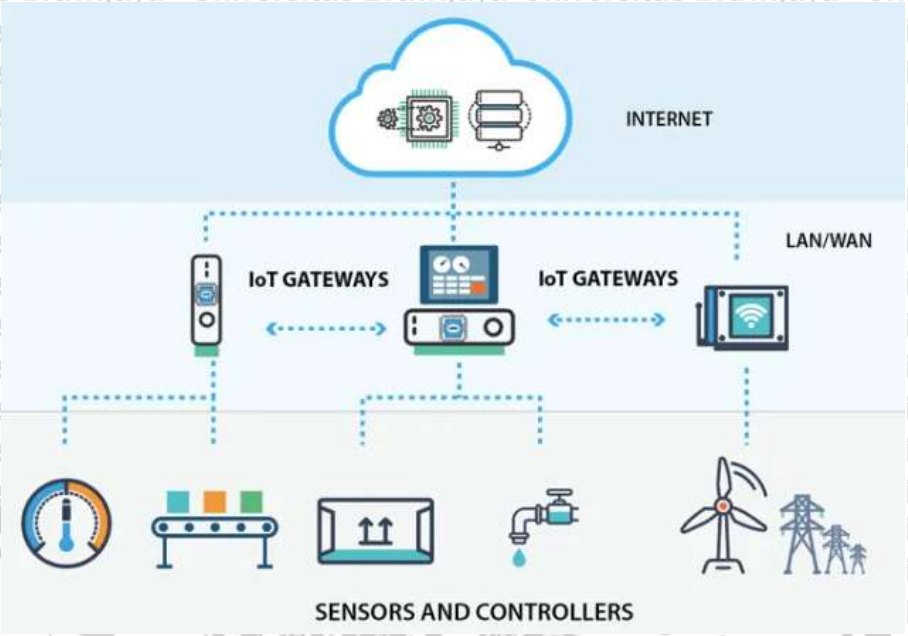
Penelitian keempat adalah penelitian dengan judul "***Monitoring of the Human Body Signal through the Internet of Things (IoT) Based LoRa Wireless Network System***" oleh Islam, Islam dan Almutairi (2019). Pada penelitian tersebut, dibangun sebuah sistem IoT yang memantau empat macam data fisiologis pengguna, yaitu: detak jantung, denyut nadi, saturasi oksigen, dan suhu tubuh dengan menggunakan media transmisi LoRa. Di dalam penelitian yang mereka lakukan, data ECG yang dikirimkan dari *node* sensor sudah diproses menjadi informasi frekuensi detak jantung dengan satuan *Beats Per Minute* (BPM). Sedangkan di dalam penelitian ini, data ECG yang dikirimkan dari *node* sensor masih *raw* atau masih berupa sinyal ECG yang direkam selama 10 detik dengan frekuensi 100 Hz. Sinyal ECG tidak diproses terlebih dahulu menjadi informasi frekuensi detak jantung dikarenakan data yang masih berupa sinyal ECG mengandung lebih banyak informasi yang dapat dianalisis daripada data yang sudah diproses menjadi informasi frekuensi detak jantung. Perbedaan lainnya adalah pemantauan ECG pada sistem di penelitian ini dapat dilakukan secara *remote*, sedangkan pada penelitian tersebut tidak. Penelitian tersebut dijadikan contoh untuk pengaplikasian IoT berbasis LoRa pada pemantauan data kesehatan.

## 2.2 Dasar Teori

Subbab ini berisi teori-teori yang digunakan atau dijadikan dasar dalam melakukan penelitian. Landasan teori dibuat agar penelitian dapat dilakukan dengan benar atau tidak menyimpang dari teori yang sudah ada.

### 2.2.1 Internet of Things (IoT)

*Internet of Things* (IoT) merupakan sebuah istilah yang mendeskripsikan sebuah sistem yang terdiri dari manusia, perangkat, dan layanan yang saling terhubung satu sama lain. Sehingga IoT memungkinkan terhubungnya benda apapun dengan siapapun kapanpun, dimanapun (Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, 2014). Sebuah penelitian yang dilakukan oleh Minerva, Biru dan Rotondi (2015) mendefinisikan bahwa penerapan IoT harus memiliki fitur-fitur seperti interkoneksi antar benda yang relevan, koneksi ke internet, identifikasi benda, ketersediaan kapanpun dimanapun, kemampuan merekam data sensorik, kemampuan melakukan aksi, kecerdasan, interoperabilitas, *self-configurability*, dan *programmability*. Sehingga, arsitektur yang dibutuhkan untuk membangun sistem IoT terdiri dari sensor/aktuator, *gateway*, dan aplikasi. Arsitektur IoT dapat dilihat pada Gambar 2.1.



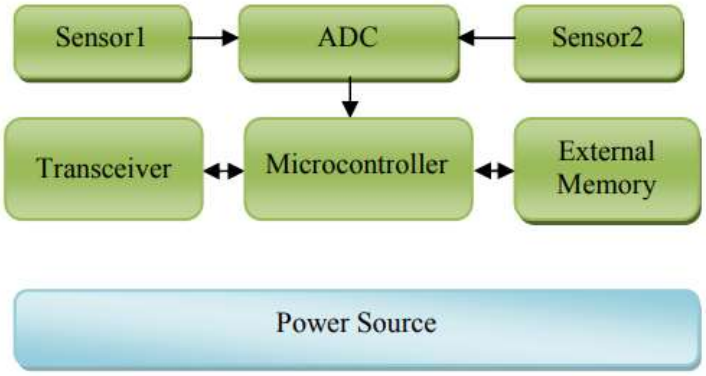

**Gambar 2.1 Arsitektur IoT**

Menurut Minerva, Biru dan Rotondi (2015), kompleksitas sistem dimana IoT diterapkan sangat mempengaruhi pandangan terhadap IoT. Bagi sistem dengan kompleksitas rendah, IoT hanya merupakan sebuah jaringan yang menghubungkan beberapa benda ke internet. Benda yang dihubungkan tersebut harus memiliki kemampuan untuk merasakan atau/dan melakukan sesuatu dan dapat diprogram. Lalu informasi yang diperoleh oleh benda dapat diakses darimanapun, kapanpun, dan menggunakan apapun. Sedangkan bagi sistem dengan kompleksitas tinggi, IoT menggambarkan jaringan kompleks yang adaptif dan dapat menkonfigurasikan dirinya sendiri untuk menghubungkan benda ke internet melalui penggunaan protokol komunikasi yang sesuai dengan standar. Benda yang terhubung memiliki representasi fisik, kemampuan merasakan dan melakukan sesuatu, dapat diprogram, dan dapat diidentifikasikan secara unik. Representasi fisik yang dimaksud mengandung informasi-informasi terkait benda yang direpresentasikan seperti identitas, status, lokasi, atau informasi lainnya yang relevan. Benda tersebut menyediakan layanan baik dengan atau tanpa campur tangan manusia lewat kemampuan identifikasi, perekaman data, komunikasi, dan aksi. Layanan tersebut dapat diakses lewat antarmuka yang tersedia dimanapun dan kapanpun dengan juga mempertimbangkan faktor keamanan.

Menurut Atzori, Iera dan Morabito (2010), IoT memiliki potensi untuk diaplikasikan pada berbagai macam sektor, namun saat ini masih sedikit yang mengaplikasikannya. Banyak domain dan lingkungan yang dapat ditingkatkan kualitasnya apabila diaplikasikan dengan arsitektur IoT. Beberapa dari domain tersebut adalah domain transportasi dan logistik, domain kesehatan, domain lingkungan cerdas, dan domain personal dan sosial.

### 2.2.2 *Node* Sensor

*Node* sensor merupakan sebuah *node* yang berfungsi untuk merekam data sensorik, melakukan pemrosesan yang diperlukan terhadap data tersebut, dan berkomunikasi dengan *node* lainnya dalam jaringan IoT. Menurut Karray et al. (2014), *node* sensor pada IoT dibangun berbasiskan *microcontroller* (MCU) dan hanya terdiri dari *transceiver* radio, MCU, memori eksternal, sensor, dan baterai. Arsitektur *node* sensor ditunjukkan pada Gambar 2.2.




**Gambar 2.2 Arsitektur *node* sensor**

Masing-masing komponen pada *node* sensor memiliki fungsi yang berbeda-beda. MCU berfungsi untuk menjalanan proses, memproses data, dan mengatur fungsionalitas-fungsionalitas dari *node* sensor. *Transceiver* radio berfungsi untuk berkomunikasi dan mentransmisikan data sensorik ke *node* lain. Memori eksternal berfungsi untuk menyimpan data-data terkait aplikasi dan menyimpan program yang akan dijalankan. Sensor berfungsi untuk merekam data sensorik dari lingkungan dimana *node* sensor diaplikasikan. Dan yang terakhir adalah baterai yang berfungsi sebagai penyedia sumber daya agar *node* sensor dapat berjalan.

### 2.2.3 *Gateway* pada IoT

Pada IoT, *gateway* merupakan sebuah perangkat yang menjembatani komunikasi antara domain sensorik dan domain aplikasi. Sehingga, *gateway* termasuk ke dalam salah satu komponen paling penting di dalam pengaplikasian konsep IoT (Chen, Jia dan Li, 2011). Dikarenakan perannya yang merupakan sebuah jembatan komunikasi, *gateway* pada IoT harus memiliki beberapa fitur utama. Fitur yang pertama adalah antarmuka komunikasi yang beragam. *Gateway* pada IoT membutuhkan antarmuka yang beragam dikarenakan domain sensorik dan domain aplikasi biasanya menggunakan media komunikasi yang berbeda untuk berkomunikasi, ditambah lagi dengan *node-node* pada domain sensorik yang juga belum tentu menggunakan media yang seragam untuk berkomunikasi. Lalu fitur yang kedua adalah kemampuan untuk mengkonversikan protokol. Sesuai dengan fitur pertama, penggunaan media komunikasi yang berbeda tentunya membutuhkan protokol yang berbeda juga. Sehingga sebagai pihak yang berperan untuk menjembatani komunikasi antara dua domain yang menggunakan media

komunikasi yang berbeda, *gateway* harus dapat mengkonversikan protokol yang digunakan pada kedua domain tersebut. Fitur utama yang terakhir adalah kemampuan untuk mengelola dan untuk dikelola. Maksud dari fitur ini adalah sebuah *gateway* harus dapat mengelola *node-node* yang ada pada domain sensorik. Lalu, *gateway* juga harus dapat dikelola oleh *network administrator*.

### 2.2.4 LoRa

LoRa merupakan singkatan dari *Long Range*. LoRa adalah sebuah teknik modulasi *spread spectrum* yang berdasarkan teknologi *chirp spread spectrum* (CSS) (Semtech, 2018). Teknologi LoRa dikembangkan karena tingginya permintaan akan perangkat jaringan nirkabel yang memiliki konektifitas jarak jauh, hemat daya, dan berbiaya rendah. LoRa menggunakan frekuensi 915 MHz (Semtech, 2018). Frekuensi tersebut termasuk kedalam frekuensi *sub-GHz* sehingga memiliki kemampuan penetrasi yang baik ke objek yang menghalangi rambatan sinyal seperti tembok karena menurut (de Jong, Camire dan Rogers, 2011)

Menurut Devalal dan Karthikeyan (2018), jauhnya jangkauan dari LoRa disebabkan oleh modulasinya yang menggunakan CSS dan tingginya sensitifitas pada sisi penerima menggunakan LoRa. Sedangkan hematnya penggunaan daya pada perangkat LoRa disebabkan oleh model komunikasi yang digunakan, yaitu komunikasi *asynchronous* yang artinya suatu node hanya akan melakukan komunikasi apabila ada data yang akan dikirimkan layaknya metode ALOHA. Penggunaan topologi star juga salah satu hal yang menyebabkan hematnya penggunaan daya karena mempersingkat jalur pengiriman data.

### 2.2.5 *Electrocardiogram*

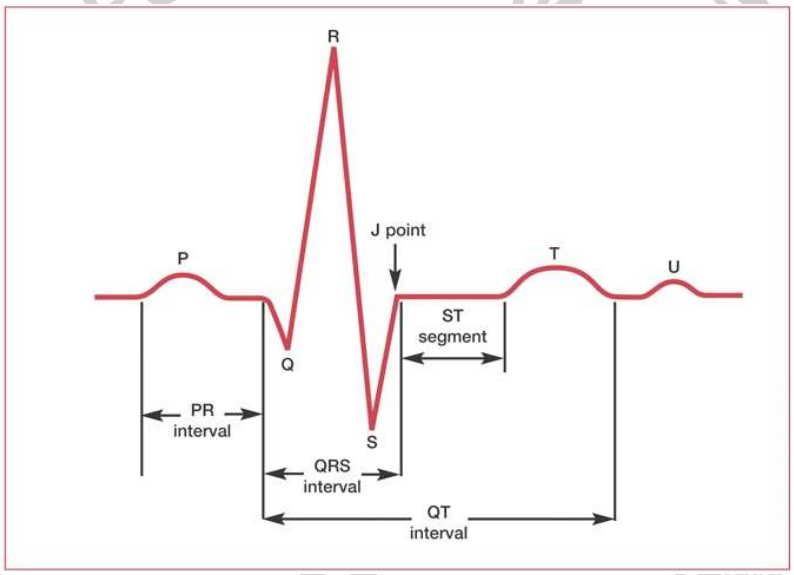
*Electrocardiogram* (ECG) merupakan sebuah tes yang mengukur aktifitas kelistrikan pada detak jantung. Dalam satu kali jantung berdetak, sebuah impuls listrik bergerak melewati jantung. Impuls tersebutlah yang membuat otot jantung memompa darah (American Heart Association, 2015). ECG dilakukan dengan cara menempelkan sensor dalam bentuk elektroda langsung ke kulit pada dada dan/atau bagian tubuh lainnya. Sinyal-sinyal yang ditangkap lalu direkam dan dianalisis oleh dokter. Gambar 2.3 merupakan contoh dari grafik data ECG.

**Gambar 2.3 Contoh grafik ECG**
Sumber: (Diehl, 2011)

ECG dilakukan untuk melihat apakah jantung seseorang berdetak terlalu cepat, terlalu lambat, atau normal. ECG juga dapat dilakukan untuk mendiagnosa apakah seseorang sedang terkena serangan jantung, memiliki kecacatan pada jantung,

atau masalah pada jantungnya. Namun, untuk dapat mendiagnosa kondisi jantung pasien, data hasil pemeriksaan ECG harus dianalisis oleh pihak yang ahli pada bidang kardiologi. Data ECG hanya dapat dianalisis apabila perekamannya dilakukan minimal dengan frekuensi 100 Hz (Kwon, Jeong dan Kim, 2018). Selain itu, data ECG juga mulai dapat dianalisis apabila perekamannya telah dilakukan selama 10 detik atau lebih (Hodgart dan Macfarlane, 2004). Untuk kondisi *non-critical*, *delay* yang masih dapat ditoleransi pada pengiriman data ECG adalah sampai dengan 4 detik. Sedangkan apabila kondisinya *critical*, *delay* yang masih dapat ditoleransi adalah hanya sampai dengan 3 detik (Alesanco dan García, 2010).

Analisis data ECG pasien dilakukan dengan memperhatikan pola dasar aktifitas listrik pada ECG yang terdiri dari 3 gelombang, yaitu: gelombang P; gelombang Q; gelombang R; gelombang S; dan gelombang T seperti yang digambarkan pada Gambar 2.4.




**Gambar 2.4 Pola dasar aktifitas listrik pada ECG**
Sumber: (Ashley dan Niebauer, 2004)

Gelombang P merupakan gelombang serupa lengkungan kecil yang merepresentasikan depolarisasi atrium. Tiga gelombang selanjutnya (Q, R, dan S) adalah gelombang yang merepresentasikan depolarisasi ventrikular. Gelombang Q merepresentasikan depolarisasi septum interventrikular, Gelombang R merepresentasikan depolarisasi dari massa utama pada ventrikel, dan Gelombang S merepresentasikan depolarisasi terakhir pada ventrikel. Dan yang terakhir adalah gelombang T yang merepresentasikan repolarisasi ventrikular. Repolarisasi atrium tidak terlihat pada grafik dikarenakan sinyalnya tertimpa oleh gelombang Q, R, dan S (Ashley dan Niebauer, 2004).

### 2.2.6 WEMOS LOLIN D32 (ESP32)

ESP32 adalah sebuah *chip* MCU yang terintegrasi dengan WiFi dan Bluetooth yang dirancang dengan teknologi *ultra-low-power* 40 nm milik Taiwan Semiconductor Manufacturing Company (TSMC). ESP32 dirancang untuk digunakan pada aplikasi *mobile* (bergerak), *wearable electronic* (elektronik yang dapat dikenakan), dan IoT. Perangkat ini juga sudah memenuhi berbagai karakteristik dari *chip* hemat daya, seperti *clock gating* yang baik, memiliki beberapa mode daya, dan penskalaan daya dinamis (Espressif Systems, 2019b).

WEMOS LOLIN D32 merupakan MCU berbasis ESP32 yang diproduksi oleh WEMOS Electronics. Modul ESP32 yang digunakan oleh perangkat ini bertipe ESP32-WROOM-32. Perangkat ini secara *default* telah dipasang *firmware* Micropython namun dapat diganti dengan Arduino apabila diinginkan (WEMOS Electronics, 2018). Bentuk fisik WEMOS LOLIN D32 ditunjukkan pada Gambar 2.5 dan spesifikasi lengkapnya ditunjukkan pada Tabel 2.2 dan Tabel 2.3.


**Gambar 2.5 WEMOS LOLIN D32**
Sumber: (WEMOS Electronics, 2018)

**Tabel 2.2 Spesifikasi WEMOS LOLIN D32**

| Keterangan | Spesifikasi |
|---|---|
| Mikrokontroler | ESP32-WROOM-32 |
| *Board power supply* (USB) | 5 V |
| Baterai yang didukung | Baterai Lipo 3.7 V |
| Tegangan operasi | 3.3 V |
| Pin I/O digital | 22 |
| Pin *input* analog | 6 (VP, VN, 32, 33, 34, 35) |
| Resolusi pin *input* analog | 12 bit |
| Pin *output* analog | 2 (25, 26) |

| Keterangan | Spesifikasi |
|---|---|
| *Built-in* LED | 5 |
| Kecepatan *clock* (maks) | 240 MHz |
| *Flash* | 4 MB |
| Panjang | 57 mm |
| Lebar | 25.4 mm |
| Berat | 6.1 g |

Sumber: (WEMOS Electronics, 2018)

**Tabel 2.3 Spesifikasi ESP32-WROOM-32**

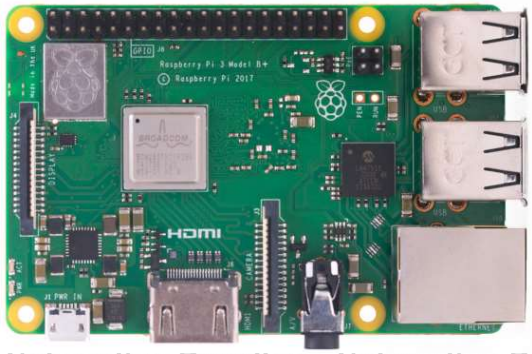
| Kategori | Keterangan | Spesifikasi |
|---|---|---|
| Sertifikasi | Sertifikasi RF | FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC |
| | Sertifikasi Wi-Fi | Wi-Fi Alliance |
| | Sertifikasi Bluetooth | BQB |
| | Sertifikasi lingkungan | RoHS/REACH |
| Pengujian | Kehandalan | HTOL/HTSL/uHAST/TCT/ESD |
| Wi-Fi | Protokol | 802.11 b/g/n (802.11n sampai 150 Mbps) |
| | | Mendukung aggregasi A-MPDU dan A-MSDU dan 0.4 µs *guard interval* |
| | Rentang frekuensi | 2.4 GHz ~ 2.5 GHz |
| Bluetooth | Protokol | Bluetooth v4.2 BR/EDR dan BLE |
| | Radio | *Receiver* NZIF dengan sensitifitas –97 dBm |
| | | *Transmitter* kelas-1, kelas-2, dan kelas-3 |
| | | AFH |
| | Audio | CVSD and SBC |
| Perangkat keras | CPU | Xtensa dual-core (or single-core) 32-bit LX6 microprocessor |
| | RAM | 520 KiB SRAM |
| | Antarmuka modul | SD *card*, UART, SPI, SDIO, I2C, LED PWM, *Motor* PWM, I2S, IR, *pulse counter*, GPIO, *capacitive touch sensor*, ADC, DAC |
| | Sensor tertanam | *Hall sensor* |

| Kategori | Keterangan | Spesifikasi |
|---|---|---|
| | Kristal terintegrasi | 40 MHz |
| | *Flash* SPI terintegrasi | 4 MB |
| | Tegangan operasi/sumber daya | 3.0 V ~ 3.6 V |
| | Arus operasi | Rata-rata: 80 mA |
| | Arus minimum yang diberikan oleh sumber daya | 500 mA |
| | Suhu operasi yang direkomendasikan | –40 °C ~ +85 °C |
| | Ukuran paket | (18.00±0.10) mm × (25.50±0.10) mm × (3.10±0.10) mm |
| | *Moisture Sensitivity Level* (MSL) | *Level* 3 |

Sumber: (Espressif Systems, 2019a)

### 2.2.7 Raspberry Pi 3 Model B+

Raspberry Pi adalah sebuah komputer berbiaya rendah yang memiliki ukuran layaknya sebuah kartu kredit dan dapat disambungkan dengan layar komputer, *keyboard*, dan *mouse*. Perangkat ini mampu melakukan hal apapun yang dapat dilakukan oleh komputer *desktop*, dari mulai menjelajah internet dan memutar video HD, sampai membuat *spreadsheet*, dokumen, dan memainkan *games*. Terlebih lagi, perangkat ini memiliki kemampuan untuk berinteraksi dengan dunia luar, dan telah digunakan dalam berbagai macam proyek digital, mulai dari proyek musik sampai proyek stasiun cuaca (Saville, 2019).




**Gambar 2.6 Raspberry Pi 3 model b+**
Sumber: (Raspberry Pi Foundation, 2016)

Raspberry Pi 3 model B+ adalah produk terbaru dari Raspberry Pi generasi ketiga. Perangkat ini diperkasai dengan prosesor *quad core* 64-bit 1.4 GHz, modul *wireless* LAN (WLAN) *dual-band* 2.4 GHz dan 5 GHz, modul Bluetooth 4.2 atau *Bluetooh Low Energy* (BLE), Ethernet, dan *Power over Ethernet* (PoE). Modul WLAN

yang digunakan sudah melalui uji sertifikasi *modular compliance* (Raspberry Pi Foundation, 2016). Bentuk fisik Raspberry Pi 3 model B+ ditunjukkan pada Gambar 2.6 sedangkan spesifikasi lengkapnya ditunjukkan pada Tabel 2.4.

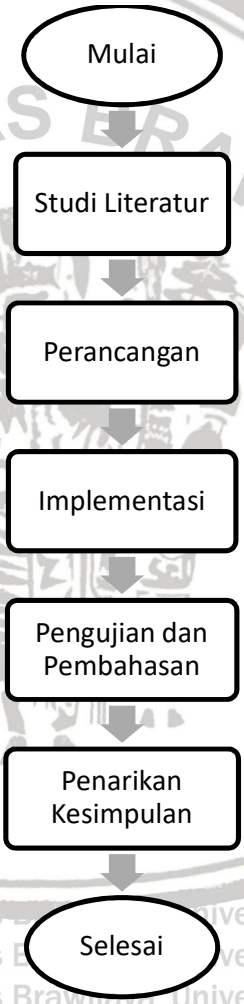
**Tabel 2.4 Spesifikasi Raspberry Pi 3 model B+**

| Keterangan | Spesifikasi |
|---|---|
| Prosesor | Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4 GHz |
| Memori | 1 GB LPDDR2 SDRAM |
| Konektifitas | 2.4 GHz dan 5 GHz IEEE 802.11.b/g/n/ac *wireless* LAN, Bluetooth 4.2, BLE |
| | Gigabit Ethernet lewat USB 2.0 (*throughput* maksimal 300Mbps) |
| | 4 × *port* USB 2.0 |
| Akses | *Extended* 40-*pin* GPIO *header* |
| Video dan suara | 1 × HDMI |
| | *Port* tampilan MIPI DSI |
| | *Port* kamera MIPI CSI |
| | 4 kutub *output stereo* dan *port* video komposit |
| Multimedia | H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics |
| SD *card* yang didukung | *Micro* SD |
| *Input* daya | 5 V/2.5 A DC via konektor *micro* USB |
| | 5 V DC via *header* GPIO |
| | *Power over Ethernet* (PoE)–*enabled* (membutuhkan PoE HAT terpisah) |
| Lingkungan | Suhu operasi, 0–50°C |

Sumber: (Raspberry Pi Foundation, 2016)

# BAB 3 METODOLOGI PENELITIAN

Bab ini menjelaskan tahapan-tahapan sistematis yang dilakukan untuk menyelesaikan penelitian ini. Penelitian ini merupakan penelitian bertipe implementatif perancangan yang artinya penelitian ini akan menghasilkan sebuah rancangan produk dengan menerapkan sebagian prinsip-prinsip rekayasa. Produk yang dirancang pada penelitian ini adalah sebuah mekanisme pengiriman data ECG berbasis LoRa. Gambar 3.1 merupakan diagram alir dari metodologi yang dipilih untuk digunakan pada penelitian ini.




**Gambar 3.1 Metodologi penelitian**

## 3.1 Studi Literatur

Tahapan ini merupakan tahapan dimana peneliti mencari dan mempelajari penelitian-penelitian sebelumnya yang terkait dengan penelitian yang akan dilakukan oleh peneliti. Penelitian-penelitian yang dijadikan pendukung adalah penelitian dengan topik IoT, LoRa, dan ECG. Hal ini dilakukan untuk mendukung

penelitian yang akan dijalankan. Pada tahapan ini, peneliti juga mempelajari dasar-dasar teori yang terkait dengan penelitian ini, seperti: IoT, *node* sensor, *gateway* pada IoT, LoRa, ECG, ESP32, dan Raspberry Pi. Hal ini perlu dilakukan agar peneliti dapat memahami terlebih dahulu teori-teori dasar yang berlaku dan berpengaruh pada penelitian yang dilakukan sehingga tidak keliru dalam melakukan penelitian. Sumber-sumber yang digunakan adalah sumber yang kredibel dan dapat dipertanggung jawabkan seperti buku, jurnal, halaman dari situs web resmi, dokumen resmi, dan prosiding.

## 3.2 Perancangan

Perancangan merupakan tahapan dimana peneliti membuat rancangan-rancangan tentang bagaimana implementasi dan pengujian pada penelitian ini dilakukan. Hal ini dilakukan agar implementasi dan pengujian dapat dilakukan secara terarah. Perancangan ini terdiri dari beberapa tahapan, yaitu mendefinisikan sistem secara umum, merancang lingkungan penelitian, merancang arsitektur jaringan, merancang *node* sensor, merancang LoRa *gateway*, dan merancang pengujian.

### 3.2.1 Deskripsi Umum Sistem

Tahapan ini merupakan tahapan pendefinisian sistem secara umum. Dimulai dari mendefinisikan *node-node* apa saja yang membentuk sistem yang dibuat. Lalu mendefinisikan peran dari masing-masing *node* tersebut dalam sistem. Setelah itu, mendefinisikan media transmisi dan protokol untuk pengiriman yang akan digunakan untuk transmisi data pada sistem. Dan menggambarkan secara umum seperti apa cara kerja sistem yang akan dibuat.

### 3.2.2 Lingkungan Penelitian

Tahapan ini merupakan tahapan pendefinisian lingkungan untuk penelitian. Lingkungan penelitian terdiri dari lingkungan perangkat keras dan lingkungan perangkat lunak. Lingkungan perangkat keras menguraikan perangkat keras apa saja yang harus ada agar sistem pada penelitian dapat berjalan. Sedangkan lingkungan perangkat lunak menguraikan perangkat lunak apa saja yang harus ada agar sistem pada penelitian dapat berjalan.

### 3.2.3 Perancangan Arsitektur Jaringan

Tahapan ini merupakan tahapan merancang arsitektur jaringan yang akan digunakan agar *node-node* yang membentuk sistem dapat berkomunikasi. Tahapan ini juga mendefinisikan alamat IP yang akan digunakan oleh masing-masing *node* dan mendefinisikan bagaimana cara komunikasi antar *node* pada sistem yang akan dibuat. Selain mendefinisikan cara komunikasi antar *node*, alur data pada sistem juga didefinisikan pada tahapan ini.

### 3.2.4 Perancangan *Node* Sensor

Perancangan *node* sensor merupakan tahapan dimana cara kerja *node* sensor dalam menjalankan fungsinya dan komponen-komponen yang membentuk *node*

sensor diuraikan. Fungsi-fungsi yang diuraikan meliputi merekam data ECG dari pengguna, memformat data tersebut agar dapat dikirim ke LoRa *gateway*, dan mengirimkan data yang telah diformat ke LoRa *gateway* menggunakan LoRa. Perancangan *node* sensor juga menguraikan bagaimana komponen-komponen perangkat kerasnya dirangkai agar dapat bekerja.

### 3.2.5 Perancangan LoRa *Gateway*

Perancangan LoRa *gateway* merupakan tahapan dimana fungsionalitas LoRa *gateway* dan komponen-komponen yang pembentuknya diuraikan. Fungsionalitas yang diuraikan meliputi penerimaan data ECG dari node sensor, proses autentikasi ke aplikasi pada *cloud*, dan pengunggahan data ECG ke aplikasi. Perancangan LoRa *gateway* juga menguraikan bagaimana komponen-komponen perangkat kerasnya dirangkai agar dapat bekerja.

### 3.2.6 Perancangan Pengujian

Tahapan ini merupakan tahapan pendefinisian bagaimana pengujian sistem dilakukan dengan cara menyusun skenario pengujian. Skenario pengujian tersebut berisi aspek yang diuji, instruksi cara pengujiannya, dan parameter kesuksesan dari pengujian yang dilakukan. Skenario pengujian yang disusun terdiri dari skenario pengujian fungsionalitas sistem dan skenario pengujian kinerja sistem.

## 3.3 Implementasi

Tahapan Implementasi merupakan tahapan mewujudkan perancangan yang telah dibuat sebelumnya. Sehingga implementasi yang dilakukan meliputi implementasi *node* sensor dan implementasi LoRa *gateway*. Pada implementasi *node* sensor, yang dilakukan adalah membuat *node* sensor yang dapat merekam data ECG dari pengguna, memformat data tersebut agar dapat dikirim ke LoRa *gateway*, dan mengirimkan data yang telah diformat ke LoRa *gateway* menggunakan teknologi LoRa. Sedangkan pada implementasi LoRa *gateway*, yang dilakukan adalah membuat LoRa *gateway* yang dapat menerima data ECG terformat dari node sensor, melakukan autentikasi *token* ke aplikasi pada *cloud*, dan meneruskan data ECG tersebut dengan cara mengunggahnya ke aplikasi.

## 3.4 Pengujian dan Pembahasan

Tahapan pengujian dan pembahasan merupakan tahapan dimana sistem diuji apakah masing-masing fungsionalitasnya dapat berjalan dengan benar dan membahas apakah hasil tersebut sudah mencapai tujuan dari dibuatnya sistem atau tidak. Sesuai dengan skenario pengujian yang telah didefinisikan pada perancangan, pengujian dibagi menjadi pengujian fungsionalitas sistem yang menguji tiap-tiap fungsionalitas sistem dan pengujian kinerja yang menguji parameter-parameter yang menggambarkan kinerja dari sistem.

### 3.5 Penarikan Kesimpulan

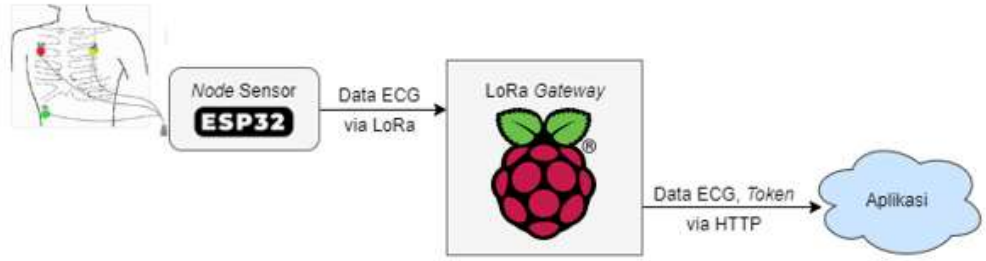
Tahapan ini menyimpulkan hasil dari pengujian yang telah dilakukan dan menjawab pertanyaan-pertanyaan pada rumusan masalah yang telah dijabarkan. Pada tahap ini peneliti juga menguraikan saran terkait pengembangan lebih lanjut dari penelitian ini untuk pembaca yang tertarik untuk mengembangkannya.

# BAB 4 PERANCANGAN

Bab ini menjelaskan beberapa hal. Hal yang pertama adalah deskripsi umum sistem yang menjelaskan secara umum dari sistem yang dibuat. Lalu, hal yang kedua adalah lingkungan penelitian yang menjabarkan mengenai perangkat keras dan perangkat lunak yang digunakan dalam penelitian ini. Setelah itu, hal yang selanjutnya adalah perancangan arsitektur jaringan, perancangan *node* sensor, dan perancangan LoRa *gateway*. Lalu hal yang terakhir adalah perancangan pengujian.

## 4.1 Deskripsi Umum Sistem

Sistem yang dibangun terdiri dari *node* sensor, LoRa *gateway*, dan aplikasi. Aplikasi yang digunakan berada pada *cloud*, sehingga dibutuhkan koneksi ke internet untuk dapat mengirimkan data ke aplikasi. Sedangkan, media transmisi yang digunakan oleh *node* sensor adalah LoRa yang tidak mendukung koneksi ke internet. Oleh karena itu, ditambahkan LoRa *gateway* ke dalam sistem sebagai perantara untuk meneruskan data dari *node* sensor ke aplikasi. LoRa *gateway* memiliki antarmuka LoRa untuk menerima data dari *node* sensor dan antarmuka WiFi untuk meneruskan data ke aplikasi via HTTP.




**Gambar 4.1 Gambaran umum sistem**

Gambar 4.1 merupakan gambaran umum sistem secara garis besar dimana bagian yang berwarna abu-abu merupakan komponen yang difokuskan dalam penelitian ini. Sistem yang akan dibuat nantinya akan terdiri dari *node* sensor ECG yang berupa ESP32 dengan antarmuka LoRa, LoRa *gateway* yang berupa Raspberry Pi dengan antarmuka LoRa dan WiFi, dan aplikasi pada *cloud* yang dikembangkan pada penelitian Pramukantoro et al. (2017). Pada sistem yang dibuat, *node* sensor akan mengirimkan data ECG ke LoRa *gateway* menggunakan LoRa lalu diteruskan oleh LoRa *gateway* dengan menyertakan *token* terautentikasi ke aplikasi pada *cloud* dengan menggunakan protokol HTTP.

## 4.2 Lingkungan Penelitian

Lingkungan penelitian terdiri dari lingkungan perangkat keras dan lingkungan perangkat lunak. Subbab akan menjelaskan perangkat-perangkat apa saja yang digunakan dan keterangan singkat terkait peranan perangkat tersebut dalam penelitian ini.

### 4.2.1 Lingkungan Perangkat Keras

Perangkat-perangkat keras yang digunakan pada penelitian ini diuraikan pada Tabel 4.1.

**Tabel 4.1 Lingkungan perangkat keras**

| No | Perangkat | Jumlah | Keterangan |
|---|---|---|---|
| 1 | WEMOS LOLIN D32 | 1 | Mikrokontroler yang berperan sebagai *node* sensor dan berfungsi untuk mengambil data dari modul sensor dan mengirimkannya ke LoRa *gateway* dengan menggunakan LoRa. |
| 2 | AD8232 | 1 | Modul yang berfungsi untuk merekam data ECG dari pengguna. |
| 3 | HopeRF RFM95W | 2 | Modul yang berfungsi untuk mentransmisikan atau menerima data menggunakan LoRa. 1 buah untuk dipasang pada *node* sensor dan 1 buah untuk dipasang pada LoRa *gateway*. |
| 4 | Raspberry Pi model 3B+ | 1 | Komputer berukuran kecil yang bertindak sebagai LoRa *gateway* untuk menerima data dari *node* sensor melalui LoRa lalu meneruskannya menuju ke aplikasi pada *cloud.* |
| 5 | MicroSD card 32GB | 1 | Media penyimpanan untuk Raspberry Pi. |
| 6 | Powerbank 10000mAh | 2 | Sumber tenaga masing-masing untuk mikrokontroler dan Raspberry Pi. |
| 7 | Laptop ASUS X450JB | 1 | Perangkat yang digunakan untuk memprogram dan mengontrol *node* sensor dan Raspberry Pi. |

### 4.2.2 Lingkungan Perangkat Lunak

Perangkat-perangkat lunak yang digunakan pada penelitian ini diuraikan pada Tabel 4.2.

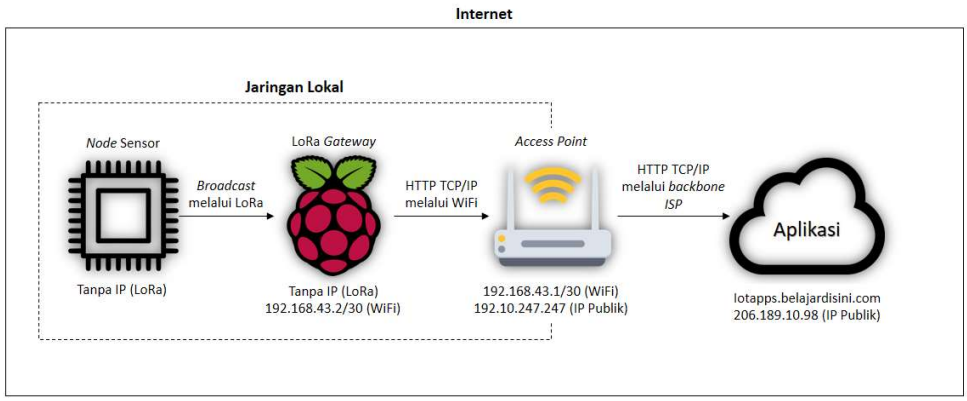
**Tabel 4.2 Lingkungan perangkat lunak**

| No | Perangkat | Keterangan |
|---|---|---|
| 1 | Micropython | *Firmware* yang digunakan pada *node* sensor. |
| 2 | uPyCraft | *Integrated Development Environment* (IDE) yang digunakan untuk memprogram *node* sensor. |
| 3 | Raspbian Stretch | Sistem operasi yang digunakan pada Raspberry Pi. |
| 5 | Thonny | IDE yang digunakan untuk memprogram Raspberry Pi. |

| No | Perangkat | Keterangan |
|---|---|---|
| 6 | Python 3.7.3 | Bahasa pemrograman yang digunakan untuk memprogram *node* sensor dan LoRa *gateway*. |
| 7 | Windows 10 64-bit | Sistem operasi yang digunakan pada laptop. |

## 4.3 Perancangan Arsitektur Jaringan

Perancangan arsitektur jaringan mendefinisikan bagaimana *node* sensor, LoRa *gateway*, dan aplikasi pada *cloud* saling terhubung dan berkomunikasi. Hal tersebut juga yang memungkinkan terjadinya transmisi data antar komponen sistem. Rancangan arsitektur jaringan pada sistem ini diilustrasikan pada Gambar 4.2.




**Gambar 4.2 Arsitektur jaringan pada sistem**

Pada arsitektur jaringan yang telah diilustrasikan pada Gambar 4.2, terdapat sebuah jaringan lokal atau intranet yang terdiri dari *node* sensor, LoRa *gateway*, dan *access point*. *Node* sensor tidak memiliki satupun alamat IP karena antarmuka yang digunakan hanya LoRa. Begitu pula pada LoRa *gateway*, perangkat tersebut juga tidak memiliki alamat IP untuk antarmuka LoRa-nya. Pada LoRa *gateway*, yang memiliki alamat IP hanyalah antarmuka WiFi, yaitu 192.168.43.2 dengan *netmask* /29. *Netmask* /29 digunakan karena alamat IP pada jaringan lokal hanya digunakan untuk antarmuka WiFi pada LoRa *gateway*, antarmuka WiFi pada *access point* yang berperan sebagai *gateway* ke internet, dan keperluan *broadcast*. Pada *access point* terdapat 2 antarmuka komunikasi, yaitu antarmuka WiFi dengan alamat IP 192.168.43.1 dengan *netmask* /29 dan antarmuka untuk tersambung ke internet melalui *backbone* ISP dengan alamat IP 192.10.247.247. Dan aplikasi yang berada pada *cloud* memiliki nama domain iotapps.belajardisini.com dengan alamat IP 206.189.94.98.

Untuk alur transmisi datanya, dimulai dari *node* sensor yang mengirimkan data ECG ke LoRa *gateway* dengan cara melakukan *broadcast* melalui LoRa. Lalu data ECG yang diterima oleh LoRa *gateway* akan diteruskan ke *access point* menggunakan protokol HTTP pada lapisan aplikasinya dan protokol TCP/IP pada

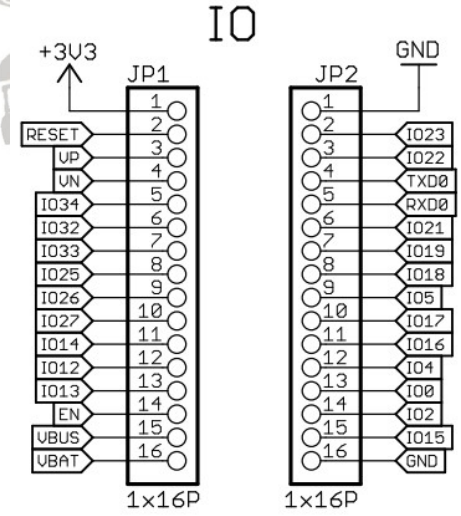
lapisan *network*-nya melalui WiFi. Setelah itu, data yang diterima oleh *access point* akan diteruskan ke aplikasi melalui *backbone ISP*.

## 4.4 Perancangan *Node* Sensor

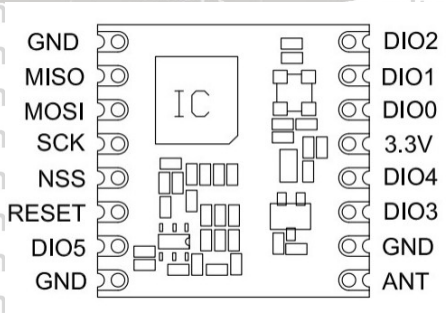
Subbab ini menjelaskan perancangan *node* sensor yang akan digunakan pada penelitian ini. Perancangan tersebut meliputi perancangan perangkat keras *node* sensor yang menjelaskan bagaimana perangkat keras *node* sensor dirangkai dan perancangan fungsionalitas *node* sensor yang menjelaskan bagaimana fungsionalitas-fungsionalitas *node* sensor nantinya akan berjalan.

### 4.4.1 Perancangan Perangkat Keras *Node* Sensor

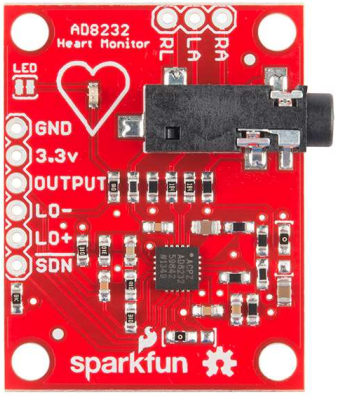
*Node* sensor pada penelitian ini terdiri dari sebuah modul sensor ECG AD8232, sebuah mikrokontroler ESP32, dan sebuah modul LoRa RFM95w. Ketiga komponen tersebut dirangkai dengan menyambungkan pin-pin dari masing-masing komponen. *Layout* pin dari masing-masing komponen ditunjukkan pada Gambar 4.3 yang merupakan *layout* pin mikrokontroler ESP32, Gambar 4.4 yang merupakan *layout* pin modul LoRa RFM95w, dan Gambar 4.5 yang merupakan *layout* pin modul sensor AD8232.




**Gambar 4.3 *Layout* pin ESP32**




**Gambar 4.4 *Layout* pin RFM95w**




**Gambar 4.5 *Layout* pin AD8232**

Pasangan pin atau *pinout* dari masing-masing komponen diuraikan menggunakan tabel. Tabel 4.3 menunjukkan *pinout* antara ESP32 dengan AD8232. Tabel 4.4. menunjukkan *pinout* antara ESP32 dengan AD8232.

**Tabel 4.3 *Pinout* ESP32 dengan AD8232**

| ESP32 | AD8232 |
|---|---|
| GND | GND |
| 3.3V | 3.3V |
| GPIO34 | OUT |
| GPIO17 | LO- |
| GPIO16 | LO+ |

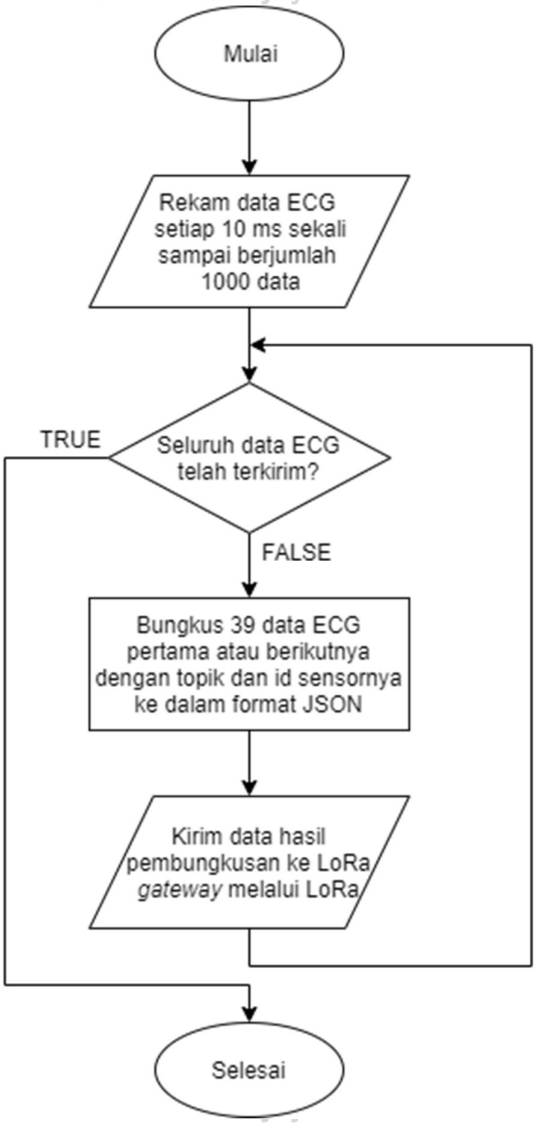
**Tabel 4.4 *Pinout* ESP32 dengan RFM95w**

| ESP32 | RFM95w |
|---|---|
| GND | GND |
| 3.3V | 3.3V |
| GPIO23 | MOSI |
| GPIO19 | MISO |
| GPIO18 | SCK |
| GPIO05 | NSS |
| GPIO2 | DIO0 |
| GPIO14 | RESET |

Mikrokontroler pada sistem ini menggunakan *firmware* MicroPython. Sehingga bahasa pemrograman yang digunakan untuk *sensing* ECG dan pengiriman data ke LoRa *gateway* adalah bahasa pemrograman Python.

### 4.4.2 Perancangan Fungsionalitas *Node* Sensor

*Node* sensor memiliki tiga fungsi utama yang akan terus berulang, yaitu merekam data ECG dari pengguna, memformat data tersebut, dan mengirimkan data yang sudah terformat ke LoRa *gateway* melalui LoRa. Gambar 4.6 merupakan diagram alir dari alur kerja *node* sensor dalam satu siklus.




**Gambar 4.6 Diagram alir *node* sensor**

Pertama, *node* sensor akan merekam data ECG dari pengguna. Dalam sekali siklus perekaman, *node* sensor mengumpulkan seribu data ECG yang direkam setiap 10 ms sekali sehingga akan menghasilkan 10 detik data ECG pengguna yang akan disimpan dalam bentuk *list*. Setiap data ECG direpresentasikan dengan angka 0-4095 karena pin ADC yang digunakan pada ESP32 hanya memiliki resolusi 12 bit. Perekaman data dilakukan setiap 10 ms sekali sampai seribu data dilakukan agar data ECG yang diperoleh nantinya akan membentuk 10 detik data ECG dengan frekuensi 100 Hz agar dapat dianalisis sesuai dengan yang telah dijelaskan pada

subbab 2.2.5. Kedua, dilakukan pemeriksaan apakah seluruh data ECG yang direkam telah terkirim atau belum. Apabila belum, *node* sensor akan memasuki langkah ketiga yaitu, data yang berhasil dikumpulkan tersebut dibungkus dengan topik beserta id sensornya dan diformat ke dalam format JSON. Gambar 4.7 menguraikan hasil pembungkusan data dan pengubahan format data mejadi JSON.

```
[(topik:String), (idSensor:String), (data:List)]
```

**Gambar 4.7 Format pembungkusan data dalam JSON**

Namun, dikarenakan ukuran maksimum *payload* LoRa yang terbatas hanya 255 *bytes* dan harus ditambahkannya topik dan id sensor pada *payload* tersebut, *node* sensor hanya mampu mengirimkan 39 data hasil rekaman ECG yang berjumlah 234 *bytes* (251 *bytes* apabila sudah diformat dengan topik dan id sensor) dalam sekali pengiriman. Oleh karena itu, data ECG yang dibungkus hanya berjumlah 39 untuk sekali pengiriman. Keempat, data yang telah dibungkus dan telah berformat JSON dikirim ke LoRa *gateway* melalui LoRa. Setelah itu, *node* sensor akan memeriksa kembali apakah data hasil perekaman ECG sudah dikirim seluruhnya atau belum. Apabila sudah, *node* sensor telah menyelesaikan satu siklus fungsi utamanya.

## 4.5 Perancangan LoRa *Gateway*

Subbab ini menjelaskan perancangan LoRa *gateway* pada penelitian ini. Perancangan tersebut meliputi perancangan perangkat keras LoRa *gateway* yang menjelaskan bagaimana komponen-komponen perangkat keras pada LoRa *gateway* dirangkai, perancangan arsitektur LoRa *gateway* yang menjelaskan bagaimana pemetaan masing-masing komponen yang menyusun LoRa *gateway*, dan perancangan masing-masing fungsionalitas dari LoRa *gateway*.

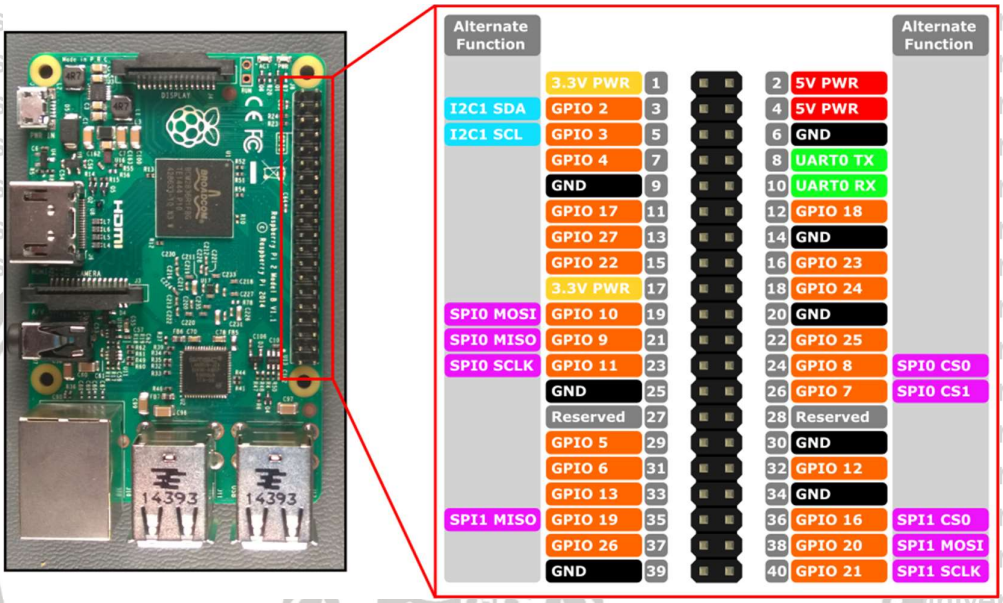
### 4.5.1 Perancangan Perangkat Keras LoRa *Gateway*

Perangkat LoRa *gateway* pada penelitian ini terdiri dari sebuah Raspberry Pi 3 Type B+ dan sebuah modul LoRa bertipe RFM95w. Kedua komponen tersebut dirangkai dengan menyambungkan pin-pin dari masing-masing komponen. *Layout* pin dari masing-masing komponen ditunjukkan pada Gambar 4.8 yang merupakan *layout* pin Raspberry Pi dan Gambar 4.4 pada subbab 4.4.1 yang merupakan *layout* pin modul LoRa RFM95w. Sedangkan pasangan pin atau *pinout* dari masing-masing komponen diuraikan menggunakan tabel. Tabel 4.5 menunjukkan *pinout* antara Raspberry Pi dengan RFM95w.

**Tabel 4.5 *Pinout* Raspberry Pi dengan RFM95w**

| Raspberry Pi | RFM9x |
|---|---|
| GPIO 17 | RESET |
| 3.3 V | 3.3 V |
| GPIO 10 | MOSI |
| GND | GND |

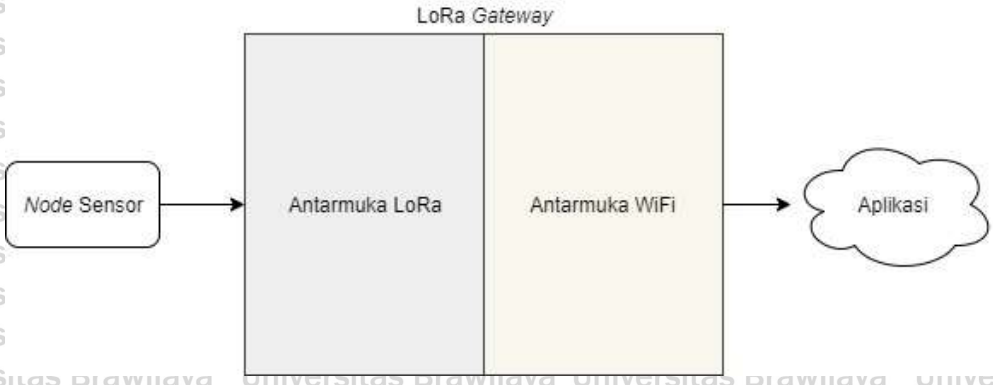
| Raspberry Pi | RFM9x |
|---|---|
| GPIO 9 | MISO |
| GPIO 25 | DIO0 |
| GPIO11 | SCK |
| GPIO 8 | NSS |




**Gambar 4.8 *Layout* pin Raspberry Pi**

### 4.5.2 Perancangan Arsitektur LoRa *gateway*

Dalam penelitian ini, LoRa *gateway* berfungsi sebagai perantara antara *node* sensor dan aplikasi pada *cloud* untuk transmisi data. Sehingga, komponen-komponen penyusun LoRa *gateway* harus dapat menerima data dari *node* sensor dan meneruskannya ke aplikasi pada *cloud*. Oleh karena alasan tersebut, dirancanglah arsitektur LoRa *gateway* seperti pada Gambar 4.9.

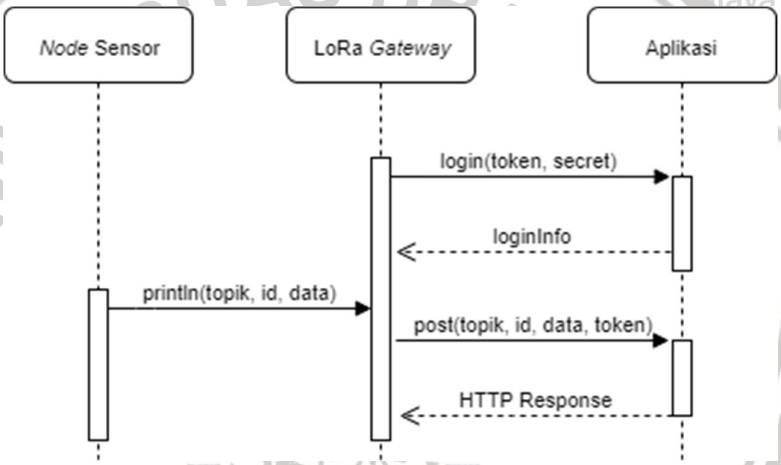



**Gambar 4.9 Arsitektur LoRa *gateway***

LoRa *gateway* yang digunakan terdiri dari dua komponen. Yang pertama adalah antarmuka LoRa yang memiliki fungsionalitas penerimaan data dari *node* sensor yang dikirimkan melalui LoRa. Lalu yang kedua adalah antarmuka WiFi yang memiliki fungsionalitas pengunggahan data yang diterima dari *node* sensor ke aplikasi yang berada pada *cloud*.

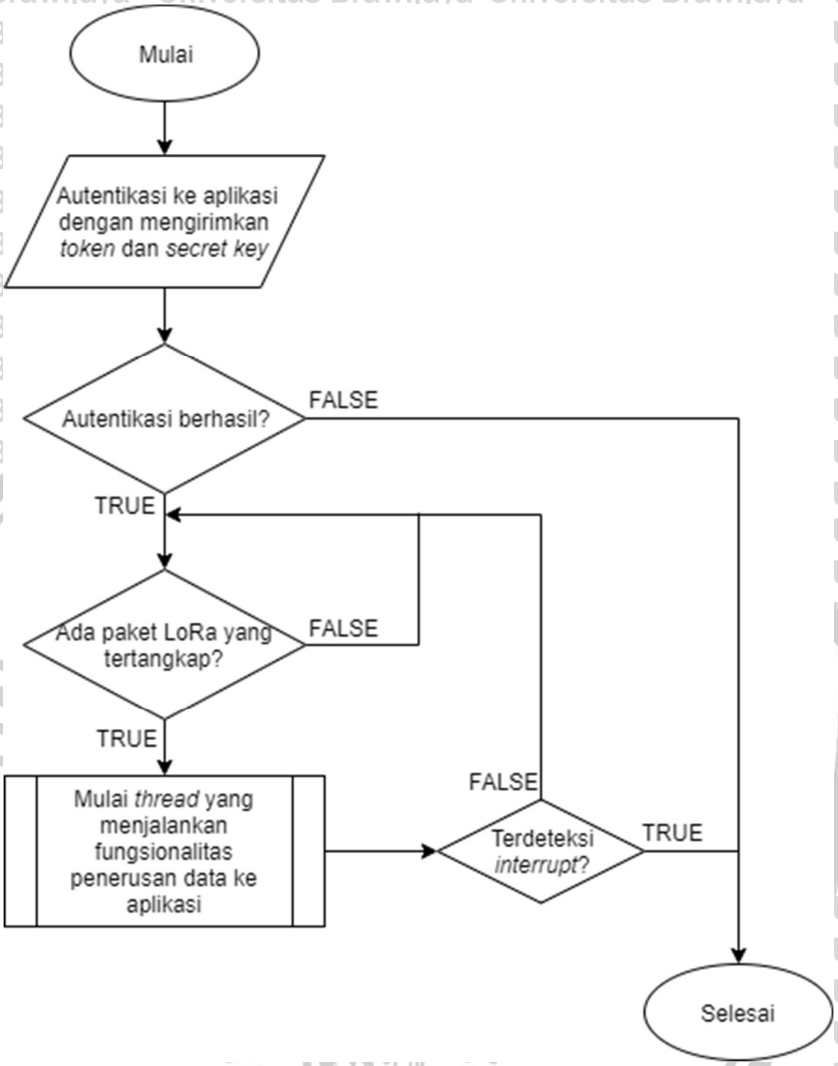
### 4.5.3 Perancangan Fungsionalitas LoRa *Gateway*

Fungsionalitas LoRa *gateway* yang dimaksud pada subbab ini adalah dimulai dari penerimaan data ECG dari *node* sensor yang dikirimkan melalui LoRa sampai pengunggahan data ECG tersebut ke aplikasi pada *cloud.* Fungsionalitas LoRa *gateway* dijelaskan dengan menggunakan *sequence diagram* dan *flow diagram* untuk agar lebih mudah dipahami. Gambar 4.10 merupakan *sequence diagram* dari fungsionalitas LoRa *gateway*.



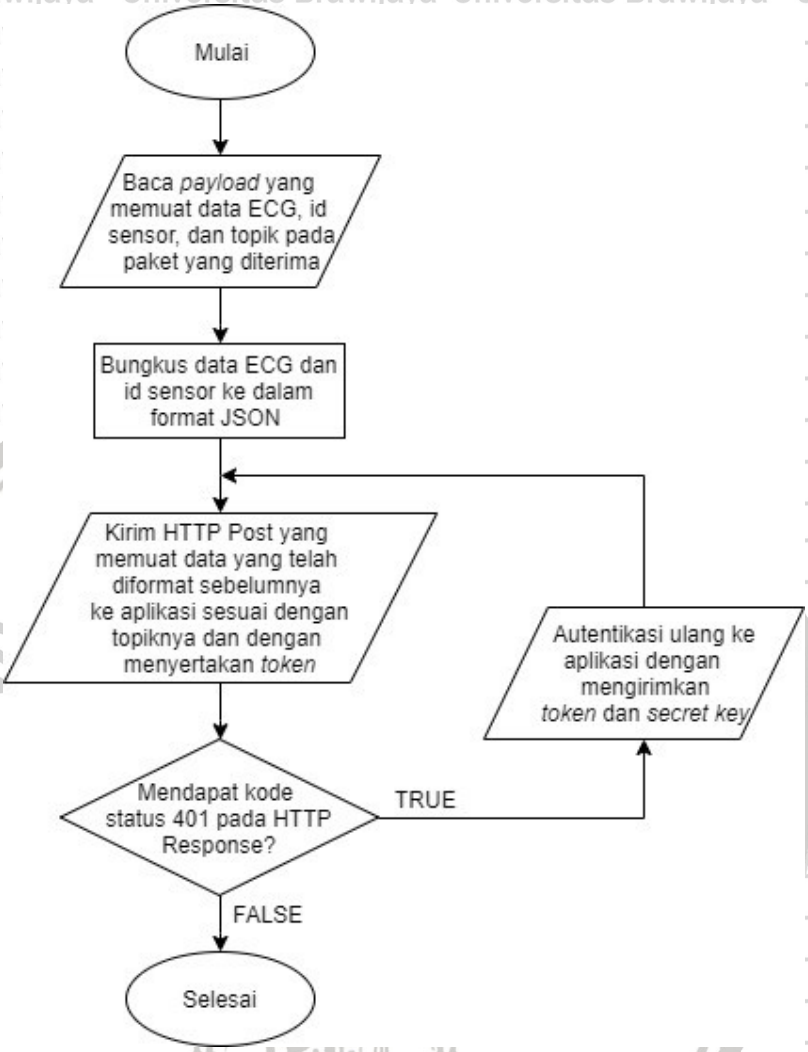

**Gambar 4.10 *Sequence diagram* LoRa *gateway***

Pertama-tama, agar LoRa *gateway* dapat mengunggah data ke aplikasi yang berada di *cloud*, LoRa *gateway* harus mengautentikasikan dirinya terlebih dahulu ke aplikasi dengan cara melakukan *login* dengan menyertakan *token* dan *secret key*-nya. Dari proses *login* tersebut LoRa *gateway* akan memperoleh sebuah *token* baru yang sudah terautentikasi pada aplikasi. Setelah proses autentikasi selesai, apabila ada *node* sensor yang mengirimkan data ECG, topik, dan ID-nya dan antarmuka LoRa pada LoRa *gateway* menangkapnya, data tersebut akan dibaca oleh LoRa *gateway* lalu diteruskan ke aplikasi dengan menyertakan *token* milik LoRa *gateway* yang telah terautentikasi pada tahap pertama. Setelah itu, aplikasi pada *cloud* akan mengirimkan pesan balasan dalam bentuk HTTP *Response*. Pesan HTTP *Response* tersebut adalah pesan yang mengandung informasi apakah penerusan data dari *node* sensor ke aplikasi berhasil dilakukan tanpa kesalahan atau tidak. Kode status 200 merupakan kode yang menandakan bahwa pengunggahan berhasil dilakukan. Untuk lebih jelasnya, proses-proses yang dijalankan di dalam LoRa *gateway* akan dijelaskan melalui *flow diagram* pada Gambar 4.11 dan pada Gambar 4.12.




**Gambar 4.11 *Flow diagram* fungsionalitas LoRa *gateway***

Tahapan pertama sebelum LoRa *gateway* memulai penangkapan paket LoRa adalah autentikasi ke aplikasi pada *cloud*. Autentikasi dilakukan dengan cara mengirimkan *request* HTTP *Post* ke aplikasi dengan menyertakan *token* dan *secret key* yang sesuai. Apabila autentikasi berhasil, maka LoRa *gateway* akan memulai penangkapan paket LoRa. Namun, apabila LoRa *gateway* tidak berhasil melakukan *login* ke aplikasi, maka fungsionalitasnya akan terhenti dan harus dijalankan ulang. Lalu, apabila ada paket LoRa yang tertangkap oleh LoRa *gateway*, LoRa *gateway* akan menginisialisasikan suatu *thread* baru yang bertugas untuk menjalankan fungsionalitas penerusan data ke aplikasi pada *cloud*. Hal ini dilakukan agar pemrosesan paket LoRa yang ditangkap dapat dilakukan secara paralel sehingga tidak memerlukan adanya antrian. Pemeriksaan tertangkapnya paket LoRa dilakukan secara berulang sampai LoRa *gateway* menangkap adanya *interrupt*. Proses yang dilakukan pada pemrosesan paket LoRa oleh masing-masing *thread* yang dibuat akan dijelaskan melalui *flow diagram* pada Gambar 4.12.




**Gambar 4.12 Fungsionalitas pemrosesan paket LoRa pada *thread***

Hal pertama yang dilakukan oleh *thread* setelah diinisialisasikan oleh LoRa *gateway* adalah membaca *payload* yang ada di dalam paket LoRa yang diterima. *Payload* tersebut memuat topik, id sensor, dan data ECG dari *node* sensor. Setelah itu, *thread* akan membungkus data JSON baru yang hanya memuat data ECG dan id sensor saja tanpa topik. Data JSON baru tersebut akan dijadikan *payload* yang akan dikirimkan ke aplikasi. *Payload* tersebut dikirimkan ke aplikasi dengan menggunakan metode HTTP *Post* dengan menyematkan topik yang sesuai pada *URL* tujuannya dan menyertakan *token* yang telah terautentikasi pada *header* paketnya. Selanjutnya, LoRa *gateway* akan menerima pesan respon berformat HTTP *Response* dari aplikasi. Apabila pesan respon tersebut memiliki kode status 401, berarti *token* yang disertakan pada *header* paket HTTP *Post* merupakan *token* yang tidak valid atau belum terautentikasi oleh aplikasi. Sehingga, *thread* akan melakukan autentikasi ulang dan mengirimkan kembali paket HTTP *Post* dengan *payload* yang sama namun *token* yang disertakan adalah *token* yang baru. Namun,

apabila pesan respon yang diterima memiliki kode status selain 401, maka *thread* sudah selesai menjalankan fungsionalitasnya.

## 4.6 Perancangan Pengujian

Pengujian yang dilakukan terbagi menjadi pengujian fungsionalitas sistem yang menguji berjalannya masing-masing fungsionalitas pada komponen sistem dan pengujian kinerja sistem yang menguji parameter-parameter untuk mengetahui kinerja dari sistem.

### 4.6.1 Perancangan Pengujian Fungsionalitas Sistem

Pengujian fungsionalitas sistem merupakan pengujian yang dilakukan untuk mengetahui apakah fungsionalitas-fungsionalitas dari masing-masing komponen dapat berjalan dengan benar atau tidak. Pengujian fungsionalitas sistem dijadikan salah satu parameter untuk menyimpulkan apakah sistem yang dibuat pada penelitian ini dapat bekerja sesuai dengan tujuannya.

Pengujian fungsionalitas sistem ini dilakukan pada *node* sensor dan LoRa *gateway*. *Node* sensor akan diuji apakah ia dapat melakukan perekaman data ECG dari pengguna, membungkus data tersebut dengan topik beserta id sensornya, dan mengirimkan data tersebut ke LoRa *gateway* melalui LoRa. Sedangkan LoRa *gateway* akan diuji apakah ia dapat menerima paket data ECG yang telah dikirimkan oleh *node* sensor melalui LoRa, melakukan autentikasi *token* ke aplikasi, dan meneruskan data yang dikirimkan oleh *node* sensor ke aplikasi pada *cloud*. Secara spesifik, skenario pengujian fungsional dijelaskan pada Tabel 4.6.

**Tabel 4.6 Skenario pengujian fungsionalitas sistem**

| Kode | Aspek | Skenario Pengujian |
|---|---|---|
| PF_01 | Kemampuan *node* sensor untuk merekam data ECG dari pengguna. | - Diketahui ketiga elektroda pada modul sensor telah melekat masing-masing pada dada kiri, dada kanan, dan pinggang kanan pengguna.<br>- Diketahui *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB.<br>- Laptop memerintahkan *node* sensor untuk menjalankan program perekaman data ECG dari pengguna.<br>- Data ECG yang berhasil direkam akan disimpan ke dalam file .txt pada *node* sensor. |
| PF_02 | Kemampuan *node* sensor untuk membungkus data ECG dengan topik yang sesuai beserta id sensornya dalam format JSON. | - Diketahui *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB.<br>- Diketahui *node* sensor telah menyimpan *file* yang berisi rekaman data ECG |

| Kode | Aspek | Skenario Pengujian |
|---|---|---|
| | | pengguna selama 10 detik yang dihasilkan dari pengujian dengan kode PF_01.<br>- Laptop memerintahkan *node* sensor untuk menjalankan program pembungkusan data ECG pengguna dengan topik dan id sensornya.<br>- Hasil pembungkusan data ECG pengguna dengan topik dan id sensornya akan ditampilkan pada layar laptop. |
| PF_03 | Kemampuan *node* sensor mengirimkan data ECG, topik, dan id sensor yang telah dibungkus dalam format JSON ke LoRa *gateway* melalui LoRa. | - Diketahui *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB.<br>- Diketahui LoRa *gateway* dalam keadaan hidup dan telah tersambung dengan laptop menggunakan *Secure Shell* (SSH) melalui jaringan WiFi.<br>- Diketahui *node* sensor telah berhasil membungkus data dengan topik beserta id sensornya dengan format JSON sesuai pengujian kode PF_02.<br>- Laptop memerintahkan LoRa *gateway* untuk menjalankan program penerimaan data ECG pengguna.<br>- Laptop memerintahkan *node* sensor untuk menjalankan program pengiriman data ECG pengguna.<br>- Pesan berhasil akan ditampilkan apabila data ECG pengguna dalam format JSON telah berhasil dikirim dan data yang diterima oleh LoRa *gateway* yang berisi data ECG beserta topik dan id sensornya akan ditampilkan pada layar laptop. |
| PF_04 | Kemampuan LoRa *gateway* harus mampu menerima data JSON yang berisi data ECG, id sensor, dan topik yang telah dikirimkan oleh *node* sensor melalui LoRa | - Diketahui *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB.<br>- Diketahui LoRa *gateway* dalam keadaan hidup dan telah tersambung dengan laptop menggunakan SSH melalui jaringan WiFi.<br>- Diketahui *node* sensor telah menyimpan data ECG pengguna yang berformat JSON yang diperoleh dari pengujian dengan kode PF_02.<br>- Laptop memerintahkan LoRa *gateway* untuk menjalankan program penerimaan data ECG pengguna. |

| Kode | Aspek | Skenario Pengujian |
|---|---|---|
| | | - Laptop memerintahkan *node* sensor untuk menjalankan program pengiriman data ECG pengguna.<br>- Pesan berhasil akan ditampilkan apabila data ECG pengguna dalam format JSON telah berhasil dikirim dan data yang diterima oleh LoRa *gateway* yang berisi data ECG beserta topik dan id sensornya akan ditampilkan pada layar laptop. |
| PF_05 | Kemampuan LoRa *gateway* mengautentikasikan *token* miliknya ke aplikasi pada *cloud* dengan menggunakan protokol HTTP | - Diketahui LoRa *gateway* telah dalam keadaan hidup dan tersambung dengan laptop menggunakan SSH melalui jaringan WiFi.<br>- Diketahui LoRa *gateway* telah memiliki *token* yang terdaftar pada aplikasi.<br>- Laptop memerintahkan LoRa *gateway* untuk menjalankan program autentikasi *token*.<br>- Pesan berhasil dan *token* yang telah terautentikasi akan ditampilkan pada layar laptop apabila autentikasi *token* berhasil dilakukan. |
| PF_06 | Kemampuan LoRa *gateway* meneruskan data ECG beserta id sensor sesuai dengan topiknya ke aplikasi pada *cloud* menggunakan protokol HTTP | - Diketahui LoRa *gateway* dalam keadaan hidup dan telah tersambung dengan laptop menggunakan SSH melalui jaringan WiFi.<br>- Diketahui LoRa *gateway* telah menyimpan data ECG pengguna, id sensor, dan topiknya yang telah dikirimkan oleh *node* sensor pada pengujian PF_03 dan *token* yang telah terautentikasi pada pengujian PF_05.<br>- Laptop memerintahkan LoRa *gateway* untuk menjalankan program penerusan data ECG pengguna.<br>- Pesan berhasil akan ditampilkan pada layar laptop apabila data ECG pengguna telah berhasil diteruskan ke aplikasi.<br>- Data ECG pengguna yang telah berhasil diteruskan ke aplikasi dapat diakses pada domain iotapps.belajardisini.com. |

### 4.6.2 Perancangan Pengujian Kinerja

Pengujian kinerja merupakan pengujian yang dilakukan untuk mengetahui parameter-parameter yang mengukur kinerja dari sistem. Pengujian kinerja sistem

dijadikan salah satu parameter untuk menyimpulkan apakah sistem yang dibuat pada penelitian ini dapat bekerja untuk memenuhi tujuannya.

Parameter yang diuji pada pengujian ini adalah kinerja dan ketersediaan. Pada parameter kinerja, yang diuji adalah *delay* pengiriman data ECG dari *node* sensor ke LoRa *gateway* melalui LoRa, *end-to-end delay* yang merupakan *delay* pengiriman data dari *node* sensor hingga diterima oleh aplikasi pada *cloud*, dan *end-to-end delay* pengiriman hasil satu siklus perekaman data ECG dari *node* sensor hingga diterima oleh aplikasi pada *cloud*. Sedangkan pada parameter ketersediaan, yang diuji adalah *Packet Delivery Ratio* pada jarak-jarak tertentu. Secara spesifik, skenario pengujian kinerja terdapat pada Tabel 4.7.

**Tabel 4.7 Skenario pengujian kinerja**

| Kode | Aspek | Skenario Pengujian |
|---|---|---|
| PK_01 | *Delay* pengiriman data ECG dari *node* sensor ke LoRa *gateway* melalui LoRa | Pengujian dilakukan dengan cara mengirimkan satu paket data ECG sekunder (39 data) ke LoRa *gateway* menggunakan LoRa dan merekam *log* waktu pengiriman dari *node* sensor dan penerimaan pada LoRa *gateway* dengan menggunakan *Network Time Protocol* dan disimpan dalam format *file* .csv pada LoRa *gateway*. Pengiriman paket dilakukan sebanyak 60 kali dengan interval 5 detik. Pengujian dilakukan 8 kali dengan jarak antara *node* sensor dengan LoRa *gateway* sebagai pembeda. Jarak-jarak tersebut adalah 100 meter, 200 meter, 300 meter, 400 meter, 500 meter, 600 meter, 700 meter, dan 800 meter. Untuk memperoleh nilai *network delay*-nya, dilakukan operasi pengurangan antara waktu penerimaan data pada LoRa *gateway* dan waktu pengiriman data dari *node* sensor. |
| PK_02 | *End-to-end delay* pengiriman data ECG dari *node* sensor sampai ke aplikasi pada *cloud* | Pengujian dilakukan dengan cara mengirimkan satu paket data ECG sekunder (39 data) ke LoRa *gateway* menggunakan LoRa dan LoRa *gateway* meneruskannya ke aplikasi pada *cloud*. Pada saat itu, LoRa *gateway* akan merekam *log* waktu mulainya pengiriman serta waktu sampainya data pada aplikasi menggunakan *Network Time Protocol* dan disimpan dalam format *file* .csv. Pengiriman paket dilakukan sebanyak 60 kali dengan interval 5 detik. Pengujian dilakukan 8 kali dengan jarak antara *node* sensor dan LoRa *gateway* sebagai pembeda. Jarak-jarak tersebut adalah 100 |

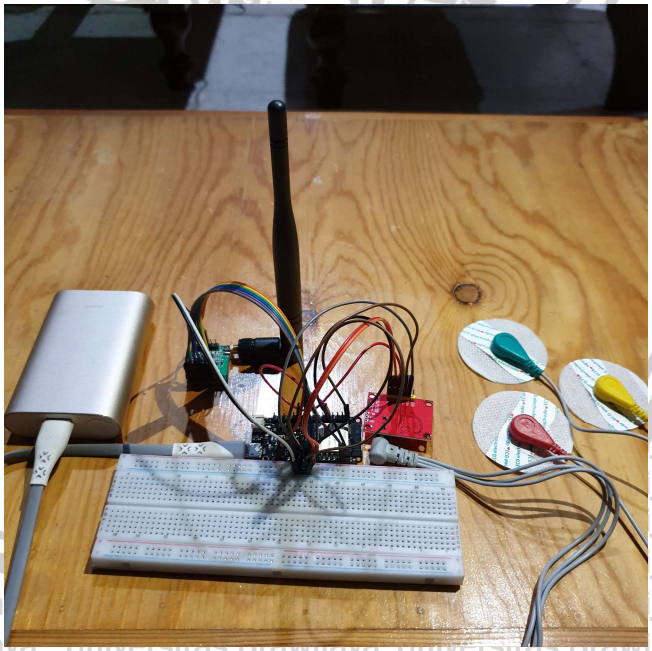
| Kode | Aspek | Skenario Pengujian |
|---|---|---|
| | | meter, 200 meter, 300 meter, 400 meter, 500 meter, 600 meter, 700 meter, dan 800 meter. Untuk memperoleh nilai *end-to-end delay*-nya, dilakukan operasi pengurangan antara waktu diterimanya data pada aplikasi dan waktu pengiriman data dari *node* sensor. |
| PK_03 | *End-to-end delay* pengiriman hasil satu siklus perekaman data ECG dari *node* sensor sampai ke aplikasi pada *cloud* | Pengujian dilakukan dengan cara mengirimkan data ECG sekunder yang berjumlah 1000 data ke LoRa *gateway* menggunakan LoRa dan LoRa *gateway* meneruskannya ke aplikasi pada *cloud*. Pada saat itu, *log* waktu mulainya pengiriman data pertama serta waktu sampainya data terakhir pada aplikasi akan direkam menggunakan *Network Time Protocol* dan disimpan dalam format *file* .csv. Pengujian dilakukan 60 kali dengan menggunakan data yang sama pada jarak 100 meter. Untuk memperoleh nilai *end-to-end delay*, dilakukan operasi pengurangan antara waktu penerimaan data terakhir pada aplikasi dan waktu pengiriman data pertama dari *node* sensor. |
| PK_04 | *Packet Delivery Ratio* yang dikriimkan oleh *node* sensor melalui LoRa. | Pengujian dilakukan dengan cara mengirimkan satu paket data ECG sekunder (39 data) ke LoRa *gateway* menggunakan LoRa dan menghitung jumlah paket yang diterima pada LoRa *gateway*. Perhitungan jumlah paket yang masuk ke LoRa *gateway* dilakukan dengan menghitung jumlah *log* waktu paket yang direkam pada saat menguji *delay* transmisi dan *delay end-to-end*. Pengiriman data dilakukan sebanyak 60 kali dengan interval 5 detik. Pengujian dilakukan 8 kali dengan jarak antara *node* sensor dan LoRa *gateway* sebagai pembeda. Jarak-jarak tersebut adalah 100 meter, 200 meter, 300 meter, 400 meter, 500 meter, 600 meter, 700 meter, dan 800 meter. Untuk memperoleh nilai *PDR*-nya, dilakukan operasi pembagian jumlah paket yang masuk ke LoRa *gateway* dengan jumlah paket yang dikirim oleh *node* sensor. |

# BAB 5 IMPLEMENTASI

Bab ini menjelaskan implementasi sistem yang terdiri dari implementasi *node* sensor dan implementasi LoRa *gateway*. Implementasi *node* sensor terdiri dari implementasi perekaman data ECG pengguna dan implementasi pengiriman data ECG ke LoRa *gateway*. Sedangkan implementasi LoRa *gateway* terdiri dari implementasi LoRa *gateway* sebagai penerima dan penerus paket data ECG dari *node* sensor ke aplikasi pada *cloud*. Penelitian ini menggunakan aplikasi pada *cloud* dari penelitian Pramukantoro et al. (2017) sehingga bab ini hanya berfokus kepada *node* sensor dan LoRa *gateway* tanpa menjelaskan aplikasinya.

## 5.1 Implementasi *Node* Sensor

Subbab ini menjelaskan implementasi yang dilakukan untuk membangun *node* sensor sehingga dapat melakukan fungsi perekaman data ECG dari pengguna dan dapat melakukan fungsi pengiriman data ECG yang direkam ke LoRa *gateway* melalui LoRa. Implementasi perangkat keras untuk *node* sensor ditunjukkan pada Gambar 5.1.



**Gambar 5.1 Implementasi perangkat keras *node* sensor**

Sesuai dengan perancangan *node* sensor, implementasi modul sensor yang digunakan untuk merekam aktifitas kelistrikan pada jantung adalah AD8232 dan mikrokontroler yang digunakan untuk menangkap hasil *sensing*-nya adalah ESP32. Konfigurasi pin untuk menyambungkan kedua perangkat tersebut dilakukan sesuai dengan *pinout* pada Tabel 4.3 dengan menggunakan kabel *jumper*. Sedangkan modul LoRa yang digunakan untuk melakukan pengiriman data ke *gateway* adalah RFM95w. Konfigurasi pin untuk menyambungkan RFM95w dengan ESP32 dilakukan sesuai dengan *pinout* pada Tabel 4.4 juga dengan

menggunakan kabel *jumper*. Namun terdapat pengecualian pada penyambungan pin GND dan pin 3.3 V, dimana pada sistem ini membutuhkan tambahan sebuah *breadboard* karena mikrokontroler hanya memiliki satu pin 3.3 V, sedangkan modul sensor ECG dan modul LoRa masing-masing membutuhkan satu pin GND dan satu pin 3.3 V untuk sumber tenaganya. Untuk implementasi fungsionalitas dari *node* sensor ditunjukkan pada Tabel 5.1.

**Tabel 5.1 *Pseudocode node* sensor**

*Pseudocode* Node Sensor

```
1  FUNCTION main()
2      config <-- konfigurasi lora
3      lora <-- controller transceiver lora
4      adc <-- representasi pin 34
5      lo1 <-- representasi pin 16
6      lo2 <-- representasi pin 17
7      memberikan pelemahan arus pada adc sebesar 11 dB
8      data <-- hasil pemanggilan fungsi rekamData(adc, lo1,
   lo2)
9      memanggil fungsi kirimData(lora, data)
10 END-FUNCTION
```

*Node* sensor pertama-tama menginisialisasikan beberapa variabel yang diperlukan yaitu `config` yang merupakan konfigurasi dari lora, `lora` yang merupakan *controller* dari *transceiver* LoRa, `adc` yang merupakan representasi untuk mengakses pin 34 dimana input sinyal ECG dari modul sensor dibaca, `lo1` yang merupakan representasi untuk mengakses pin 16 dimana input sinyal *leadsoff* negatif dari modul sensor dibaca, dan `lo2` yang merupakan representasi untuk mengakses pin 17 dimana input sinyal *leadsoff* positif dari modul sensor dibaca. Setelah itu, *node* sensor memberikan pelemahan arus pada `adc` sebesar 11 dB. Selanjutnya *node* sensor melakukan perekaman data ECG dari pengguna dan pengiriman data hasil rekaman ke LoRa *gateway*. Perekaman data ECG dari pengguna dilakukan dengan memanggil fungsi `rekamData` dengan argumen `adc`, `lo1`, dan `lo2`. Sedangkan pengiriman data ke LoRa *gateway* dilakukan dengan memanggil fungsi `kirimData` dengan argumen `lora` dan `data`.

### 5.1.1 Implementasi Fungsi Perekaman Data ECG dari Pengguna

Sesuai dengan namanya, fungsi ini digunakan untuk merekam data ECG dari pengguna menggunakan modul sensor AD8232. Untuk lebih spesifiknya, fungsi perekaman data ECG dari pengguna pada *node* sensor akan dijelaskan melalui *pseudocode* pada Tabel 5.2.

**Tabel 5.2 *Pseudocode* fungsi rekamData**

*Pseudocode* rekamData

```
1  FUNCTION rekamData(adc, lo1, lo2)
2      data <-- []
3      counter <-- 0
4      WHILE counter < 1000
5          IF nilai lo1 != 1 AND nilai lo2 != 1 THEN
6              menambahkan hasil pembacaan adc ke array data
7              counter <-- counter + 1
8              memasuki fase sleep selama 9ms
```

```
 9            END-IF
10        END-WHILE
11        RETURN data
12    END-FUNCTION
```

Fungsi `rekamData` memerlukan tiga argumen yaitu `adc` yang merupakan representasi pin 34, `lo1` yang merupakan representasi pin 16, dan `lo2` yang merupakan representasi pin 17. Dalam fungsi ini diinisialisasikan variabel `data` yang digunakan untuk menyimpan data hasil rekam ECG pengguna dan variabel `counter` yang digunakan untuk menghitung perulangan pembacaan nilai pada `adc` yang akan dijalankan sebanyak seribu kali. Pembacaan nilai pada `adc` hanya akan dilakukan apabila nilai *leadsoff* positif dan negatif sama-sama bukan 1 sehingga terdapat seleksi kondisi *if else* yang memeriksa nilai dari `lo1` dan `lo2`. Apabila kedua nilainya bukan 1, maka hasil pembacaan nilai `adc` akan dimasukkan kedalam array `data`. Setelah pembacaan dilakukan, `counter` akan ditambahkan dengan 1 untuk masuk ke perulangan selanjutnya. Akan tetapi, sebelum memasuki perulangan selanjutnya, *node* sensor akan memasuki fase *sleep* terlebih dahulu selama 9 ms. Hal tersebut dilakukan untuk memastikan bahwa perekaman data dilakukan sekali setiap 10 ms. Setelah perulangan pembacaan nilai pada `adc` telah dilakukan sebanyak seribu kali, fungsi `rekamData` akan mengembalikan nilai `data` ke bagian dimana fungsi tersebut dipanggil.

### 5.1.2 Implementasi Fungsi Pengiriman Data ECG ke LoRa *Gateway*

Sesuai dengan namanya, fungsi ini digunakan untuk mengirimkan data ECG yang telah berhasil direkam ke LoRa *gateway* melalui LoRa dengan menggunakan modul RFM95w. Untuk lebih spesifiknya, implementasi fungsi pengiriman data ECG ke LoRa *gateway* pada *node* sensor akan dijelaskan melalui *pseudocode* pada Tabel 5.3.

**Tabel 5.3 *Pseudocode* fungsi kirimData**

*Pseudocode* kirimData

```
 1  FUNCTION kirimData(lora, data)
 2      pointerKiri <-- 0
 3      pointerKanan <-- 39
 4      panjangData <-- panjang dari data
 5      WHILE pointerKiri < panjangData
 6          dataSend <-- data[pointerKiri:pointerKanan]
 7          payload <-- ['ecg', '0001', dataSend] dalam format
    JSON
 8          mengirimkan payload menggunakan lora
 9          pointerKiri <-- pointerKiri + 39
10          pointerKanan <-- pointerKanan + 39
11          IF pointerKanan > panjangData
12              pointerKanan <-- panjangData
13          END-IF
14      END-WHILE
15  END-FUNCTION
```

Fungsi `kirimData` memerlukan dua argumen, yaitu `lora` yang merupakan *controller* untuk modul LoRa dan `data` yang merupakan data ingin dikirim. Dalam fungsi ini diinisialisasikan variabel `pointerKiri` yang digunakan sebagai

penunjuk indeks awal, `pointerKanan` yang digunakan sebagai penunjuk indeks akhir, dan `panjangData` yang menyimpan panjang dari `data`. Didalam fungsi ini terdapat perulangan yang akan terus berjalan selama `pointerKiri` bernilai kurang dari `panjangData`. Di dalam perulangan tersebut dijalankan proses pemformatan `payload` yang ditunjukkan pada baris 6 dan 7 dimana data pada indeks ke-`pointerKiri` sampai ke-`pointerKanan` dimasukkan kedalam variabel `dataSend` dan disatukan dengan String `'ecg'` dan `'0001'` dalam bentuk *list* berformat JSON di dalam variabel `payload`. Setelah itu, fungsi tersebut akan mengirimkan `payload` menggunakan `lora` yang ada pada argumen. Setelah pengiriman data dalam satu perulangan selesai dilakukan, `pointerKiri` dan `pointerKanan` akan mengalami penambahan nilai sebesar 39. Namun, apabila nilai `pointerKanan` sudah melebihi dari `panjangData` yang berarti `pointerKanan` sudah melebihi indeks terakhir pada `data`, nilai `pointerKanan` akan diubah sesuai dengan `panjangData`.

## 5.2 Implementasi LoRa *gateway*

Subbab ini menjelaskan implementasi yang dilakukan untuk membangun LoRa *gateway* sehingga dapat menjalankan fungsi untuk menerima paket data ECG dari *node* sensor, mengautentikasikan *token* ke aplikasi, dan meneruskan data pada paket yang diterima ke aplikasi pada *cloud*. Implementasi perangkat keras pada LoRa *gateway* ditunjukkan pada Gambar 5.2.

**Gambar 5.2 Implementasi perangkat keras LoRa *gateway***

Sesuai dengan perancangan LoRa *gateway*, modul LoRa yang digunakan untuk menerima data dari *node* sensor adalah RFM95w dan komputer yang digunakan untuk memproses data tersebut adalah Raspberry Pi 3 Type B+. Konfigurasi pin untuk menyambungkan kedua perangkat tersebut dilakukan sesuai dengan *pinout* pada Tabel 4.5 dengan menggunakan kabel *jumper*. Implementasi fungsionalitas LoRa *gateway* dijelaskan melalui *pseudocode* pada Tabel 5.4.

**Tabel 5.4 *Pseudocode* LoRa *gateway***

| *Pseudocode* LoRa *gateway* | |
|---|---|
| 1 | FUNCTION main() |

*Pseudocode* LoRa *gateway*

```
    TRY
        controller <-- konfigurasi lora
        lora <-- controller transceiver lora
        loginInfo <-- hasil pemanggilan fungsi login()
        token <-- loginInfo['token']
        PRINT "LoRa Gateway telah aktif"
        WHILE True
            IF lora menangkap paket
                topik, idSensor, dataECG <-- hasil pemanggilan fungsi getData(lora)
                t <-- thread yang menjalankan fungsi fwdToApps(topik, idSensor, dataECG, token)
                memulai t
            END-IF
        END-WHILE
    EXCEPTION
        WHEN KeyboardInterrupt
            PRINT "Program dihentikan dengan interupsi"
            membersihkan GPIO
        END-WHEN
        WHEN Exception
            PRINT "Terjadi kesalahan"
            PRINT Exception
          membersihkan GPIO
        END-WHEN
    END-TRY
END-FUNCTION
```

LoRa *gateway* pertama-tama menginisialisasi beberapa variabel yang diperlukan yaitu `controller` yang merupakan konfigurasi dari LoRa, `lora` yang merupakan *controller* dari *transceiver* LoRa, `loginInfo` yang menyimpan informasi-informasi *credential* yang digunakan untuk tersambung dengan *cloud* yang diperoleh dari pemanggilan fungsi `login()`, dan `token` yang merupakan token terautentikasi yang tersimpan di dalam `loginInfo`. Setelah variabel-variabel tersebut berhasil diinisialisasikan, LoRa *gateway* akan menampilkan pesan yang menandakan bahwa LoRa *gateway* telah aktif dan siap untuk menerima paket LoRa yang dikirimkan kepada dirinya.

Di saat LoRa *gateway* telah siap untuk menerima paket LoRa yang dikirimkan kepada dirinya, ia akan menjalankan perulangan dengan parameter `True` yang membuat perulangan tersebut tidak akan berhenti sampai terjadi sesuatu yang memaksanya untuk berhenti seperti *exception*, kesalahan, atau kehilangan sumber daya. Di dalam setiap putaran pada perulangan tersebut, LoRa *gateway* melakukan seleksi kondisi yang memeriksa apakah ada paket yang tertangkap oleh `lora` atau tidak. Apabila ada, ia akan membaca paket tersebut dengan memanggil fungsi `getData` dan menyimpan hasil pembacaan paket tersebut ke dalam variabel `topik`, `idSensor`, dan `dataECG`. Selanjutnya, LoRa *gateway* akan menginisialisasi dan memulai *thread* yang bertugas untuk meneruskan data yang diterima ke aplikasi dengan menjalankan fungsi `fwdToApps`.

Fungsionalitas LoRa *gateway* juga dilengkapi dengan *catch exception* dimana apabila terjadi kesalahan saat menjalankan programnya, LoRa *gateway* akan menampilkan pesan bahwa telah terjadi kesalahan dan informasi kesalahan yang

terjadi. Namun terdapat kasus khusus apabila yang terjadi adalah *KeyboardInterrupt* yang berarti pengguna telah memberikan sinyal *interrupt* menggunakan *keyboard*. Apabila yang terjadi adalah *exception* tersebut, maka program akan menampilkan pesan bahwa program telah dihentikan dengan interupsi dan selanjutnya akan membersihkan GPIO.

### 5.2.1 Implementasi Fungsi Autentikasi *Token* ke Aplikasi

Autentikasi *token* pada LoRa *gateway* ke aplikasi dilakukan dengan menggunakan fungsi `login` yang tidak memerlukan tambahan *parameter* untuk dipanggil. Fungsi ini melakukan autentikasi *token* dengan memanfaatkan HTTP POST untuk berkomunikasi dengan aplikasi. Fungsi ini akan dijelaskan secara spesifik melalui *pseudocode* pada Tabel 5.5.

**Tabel 5.5 *Pseudocode* fungsi login**

| *Pseudocode* fungsi login | |
|---|---|
| 1 | FUNCTION login() |
| 2 | PRINT "Menghubungkan ke aplikasi..."<br>response <-- HTTP Response dari pengiriman *request* HTTP Post ke "http://api.iotapps.belajardisini.com/user/login" yang mengandung token dan secret key untuk autentikasi |
| 3 | IF response.status_code != 200 THEN |
| | PRINT "Sys : Token dan secret key tidak cocok" |
| 4 | menghentikan program seketika |
| 5 | ELSE IF response.status_code == 200 THEN |
| 6 | loginInfo <-- hasil pembacaan response.content dari format JSON |
| 7 | |
| 8 | PRINT "Berhasil terhubung ke aplikasi\n"<br>RETURN loginInfo |
| 9 | END-IF |
| 10 | END-FUNCTION |

Pada fungsi ini, hal pertama yang dilakukan oleh LoRa *gateway* adalah menampilkan pesan *log* yang memberi informasi bahwa LoRa *gateway* sedang menghubungkan dirinya ke aplikasi yang berada pada *cloud*. Selanjutnya LoRa *gateway* mengirimkan *request* HTTP *Post* ke "http://api.iotapps.belajardisini.com/user/login" dengan menyisipkan *token* miliknya dan *secret key* yang sesuai. Balasan berupa HTTP *Response* yang didapat dari pengiriman paket HTTP *Post* tersebut akan disimpan ke dalam variabel `response`. Setelah itu, LoRa *gateway* akan memeriksa *status code* yang diterima pada pesan respon dari aplikasi. Apabila nilainya bukan 200, berarti autentikasi *token* gagal dilakukan dan pesan *log* bahwa *token* beserta *secret key* yang dikirimkan tidak cocok akan ditampilkan dan program akan berhenti seketika. Namun apabila nilainya adalah 200, berarti autentikasi *token* berhasil dilakukan dan LoRa *gateway* akan memasukkan konten yang tersimpan di dalam variabel `response` ke dalam variabel `loginInfo` lalu mengembalikan variabel tersebut ke bagian dimana fungsi `login` dipanggil. Pesan *log* juga akan ditampilkan apabila autentikasi *token* berhasil dilakukan.

### 5.2.2 Implementasi Fungsi Penerimaan Paket Data ECG

Pada LoRa *gateway*, penerimaan paket data ECG dari *node* sensor dan pembacaannya dilakukan dengan menggunakan fungsi `getData` dengan satu parameter tambahan, yaitu variabel yang menyimpan *controller* LoRa. Fungsi tersebut tidak hanya menerima paket dan membacanya, melainkan juga mengekstraksi data yang ada di dalam *payload* tersebut. Fungsi ini akan dijelaskan secara spesifik melalui *pseudocode* pada Tabel 5.6.

**Tabel 5.6 *Pseudocode* fungsi getData**

*Pseudocode* fungsi getData

```
1  FUNCTION getData(lora)
2      payload <-- hasil pembacaan payload pada lora
3      payload <-- payload setelah di-decode dan dimuat dari
   format JSON
4      topik <-- payload[0]
5      idSensor <-- payload[1]
6      dataECG <-- payload[2]
7      PRINT "Menerima paket data ECG dari node " + idSensor
8      RETURN topik, idSensor, dataECG
9  END-FUNCTION
```

Pada fungsi ini, hal pertama yang dilakukan oleh LoRa *gateway* adalah membaca *payload* yang tersimpan pada variabel `lora` dan menyimpannya ke dalam variabel `payload`. Selanjutnya variabel `payload` tersebut di-*decode* dari bentuk *byte* ke *String* dan data-data yang terkandung di dalamnya akan dimuat dari format JSON *String* untuk nantinya dikembalikan oleh fungsi ini. Data-data tersebut adalah topik, id sensor, dan data ECG. Lalu, sebelum data-data tersebut dikembalikan ke bagian dimana fungsi ini dipanggil, LoRa *gateway* akan menampilkan pesan *log* yang menginformasikan bahwa LoRa *gateway* telah menerima paket data ECG dari *node* sensor.

### 5.2.3 Implementasi Fungsi Penerusan Data ECG ke Aplikasi

Pada LoRa *gateway*, penerusan data ECG ke aplikasi dilakukan dengan menggunakan fungsi `fwdToApps` dengan empat parameter tambahan, yaitu variabel-variabel yang masing-masing menyimpan topik, id sensor, data ECG, dan *token* terautentikasi. Sama seperti fungsi `login`, penerusan data ECG ke aplikasi juga memanfaatkan metode HTTP *Post* untuk berkomunikasi dengan aplikasi. Fungsi ini akan dijelaskan secara spesifik melalui *pseudocode* pada Tabel 5.7.

**Tabel 5.7 *Pseudocode* fungsi fwdToApps**

*Pseudocode* fungsi fwdToApps

```
1  FUNCTION fwdToApps(topik, idSensor, dataECG, token)
2      appsURL                                          <--
   "http://api.iotapps.belajardisini.com/topic/"+topic
3      ecg <-- {'idSensor': idSensor, 'dataEcg': dataECG}
4      PRINT "Mengunggah data ECG ke aplikasi..."
5      response <-- HTTP Response dari pengiriman paket HTTP
   POST ke appsURL dengan ecg dalam format JSON sebagai data
   yang disisipkan dan token sebagai header otorisasi
6      WHILE r.status_code == 401
```

```
Pseudocode fungsi fwdToApps
        PRINT "Token telah expired. Mencoba menghubungkan
kembali ke aplikasi...\n"
        loginInfo <-- hasil pemanggilan fungsi login()
        token <-- loginInfo['token']
        PRINT "Mencoba mengunggah data kembali..."
        response <-- HTTP Response dari pengiriman paket
HTTP POST ke appsURL dengan ecg dalam format JSON sebagai
data yang disisipkan dan token sebagai header otorisasi
    END-WHILE
    PRINT "Data berhasil diunggah ke aplikasi\n"
END-FUNCTION
```

Pada fungsi ini, hal pertama yang dilakukan oleh LoRa *gateway* adalah menginisialisasikan URL aplikasi sesuai dengan topik yang ada pada *parameter* fungsi. URL tersebut dimasukkan ke dalam variabel `appsURL` yang nantinya akan digunakan sebagai tujuan pengunggahan data ECG. Setelah itu, LoRa *gateway* juga menginisialisasikan sebuah *dictionary* yang terdiri dari dua indeks, yaitu `'idSensor'` untuk menampung id sensor pada *parameter* fungsi dan `'dataEcg'` untuk menampung data ECG yang juga ada pada *parameter* fungsi. *Dictionary* tersebut lalu dimasukkan ke dalam variabel `ecg`. Setelah inisialisasi kedua variabel tersebut selesai dilakukan, LoRa *gateway* akan mengirimkan *request* HTTP *Post* yang membawa variabel `ecg` sebagai datanya dan variabel `token` yang ada pada *parameter* fungsi sebagai *header* otorisasinya. Respon dari pengiriman *request* HTTP *Post* tersebut berbentuk HTTP *Response* dan akan disimpan ke dalam variabel `response`. Pada saat pengiriman *request* HTTP *Post*, akan ditampilkan pesan *log* yang menginformasikan bahwa LoRa *gateway* mengunggah data ke aplikasi.

Setelah respon dari aplikasi sudah tersimpan, akan terjadi sebuah perulangan yang hanya akan terus berjalan apabila kode status dari respon aplikasi yang didapat adalah 401. Arti dari kode status tersebut adalah *server*, dalam hal ini aplikasi, tidak mengizinkan pengirim *request* untuk mengakses *resource* yang diminta karena gagal menyertakan bukti autentikasi (*token*) yang sah. Dalam perulangan tersebut, autentikasi ulang akan dilakukan dan data yang sama akan diunggah kembali ke aplikasi namun dengan menyertakan *token* yang sudah diautentikasi ulang. Dalam setiap satu kali perulangan ini, LoRa *gateway* akan menampilkan pesan *log* yang menginformasikan bahwa pengunggahan data ke aplikasi gagal dilakukan karena *token* telah *expired* sehingga dilakukan autentikasi dan pengunggahan data ulang. Di akhir fungsi, apabila data telah berhasil diunggah ke aplikasi, pesan *log* bahwa pengunggahan data berhasil dilakukan akan ditampilkan.

# BAB 6 PENGUJIAN DAN PEMBAHASAN

Bab ini menjelaskan tentang pengujian-pengujian yang dilakukan terhadap sistem yang telah diimplementasikan dan membahas hasil dari pengujian tersebut. Pengujian dilakukan berdasarkan perancangan pengujian yang telah didefinisikan pada bab perancangan. Pengujian-pengujian tersebut meliputi pengujian fungsionalitas sistem dan pengujian kinerja sistem.

## 6.1 Pengujian Fungsionalitas Sistem

Pengujian fungsionalitas sistem ini menguji apakah fungsionalitas-fungsionalitas yang harus dapat dijalankan oleh sistem dapat berjalan sesuai dengan hasil yang diharapkan. Pengujian ini menentukan apakah sistem dapat mengerjakan fungsinya dengan baik atau tidak. Pengujian ini dilaksanakan pada *node* sensor dan LoRa *gateway*.

### 6.1.1 Pengujian Kemampuan *Node* Sensor Merekam Data ECG dari Pengguna

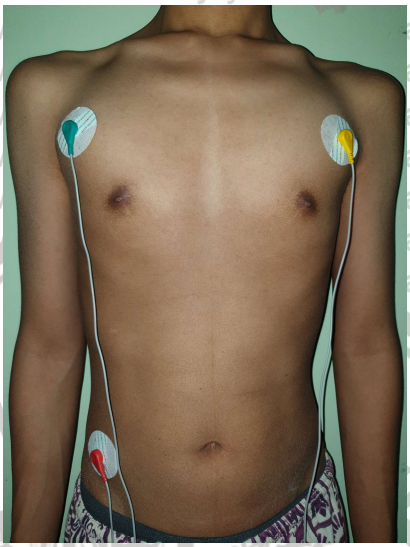
Tabel 6.1 berikut ini menunjukkan skenario kasus uji untuk pengujian kemampuan *node* sensor merekam data ECG dari pengguna.

**Tabel 6.1 Pengujian kemampuan *node* sensor merekam data ECG dari pengguna**

| **Kode** | **PF_01** |
|---|---|
| Nama Kasus Uji | Kemampuan *node* sensor merekam data ECG dari pengguna. |
| Tujuan Pengujian | Mengetahui apakah *node* sensor mampu merekam aktivitas listrik yang dihasilkan oleh jantung pengguna menggunakan sensor ECG. |
| Prosedur Pengujian | 1. Diketahui ketiga elektroda pada modul sensor telah melekat masing-masing pada dada kiri, dada kanan, dan pinggang kanan pengguna.<br>2. Diketahui *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB.<br>3. Laptop memerintahkan *node* sensor untuk menjalankan program perekaman data ECG dari pengguna.<br>4. Data ECG yang berhasil direkam akan disimpan ke dalam file .txt pada *node* sensor. |
| Hasil yang Diharapkan | *Node* sensor berhasil menyimpan data hasil rekam ECG ke dalam file .txt. |
| Hasil Pengujian | VALID |

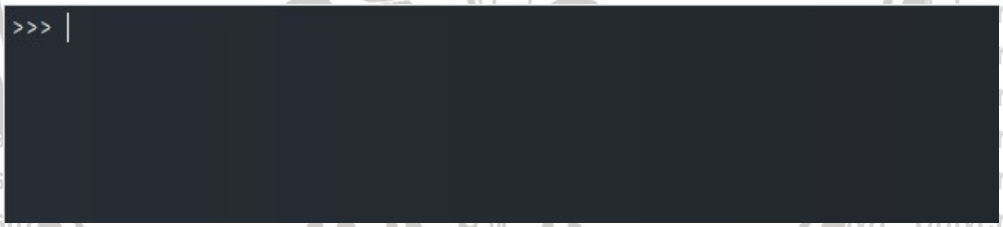
Pertama-tama ketiga elektroda modul sensor harus dipastikan melekat masing-masing pada dada kiri, dada kanan, dan pinggang kanan pengguna. Hal ini dilakukan untuk memastikan modul sensor dapat menangkap sinyal ECG dari

pengguna dengan benar. Gambar 6.1 menunjukkan posisi pemasangan ketiga elektroda modul sensor pada badan pengguna.



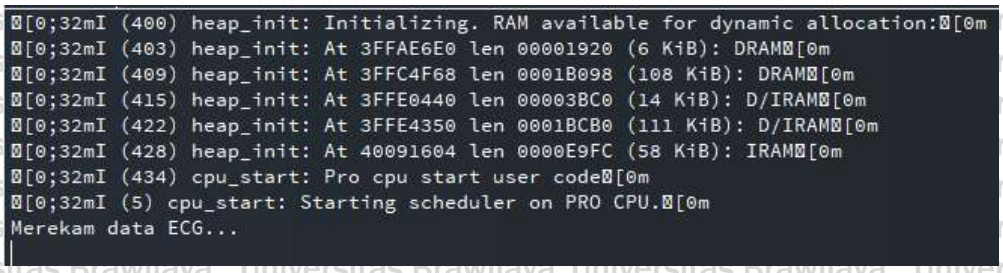
**Gambar 6.1 Pemasangan elektroda pada badan pengguna**

Setelah itu, harus dipastikan *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB. Hal ini dilakukan agar *node* sensor dapat dikontrol menggunakan laptop. Gambar 6.2 menunjukkan bahwa *node* sensor telah hidup dan tersambung dengan laptop.



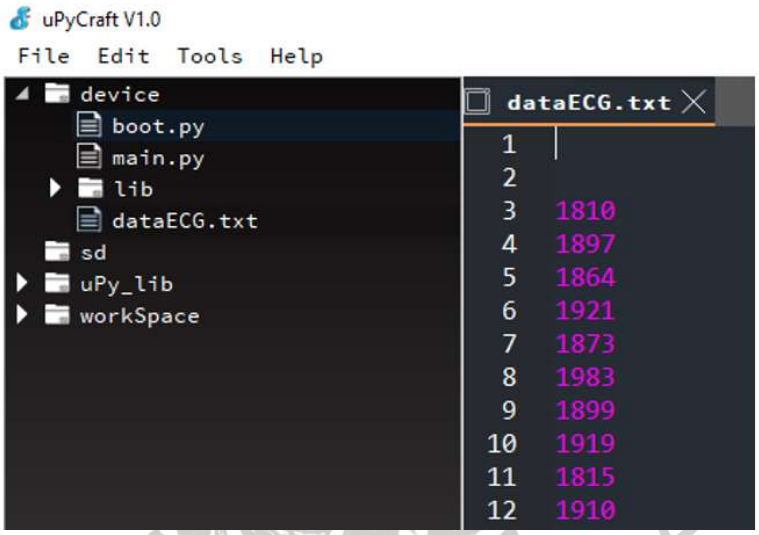

**Gambar 6.2 Tampilan *node* sensor pada laptop**

Selanjutnya laptop memerintahkan *node* sensor untuk menjalankan program perekaman data ECG dari pengguna. Hal ini dilakukan untuk memulai perekaman data ECG dari pengguna melalui *node* sensor. Gambar 6.3 menunjukkan bahwa *node* sensor telah berhasil memulai program perekaman data ECG dari pengguna.




**Gambar 6.3 Tampilan *node* sensor saat perekaman data dilakukan**

Setelah program perekaman data ECG pengguna berhasil dijalankan tanpa kesalahan, maka *node* sensor akan menyimpan data hasil rekam tersebut ke dalam sebuah *file* .txt. Data hasil rekam ECG tersebut dapat dilihat pada *file* dataECG.txt yang ada pada *node* sensor. Gambar 6.4 menunjukkan isi dari *file* dataECG.txt pada *node* sensor.




**Gambar 6.4 Isi *file* dataECG.txt**

### 6.1.2 Pengujian Kemampuan *Node* Sensor Membungkus Data ECG dengan Topik dan ID Sensor dalam Format JSON

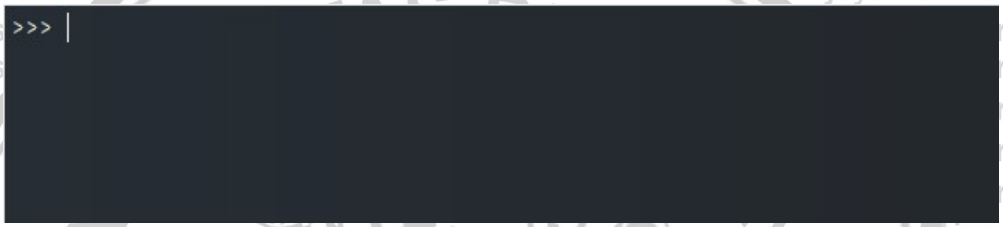
Tabel 6.2 berikut ini menjelaskan skenario kasus uji untuk pengujian kemampuan *node* sensor membungkus data ECG dengan topik dan id sensornya dalam format JSON.

**Tabel 6.2 Pengujian kemampuan *node* sensor membungkus data ECG dengan topik dan id sensor dalam format JSON**

| **Kode** | **PF_02** |
|---|---|
| Nama Kasus Uji | Kemampuan *node* sensor membungkus data ECG dengan topik yang sesuai beserta id sensornya dalam format JSON. |
| Tujuan Pengujian | Mengetahui apakah *node* sensor mampu membungkus data ECG dengan topik yang sesuai beserta id sensornya kedalam format JSON. |
| Prosedur Pengujian | - Diketahui *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB.<br>- Diketahui *node* sensor telah menyimpan *file* yang berisi rekaman data ECG pengguna selama 10 detik yang dihasilkan dari pengujian dengan kode PF_01.<br>- Laptop memerintahkan *node* sensor untuk menjalankan program pembungkusan data ECG pengguna dengan topik dan id sensor. |

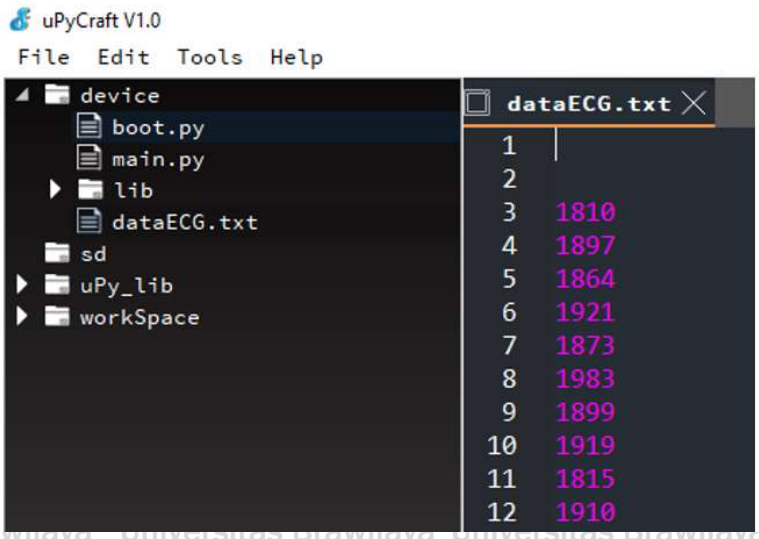
| Kode | PF_02 |
|---|---|
| | - Hasil pembungkusan data ECG pengguna dengan topik dan id sensor akan ditampilkan pada layar laptop. |
| Hasil yang Diharapkan | *Node* sensor berhasil menampilkan data hasil pembungkusan data ECG beserta topik dan id sensor yang sudah berformat JSON. |
| Hasil Pengujian | VALID |

Pertama-tama harus dipastikan *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB. Hal ini dilakukan agar *node* sensor mendapatkan sumber daya dan dapat dikontrol menggunakan laptop. Gambar 6.5 menunjukkan bahwa *node* sensor telah hidup dan tersambung dengan laptop.




**Gambar 6.5 Tampilan *node* sensor pada laptop**

Setelah itu, harus dipastikan *node* sensor telah menyimpan *file* yang berisi rekaman data ECG pengguna selama 10 detik yang dihasilkan dari pengujian kode PF_01. Hal ini dilakukan karena data ECG yang akan dibungkus nantinya adalah data hasil rekam ECG. Gambar 6.6 menunjukkan *file* yang berisi rekaman data ECG pengguna selama 10 detik yang dihasilkan dari pengujian PF_01.




**Gambar 6.6 *File* yang berisi data hasil rekam ECG pengguna**

Selanjutnya, laptop memerintahkan *node* sensor untuk menjalankan program pembungkusan data ECG pengguna dengan topik dan id sensor. Hal ini dilakukan untuk memulai pembungkusan data pada *node* sensor. Apabila program

pembungkusan data ECG pengguna berhasil dijalankan tanpa kesalahan, maka *node* sensor akan menampilkan data hasil pembungkusan tersebut pada layar laptop. Data hasil pembungkusan tersebut dapat dilihat pada layar laptop secara langsung setiap kali setelah pembungkusan dilakukan. Gambar 6.7 menunjukkan tampilan saat pembungkusan data ECG selesai dilakukan.

```
["ecg", "0001", [1810, 1897, 1864, 1921, 1873, 1983, 1899, 1919, 1815, 1910, 1803,
1868, 1813, 2114, 1818, 1875, 1786, 1987, 2623, 2995, 2193, 1933, 1840, 1804, 1801,
1787, 1840, 1776, 1854, 1824, 1875, 1842, 1926, 1835, 1936, 1904, 1990, 1904, 2029]]
```

**Gambar 6.7 Hasil pembungkusan data ECG pengguna**

### 6.1.3 Pengujian Kemampuan *Node* Sensor Mengirim Data ECG Terformat ke LoRa *Gateway* Melalui LoRa

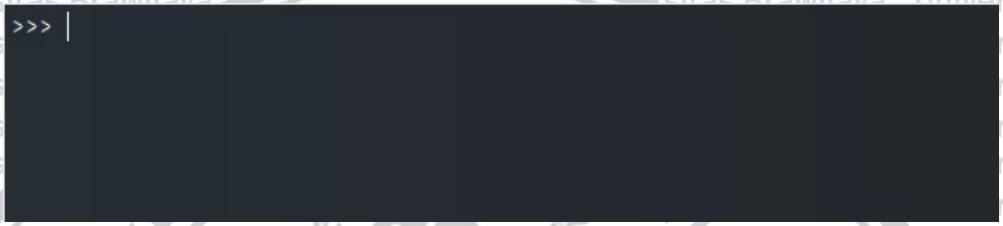
Tabel 6.3 berikut ini menjelaskan skenario kasus uji untuk pengujian kemampuan *node* sensor mengirim data ECG, topik, dan id sensor yang telah dibungkus dalam format JSON ke LoRa *gateway* melalui LoRa.

**Tabel 6.3 Pengujian kemampuan *node* sensor mengirim data ECG terformat ke LoRa *gateway* melalui LoRa**

| **Kode** | **PF_03** |
|---|---|
| Nama Kasus Uji | Kemampuan *node* sensor mengirimkan data ECG, topik, dan id sensor yang telah dibungkus dalam format JSON ke LoRa *gateway* melalui LoRa. |
| Tujuan Pengujian | Mengetahui apakah *node* sensor mampu mengirim data ECG, topik, dan id sensor dalam format JSON ke LoRa *gateway* melalui LoRa. |
| Prosedur Pengujian | 1. Diketahui *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB.<br>2. Diketahui LoRa *gateway* dalam keadaan hidup dan telah tersambung dengan laptop menggunakan SSH melalui jaringan WiFi.<br>3. Diketahui *node* sensor telah berhasil membungkus data dengan topik dan id sensor dengan format JSON sesuai pengujian kode PF_02.<br>4. Laptop memerintahkan LoRa *gateway* untuk menjalankan program penerimaan data ECG pengguna.<br>5. Laptop memerintahkan *node* sensor untuk menjalankan program pengiriman data ECG pengguna.<br>6. Pesan berhasil akan ditampilkan apabila data ECG pengguna dalam format JSON telah berhasil dikirim dan data yang diterima oleh LoRa *gateway* yang berisi data ECG beserta topik dan id sensornya akan ditampilkan pada layar laptop. |

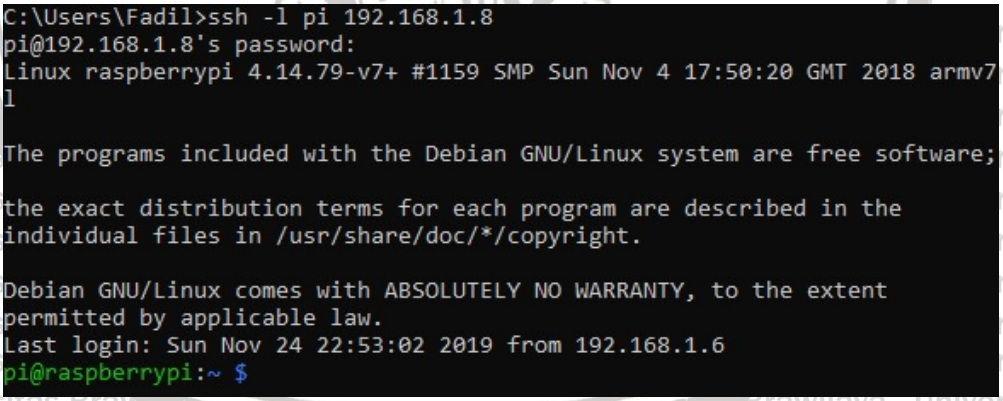
| | |
|---|---|
| Hasil yang Diharapkan | *Node* sensor menampilkan pesan bahwa data berhasil dikirim dan LoRa *gateway* menampilkan data ECG pengguna beserta dengan topik dan id sensornya |
| Hasil Pengujian | VALID |

Pertama-tama harus dipastikan *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB. Hal ini dilakukan agar *node* sensor dipastikan mendapatkan sumber data dan dapat dikontrol menggunakan laptop. Gambar 6.8 menunjukkan bahwa *node* sensor telah hidup dan tersambung dengan laptop.



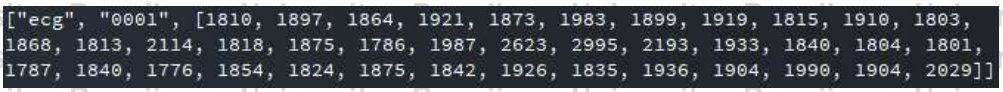

**Gambar 6.8 Tampilan *node* sensor pada laptop**

Selain itu, harus dipastikan juga LoRa *gateway* sudah dalam keadaan hidup dan telah tersambung dengan laptop menggunakan SSH melalui jaringan WiFi. Hal ini dilakukan agar LoRa *gateway* dipastikan mendapat sumber daya dan dapat dikontrol menggunakan laptop. Gambar 6.9 menunjukkan bahwa LoRa *gateway* telah berada dalam keadaan hidup dan tersambung dengan laptop.




**Gambar 6.9 Tampilan LoRa *gateway* pada SSH laptop**

Selain itu, harus dipastikan juga *node* sensor telah berhasil membungkus data dengan topik dan id sensor dengan format JSON sesuai pengujian kode PF_02. Hal ini dilakukan karena nantinya data yang akan dikirimkan oleh *node* sensor adalah berupa data JSON yang berisi data ECG, topik, dan id sensor seperti yang dibuat pada pengujian PF_02. Gambar 6.10 merupakan contoh dari data JSON yang berisi data ECG, topik, dan id sensor.




**Gambar 6.10 Data JSON yang berisi data ECG, topik, dan id sensor**

Setelah itu, laptop memerintahkan LoRa *gateway* untuk menjalankan program penerimaan data ECG pengguna. Hal ini dilakukan karena untuk memastikan keberhasilan pengiriman data, data yang dikirim harus dapat dilihat atau ditampilkan oleh penerima. Gambar 6.11 menunjukkan tampilan saat program penerimaan data pada LoRa *gateway* sudah berjalan.

```
pi@raspberrypi:~ $ cd Pengujian\ Fungsional/
pi@raspberrypi:~/Pengujian Fungsional $ python3 receiveFromLoRa.py
Menunggu paket LoRa...
```

**Gambar 6.11 Tampilan pada LoRa *gateway* saat program penerimaan berjalan**

Selanjutnya, laptop memerintahkan *node* sensor untuk menjalankan program pengiriman data ECG pengguna. Hal ini dilakukan untuk memulai pengiriman data ECG pengguna dari *node* sensor ke LoRa *gateway* melalui LoRa dan akan menampilkan format data yang dikirimkan. Gambar 6.12 menunjukkan tampilan saat program pengiriman data pada *node* sensor sudah berjalan dan telah mengirimkan data.

```
[0;32mI (4093) network: GOT_IP[0m
Sending packet:
["ecg", "0001", [1810, 1897, 1864, 1921, 1873, 1983, 1899, 1919, 1815, 1910, 1803,
1868, 1813, 2114, 1818, 1875, 1786, 1987, 2623, 2995, 2193, 1933, 1840, 1804, 1801,
1787, 1840, 1776, 1854, 1824, 1875, 1842, 1926, 1835, 1936, 1904, 1990, 1904, 2029]]

[Memory - free: 88736   allocated: 22432]
```

**Gambar 6.12 Tampilan pada *node* sensor saat program pengiriman berjalan**

Terakhir, untuk memastikan bahwa data telah berhasil dikirim, data yang diterima oleh LoRa *gateway* akan ditampilkan. Hal tersebut dapat dilihat pada tampilan SSH LoRa *gateway.* Gambar 6.13 menunjukkan data yang diterima pada LoRa *gateway* setelah program pengiriman data ECG pengguna pada *node* sensor dijalankan.

```
pi@raspberrypi:~/Pengujian Fungsional $ python3 receiveFromLoRa.py
Menunggu paket LoRa...
Menerima paket LoRa
ID Sensor: 0001
Topik    : ecg
Data ECG : [1810, 1897, 1864, 1921, 1873, 1983, 1899, 1919, 1815, 1910, 1803,
1868, 1813, 2114, 1818, 1875, 1786, 1987, 2623, 2995, 2193, 1933, 1840, 1804,
1801, 1787, 1840, 1776, 1854, 1824, 1875, 1842, 1926, 1835, 1936, 1904, 1990,
1904, 2029]
```

**Gambar 6.13 Data yang diterima pada LoRa *gateway***

### 6.1.4 Pengujian Kemampuan LoRa *Gateway* Menerima Data ECG Terformat yang Dikirim *Node* Sensor Melalui LoRa

Tabel 6.4 berikut ini menjelaskan skenario kasus uji untuk pengujian kemampuan LoRa *gateway* menerima dan membaca data berformat JSON yang berisi data ECG, topik, dan id sensor yang telah dikirimkan oleh *node* sensor melalui LoRa.

**Tabel 6.4 Pengujian kemampuan LoRa *gateway* menerima data ECG terformat yang dikirim *node* sensor melalui LoRa**

| Kode | PF_04 |
|---|---|
| Nama Kasus Uji | Kemampuan LoRa *gateway* menerima data JSON yang berisi data ECG, topik, dan id sensor yang telah dikirimkan oleh *node* sensor melalui LoRa. |
| Tujuan Pengujian | Mengetahui apakah LoRa *gateway* mampu menerima data JSON yang berisi data ECG, topik, dan id sensor yang telah dikirimkan oleh *node* sensor melalui LoRa. |
| Prosedur Pengujian | 1. Diketahui *node* sensor dalam keadaan hidup dan telah tersambung dengan laptop melalui *port* USB.<br>2. Diketahui LoRa *gateway* dalam keadaan hidup dan telah tersambung dengan laptop menggunakan SSH melalui jaringan WiFi.<br>3. Diketahui *node* sensor telah menyimpan data ECG pengguna yang berformat JSON yang diperoleh dari pengujian dengan kode PF_02.<br>4. Laptop memerintahkan LoRa *gateway* untuk menjalankan program penerimaan data ECG pengguna.<br>5. Laptop memerintahkan *node* sensor untuk menjalankan program pengiriman data ECG pengguna.<br>6. Pesan berhasil akan ditampilkan apabila data ECG pengguna dalam format JSON telah berhasil dikirim dan data yang diterima oleh LoRa *gateway* yang berisi data ECG beserta topik dan id sensornya akan ditampilkan pada layar laptop. |
| Hasil yang Diharapkan | *Node* sensor menampilkan pesan bahwa data berhasil dikirim dan LoRa *gateway* menampilkan data ECG pengguna beserta dengan topik dan id sensornya. |
| Hasil Pengujian | VALID |

Sesuai dengan hasil pengujian PF_03 yang telah dilakukan dan telah dijabarkan sebelumnya, *node* sensor dapat mengirimkan data ECG terformat ke LoRa *gateway* dan LoRa *gateway* dapat menerima data tersebut. Oleh karena itu, hasil dari pengujian ini juga dinilai sudah valid.

### 6.1.5 Pengujian Kemampuan LoRa *Gateway* Mengautentikasikan *Token*-nya ke Aplikasi

Tabel 6.5 berikut ini menjelaskan skenario kasus uji untuk pengujian kemampuan LoRa *gateway* mengautentikasikan *token* miliknya ke aplikasi pada *cloud* dengan menggunakan protokol HTTP.

**Tabel 6.5 Pengujian kemampuan LoRa *gateway* mengautentikasikan *token*-nya ke aplikasi**

| Kode | PF_05 |
|---|---|
| Nama Kasus Uji | Kemampuan LoRa *gateway* mengautentikasikan *token* miliknya ke aplikasi pada *cloud* dengan menggunakan protokol HTTP |
| Tujuan Pengujian | Mengetahui apakah LoRa *gateway* mampu mengautentikasikan *token* miliknya ke aplikasi pada *cloud* dengan menggunakan protokol HTTP |
| Prosedur Pengujian | 1. Diketahui LoRa *gateway* telah dalam keadaan hidup dan tersambung dengan laptop menggunakan SSH melalui jaringan WiFi.<br>2. Diketahui LoRa *gateway* telah memiliki *token* yang terdaftar pada aplikasi.<br>3. Laptop memerintahkan LoRa *gateway* untuk menjalankan program autentikasi *token*.<br>4. Pesan berhasil dan *token* yang telah terautentikasi akan ditampilkan pada layar laptop apabila autentikasi *token* berhasil dilakukan. |
| Hasil yang Diharapkan | LoRa *gateway* menampilkan pesan bahwa *token* telah berhasil diautentikasi dan LoRa *gateway* menampilkan *token* yang telah terautentikasi |
| Hasil Pengujian | VALID |

Pertama-tama, harus dipastikan LoRa *gateway* telah dalam keadaan hidup dan telah tersambung dengan laptop menggunakan SSH melalui jaringan WiFi. Hal ini dilakukan agar LoRa *gateway* dipastikan mendapat sumber daya dan dapat dikontrol menggunakan laptop. Gambar 6.14 menunjukkan bahwa LoRa *gateway* telah berada dalam keadaan hidup dan tersambung dengan laptop.

```
C:\Users\Fadil>ssh -l pi 192.168.1.8
pi@192.168.1.8's password:
Linux raspberrypi 4.14.79-v7+ #1159 SMP Sun Nov 4 17:50:20 GMT 2018 armv7
l

The programs included with the Debian GNU/Linux system are free software;

the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Nov 24 22:53:02 2019 from 192.168.1.6
pi@raspberrypi:~ $
```

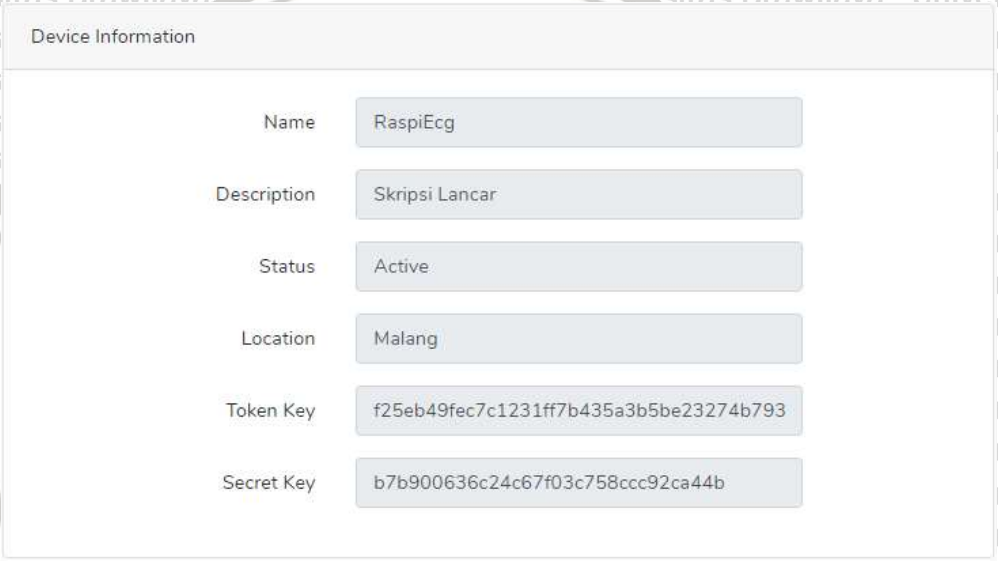
**Gambar 6.14 Tampilan LoRa *gateway* pada SSH laptop**

Selain itu, harus dipastikan juga LoRa *gateway* telah menyimpan *token* dan *secret key* yang terdaftar pada aplikasi. Hal ini dilakukan karena nantinya *token* dan *secret key* tersebut akan dikirimkan ke aplikasi untuk autentikasi. Gambar 6.15

menunjukkan *token* dan *secret key* yang tersimpan pada LoRa *gateway* dan Gambar 6.16 menunjukkan *token* dan *secret key* yang terdaftar pada aplikasi yang dapat dilihat pada halaman informasi perangkat yang ada pada domain iotapps.belajardisini.com.

```
pi@raspberrypi:~/Pengujian Fungsional $ python3 showToken.py
Token      : f25eb49fec7c1231ff7b435a3b5be23274b793df
Secret key: b7b900636c24c67f03c758ccc92ca44b
pi@raspberrypi:~/Pengujian Fungsional $
```

**Gambar 6.15 *Token* dan *secret key* pada LoRa *gateway***



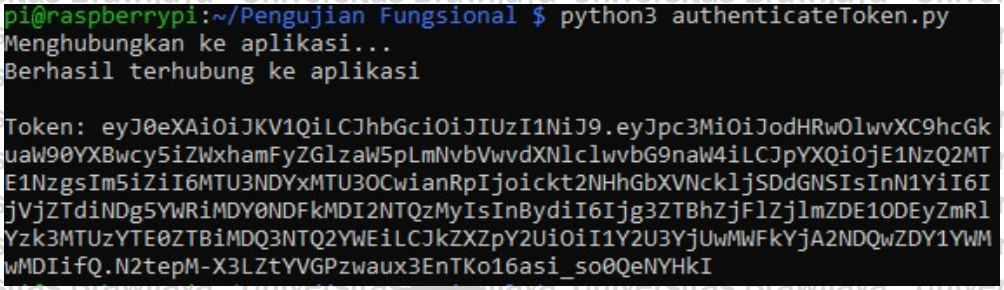

**Gambar 6.16 *Token* dan *secret key* pada aplikasi**

Setelah itu, laptop memerintahkan LoRa *gateway* untuk menjalankan program autentikasi *token*. Hal ini dilakukan untuk memulai proses autentikasi *token* dengan cara mengirimkan *request* HTTP *Post* yang menyertakan *token* dan *secret key* yang terdaftar pada aplikasi. Gambar 6.17 menunjukkan tampilan saat program proses autentikasi sedang berjalan.

```
pi@raspberrypi:~/Pengujian Fungsional $ python3 authenticateToken.py
Menghubungkan ke aplikasi...
```

**Gambar 6.17 Tampilan pada LoRa *gateway* saat autentikasi dilakukan**

Terakhir, apabila *token* berhasil terautentikasi tanpa kesalahan, maka LoRa *gateway* akan menampilkan pesan bahwa *token* telah terautentikasi dan *token* yang telah terautentikasi tersebut juga akan ditampilkan. Gambar 6.18 menunjukkan tampilan saat autentikasi *token* selesai dilakukan yang berisi pesan berhasil dan token terbaru.




**Gambar 6.18 Tampilan pesan berhasil dan *token* terautentikasi**

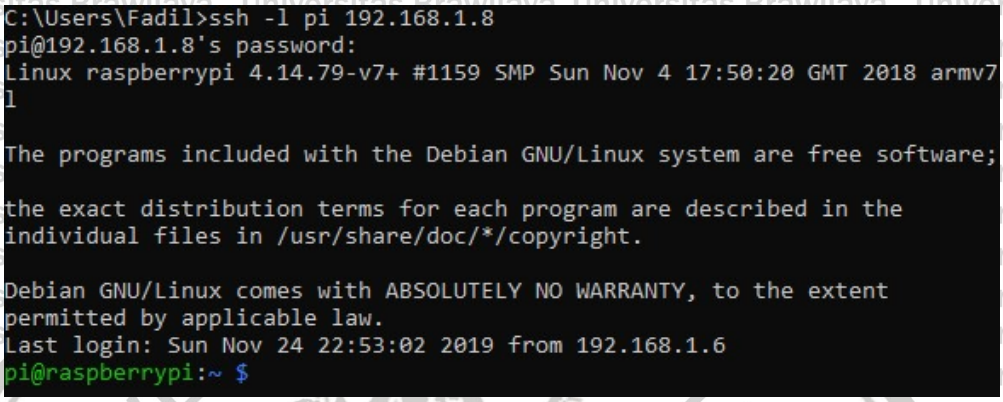
### 6.1.6 Pengujian Kemampuan LoRa *Gateway* Meneruskan Data ECG Beserta ID Sensor Sesuai dengan Topiknya Menuju ke Aplikasi

Tabel 6.6 berikut ini menjelaskan skenario kasus uji untuk pengujian kemampuan LoRa *gateway* meneruskan data ECG beserta id sensor sesuai dengan topiknya ke aplikasi pada *cloud* menggunakan protokol HTTP.

**Tabel 6.6 Pengujian kemampuan LoRa *gateway* meneruskan data ECG beserta id sensor sesuai dengan topiknya menuju ke aplikasi**

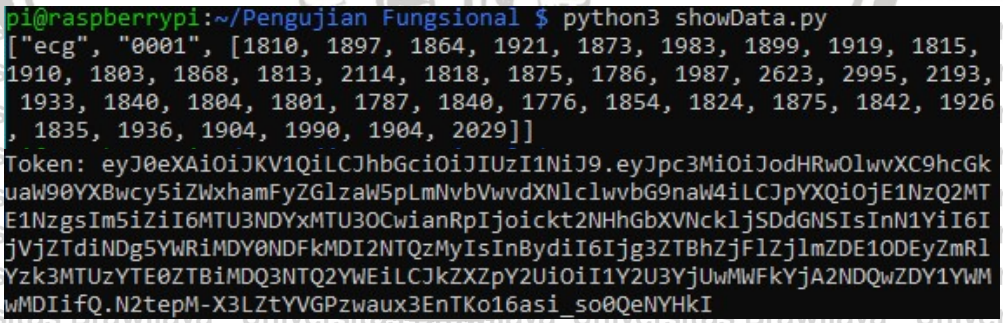
| **Kode** | **PF_06** |
|---|---|
| Nama Kasus Uji | Kemampuan LoRa *gateway* meneruskan data ECG beserta id sensor sesuai dengan topiknya ke aplikasi pada *cloud* menggunakan protokol HTTP. |
| Tujuan Pengujian | Mengetahui apakah LoRa *gateway* mampu meneruskan data ECG beserta id sensor sesuai dengan topiknya ke aplikasi pada *cloud* menggunakan protokol HTTP. |
| Prosedur Pengujian | 1. Diketahui LoRa *gateway* dalam keadaan hidup dan telah tersambung dengan laptop menggunakan SSH melalui jaringan WiFi.<br>2. Diketahui LoRa *gateway* telah menyimpan data ECG pengguna, id sensor, dan topiknya yang telah dikirimkan oleh *node* sensor pada pengujian PF_03 dan *token* yang telah terautentikasi pada pengujian PF_05.<br>3. Laptop memerintahkan LoRa *gateway* untuk menjalankan program penerusan data ECG pengguna.<br>4. Pesan berhasil akan ditampilkan pada layar laptop apabila data ECG pengguna telah berhasil diteruskan ke aplikasi.<br>5. Data ECG pengguna yang telah berhasil diteruskan ke aplikasi dapat diakses pada domain iotapps.belajardisini.com. |
| Hasil yang Diharapkan | LoRa *gateway* berhasil menampilkan pesan bahwa data berhasil diteruskan menuju ke aplikasi pada *cloud* dan data yang diteruskan dapat dilihat melalui domain iotapps.belajardisini.com. |
| Hasil Pengujian | VALID |

Pertama-tama, harus dipastikan LoRa *gateway* sudah dalam keadaan hidup dan telah tersambung dengan laptop menggunakan SSH melalui jaringan WiFi. Hal ini dilakukan agar LoRa *gateway* dipastikan memiliki sumber daya dan dapat dikontrol menggunakan laptop. Gambar 6.19 menunjukkan bahwa LoRa *gateway* telah hidup dan tersambung dengan laptop.




**Gambar 6.19 Tampilan LoRa *gateway* pada tampilan SSH laptop**

Selain itu, harus dipastikan juga LoRa *gateway* telah menyimpan data ECG pengguna, id sensor, dan topiknya yang telah dikirimkan oleh *node* sensor pada pengujian PF_03 dan *token* yang telah terautentikasi pada pengujian PF_05. Hal ini dilakukan karena nantinya data yang akan dikirim ke aplikasi adalah data yang telah dikirimkan pada pengujian PF_03 dan diperlukan penyertaan *token* yang telah terautentikasi seperti yang telah didapatkan dari pengujian PF_05 pada header otorisasinya. Gambar 6.20 menunjukkan data yang akan dikirimkan dan *token* yang akan disertakan pada *header* otorisasinya.



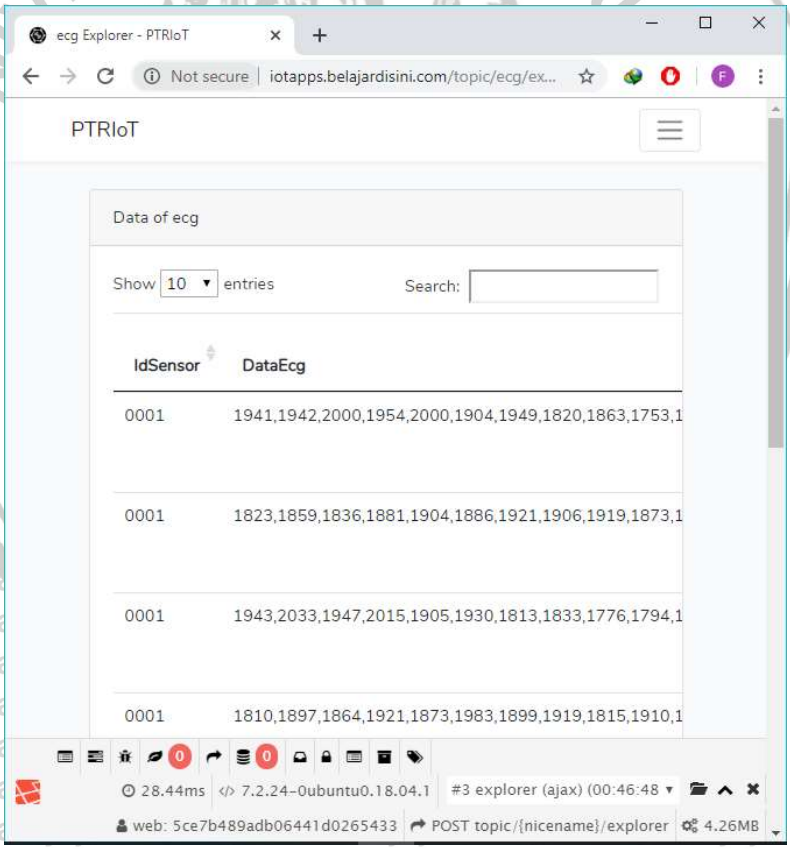

**Gambar 6.20 Data dan *token* yang akan dikirimkan oleh LoRa *gateway***

Setelah itu, laptop memerintahkan LoRa *gateway* untuk menjalankan program penerusan data ECG pengguna ke aplikasi. Hal ini dilakukan untuk memulai pengiriman data ECG terformat yang telah diterima pada pengujian PF_04 ke aplikasi. Pesan berhasil akan ditampilkan pada layar laptop apabila data ECG pengguna telah berhasil diteruskan ke aplikasi. Gambar 6.21 menunjukkan tampilan pesan yang muncul setelah data berhasil diteruskan ke aplikasi.

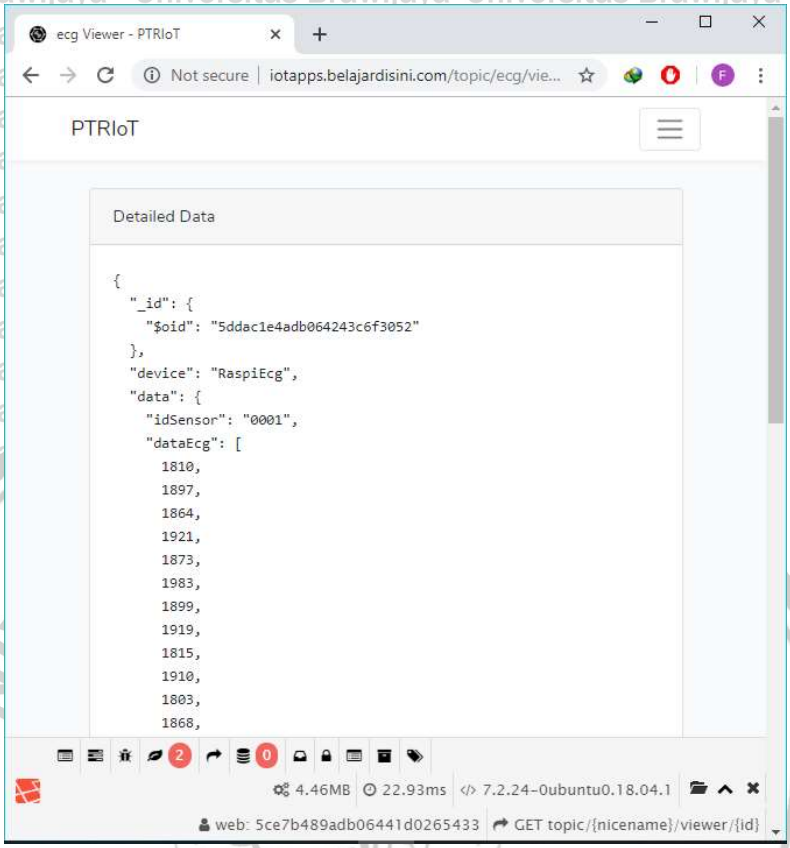
```
pi@raspberrypi:~/Pengujian Fungsional $ python3 forwardToApps.py
Mengunggah data ECG ke aplikasi...
Tujuan: http://api.iotapps.belajardisini.com/topic/ecg
Data  : {'idSensor': '0001', 'dataEcg': [1810, 1897, 1864, 1921, 1873,
1983, 1899, 1919, 1815, 1910, 1803, 1868, 1813, 2114, 1818, 1875, 1786,
 1987, 2623, 2995, 2193, 1933, 1840, 1804, 1801, 1787, 1840, 1776, 1854
, 1824, 1875, 1842, 1926, 1835, 1936, 1904, 1990, 1904, 2029]}
Data berhasil diunggah ke aplikasi
```

**Gambar 6.21 Pesan bahwa data berhasil diteruskan ke aplikasi**

Terakhir, karena data ECG pengguna yang telah berhasil diteruskan ke aplikasi pada *cloud*, seharusnya data tersebut dapat diakses pada domain iotapps.belajardisini.com. Sehingga, dilakukan percobaan pengaksesan data yang telah diunggah oleh LoRa *gateway* dengan cara mengakses halaman data yang bertopik 'ecg' pada domain iotapps.belajardisini.com dengan menggunakan *web browser*. Gambar 6.22 merupakan tampilan dari halaman yang diakses tersebut dan Gambar 6.23 merupakan tampilan detil dari salah satu *entry* data pada situs web tersebut.




**Gambar 6.22 Tampilan iotapps.belajardisini.com pada topik 'ecg'**




**Gambar 6.23 Tampilan detil data pada iotapps.belajardisini.com**

## 6.2 Pengujian Kinerja Sistem

Pengujian kinerja sistem menguji parameter-parameter kinerja pada kinerja sistem yang telah diimplementasikan. Pengujian ini dilakukan untuk mengetahui kualitas dari kinerja sistem apakah dapat memenuhi tujuan atau tidak.

### 6.2.1 Pengujian *Delay* Pengiriman Data ECG dari *Node* Sensor ke LoRa *Gateway* Melalui LoRa

Tabel 6.7 berikut ini menjelaskan skenario kasus uji untuk pengujian *delay* pengiriman data ECG dari *node* sensor ke LoRa *gateway* melalui LoRa.

**Tabel 6.7 Pengujian *delay* pengiriman data ECG dari *node* sensor ke LoRa *gateway* melalui LoRa**

| **Kode** | **PK_01** |
|---|---|
| Aspek | Kinerja |
| Nama Kasus Uji | *Delay* pengiriman data ECG dari *node* sensor ke LoRa *gateway* melalui LoRa. |
| Tujuan Pengujian | Mengetahui *delay* pengiriman data ECG dari *node* sensor ke LoRa *gateway* melalui LoRa. |

| Kode | PK_01 |
|---|---|
| Prosedur Pengujian | Pengujian dilakukan dengan cara mengirimkan satu paket data ECG sekunder (39 data) ke LoRa *gateway* menggunakan LoRa dan merekam *log* waktu pengiriman dari *node* sensor dan penerimaan pada LoRa *gateway* dengan menggunakan *Network Time Protocol* dan disimpan dalam format *file* .csv pada LoRa *gateway*. Pengiriman paket dilakukan sebanyak 60 kali dengan interval 5 detik. Pengujian dilakukan 8 kali dengan jarak antara *node* sensor dengan LoRa *gateway* sebagai pembeda. Jarak-jarak tersebut adalah 100 meter, 200 meter, 300 meter, 400 meter, 500 meter, 600 meter, 700 meter, dan 800 meter. Untuk memperoleh nilai *network delay*-nya, dilakukan operasi pengurangan antara waktu penerimaan data pada LoRa *gateway* dan waktu pengiriman data dari *node* sensor |

Hasil pengujian *delay* pengiriman data dari *node* sensor ke LoRa *gateway* melalui LoRa dapat dilihat pada Tabel 6.8.

**Tabel 6.8 Hasil pengujian *delay* pengiriman data melalui LoRa**

| No. | Jarak | *Delay* pengiriman data melalui LoRa (detik) | | |
|---|---|---|---|---|
| | | Rata-rata | Nilai tertinggi | Nilai terendah |
| 1. | 100 m | 0,299736 | 0,620823 | 0,216548 |
| 2. | 200 m | 0,306194 | 0,971474 | 0,228347 |
| 3. | 300 m | 0,334344 | 0,596061 | 0,220704 |
| 4. | 400 m | 0,376512 | 0,993208 | 0,310993 |
| 5. | 500 m | 0,432373 | 0,956751 | 0,24418 |
| 6. | 600 m | 0,462563 | 1,191328 | 0,253237 |
| 7. | 700 m | 0,424275 | 0,850788 | 0,340275 |
| 8. | 800 m | 0,467536 | 1,170082 | 0,303888 |

Hasil pengujian pada Tabel 6.8 menunjukkan waktu yang diperlukan untuk mengirim 39 data dari *node* sensor ke LoRa *gateway* melalui LoRa. Nilai yang ditampilkan pada tabel tersebut adalah nilai rata-rata, nilai tertinggi, dan nilai terendah dari *delay* untuk masing-masing jarak. Nilai ditampilkan dengan format enam angka di belakang koma.

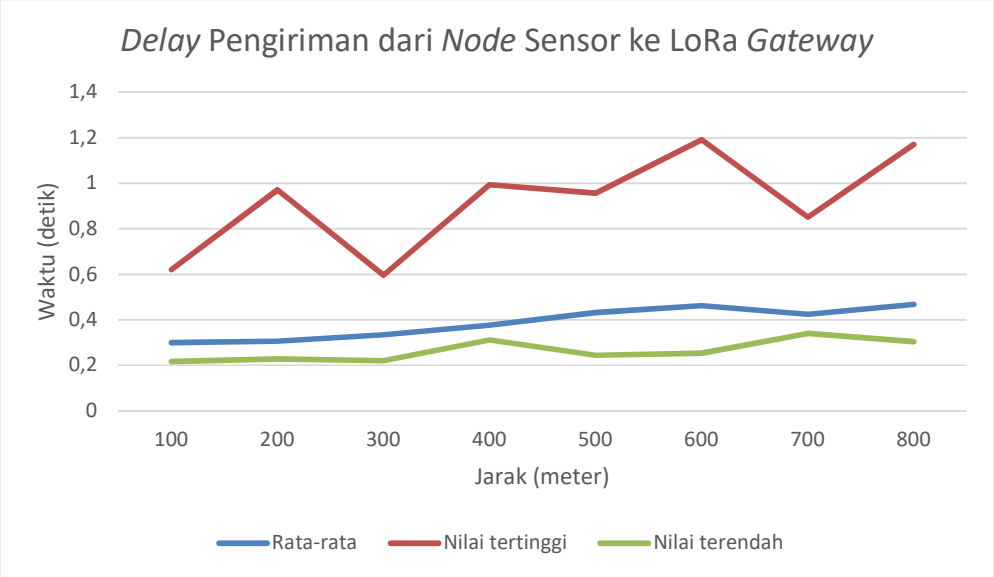
Pada pengujian di jarak 100 meter, nilai rata-rata *delay* pengiriman data melalui LoRa yang didapat adalah 0,299736. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 0,620823 dan 0,216548. Untuk jarak 200 meter, nilai rata-rata *delay* yang didapat adalah 0,306194. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 0,971474 dan 0,228347.

Pada pengujian di jarak 300 meter, nilai rata-rata *delay* pengiriman data melalui LoRa yang didapat adalah 0,33434. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 0,596061 dan 0,220704. Untuk jarak 400 meter, nilai rata-rata *delay* yang didapat adalah 0,376512. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 0,993208 dan 0,310993.

Pada pengujian di jarak 500 meter, nilai rata-rata *delay* pengiriman data melalui LoRa yang didapat adalah 0,432373. Sedangkan nilai tertinggi dan nilai

terendahnya masing-masing adalah 0,95675 dan 0,24418. Untuk jarak 600 meter, nilai rata-rata *delay* yang didapat adalah 0,462563. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 1,191328 dan 0,253237.

Pada pengujian di jarak 700 meter, nilai rata-rata *delay* pengiriman data melalui LoRa yang didapat adalah 0,424274. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 0,850788 dan 0,340274. Untuk jarak 800 meter, nilai rata-rata *delay* yang didapat adalah 0,467536. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 1,170081 dan 0,303888.




**Gambar 6.24 Grafik *delay* pengiriman data dari *node* sensor ke LoRa *gateway***

Dari hasil pengujian yang telah dijabarkan pada Tabel 6.8 dan diilustrasikan pada Gambar 6.24, dapat disimpulkan bahwa pengiriman data dari *node* sensor ke LoRa *gateway* menggunakan LoRa sampai dengan jarak 800 meter masih menghasilkan *delay* rata-rata di bawah 0,5 detik. Namun, terkadang dapat terjadi lonjakan *delay* tergantung dari kondisi lingkungan operasionalnya seperti yang ditunjukkan pada nilai maksimum *delay* pengiriman pada jarak 600 meter dan 800 meter.

### 6.2.2 Pengujian *End-to-End Delay* Pengiriman Data ECG dari *Node* Sensor Sampai ke Aplikasi pada *Cloud*

Tabel 6.9 berikut ini menjelaskan skenario kasus uji untuk pengujian *end-to-end delay* pengiriman data ECG dari node sensor sampai ke aplikasi pada *cloud*.

**Tabel 6.9 Pengujian *end-to-end delay* pengiriman data ECG dari *node* sensor sampai ke aplikasi *cloud***

| **Kode** | **PK_02** |
|---|---|
| Aspek | Kinerja |

| Kode | PK_02 |
|---|---|
| Nama Kasus Uji | *End-to-end delay* pengiriman data ECG dari *node* sensor sampai ke aplikasi pada *cloud* |
| Tujuan Pengujian | Mengetahui *end-to-end delay* pengiriman data ECG dari *node* sensor sampai ke aplikasi pada *cloud* |
| Prosedur Pengujian | Pengujian dilakukan dengan cara mengirimkan satu paket data ECG sekunder (39 data) ke LoRa *gateway* menggunakan LoRa dan LoRa *gateway* meneruskannya ke aplikasi pada *cloud*. Pada saat itu, LoRa *gateway* akan merekam *log* waktu mulainya pengiriman serta waktu sampainya data pada aplikasi menggunakan *Network Time Protocol* dan disimpan dalam format *file* .csv. Pengiriman paket dilakukan sebanyak 60 kali dengan interval 5 detik. Pengujian dilakukan 8 kali dengan jarak antara *node* sensor dan LoRa *gateway* sebagai pembeda. Jarak-jarak tersebut adalah 100 meter, 200 meter, 300 meter, 400 meter, 500 meter, 600 meter, 700 meter, dan 800 meter. Untuk memperoleh nilai *end-to-end delay*-nya, dilakukan operasi pengurangan antara waktu diterimanya data pada aplikasi dan waktu pengiriman data dari *node* sensor |

Hasil pengujian *end-to-end delay* untuk pengiriman data dari *node* sensor sampai ke aplikasi pada *cloud* dapat dilihat pada Tabel 6.10.

**Tabel 6.10 Hasil pengujian *end-to-end delay***

| No. | Jarak | End-to-end delay (detik) | | |
|---|---|---|---|---|
| | | Rata-rata | Nilai tertinggi | Nilai terendah |
| 1. | 100 m | 0,652137 | 3,136983 | 0,216548 |
| 2. | 200 m | 0,585722 | 1,782869 | 0,228347 |
| 3. | 300 m | 0,56162 | 1,094786 | 0,220704 |
| 4. | 400 m | 0,696592 | 1,593183 | 0,310993 |
| 5. | 500 m | 1,009045 | 2,980854 | 0,24418 |
| 6. | 600 m | 0,70166 | 1,72405 | 0,253237 |
| 7. | 700 m | 0,855856 | 3,681862 | 0,340275 |
| 8. | 800 m | 0,724532 | 3,905741 | 0,303888 |

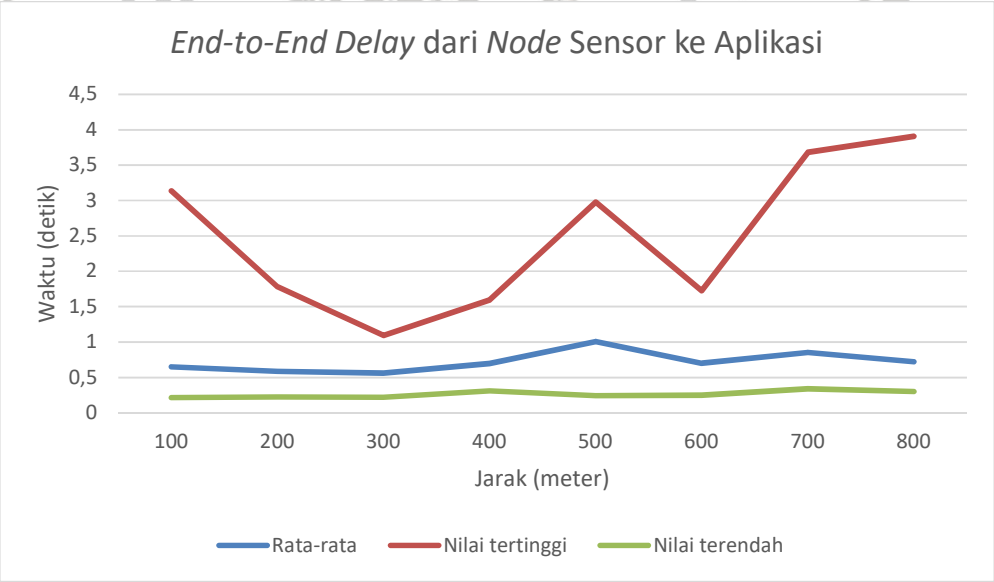
Hasil pengujian pada Tabel 6.10 menunjukkan waktu yang diperlukan untuk mengirim 39 data dari *node* sensor sampai ke aplikasi pada *cloud*. Nilai yang ditampilkan pada tabel tersebut adalah nilai rata-rata, nilai tertinggi, dan nilai terendah dari *end-to-end delay* untuk masing-masing jarak. Nilai ditampilkan dengan format enam angka di belakang koma.

Pada pengujian di jarak 100 meter, nilai rata-rata *end-to-end delay* yang didapat adalah 0,652137. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 3,136982 dan 0,216548. Untuk jarak 200 meter, nilai rata-rata *end-to-end delay* yang didapat adalah 0,585721. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 1,782868 dan 0,228347.

Pada pengujian di jarak 300 meter, nilai rata-rata *end-to-end delay* yang didapat adalah 0,561619. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 1,094786 dan 0,220704. Untuk jarak 400 meter, nilai rata-rata *end-to-end delay* yang didapat adalah 0,696592. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 1,593183 dan 0,310993.

Pada pengujian di jarak 500 meter, nilai rata-rata *end-to-end delay* yang didapat adalah 1,009045. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 2,980854 dan 0,244180. Untuk jarak 600 meter, nilai rata-rata *end-to-end delay* yang didapat adalah 0,701659. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 1,724049 dan 0,253237.

Pada pengujian di jarak 700 meter, nilai rata-rata *end-to-end delay* yang didapat adalah 0,855856. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 3,681862 dan 0,340275. Untuk jarak 800 meter, nilai rata-rata *end-to-end delay* yang didapat adalah 0,724532. Sedangkan nilai tertinggi dan nilai terendahnya masing-masing adalah 3,905741 dan 0,303888.




**Gambar 6.25 Grafik *end-to-end delay* dari *node* sensor ke aplikasi**

Dari hasil pengujian yang telah dijabarkan pada Tabel 6.10 dapat disimpulkan bahwa pengiriman 39 data *end-to-end* menggunakan LoRa sampai dengan jarak 800 meter masih menghasilkan *delay* rata-rata sampai 1,01 detik. Namun, terkadang dapat terjadi lonjakan *delay* tergantung dari kondisi lingkungan operasionalnya seperti yang ditunjukkan pada nilai maksimum *end-to-end delay* pada jarak 800 meter.

### 6.2.3 Pengujian *End-to-End Delay* Pengiriman Hasil Satu Siklus Perekaman Data ECG dari *Node* Sensor sampai ke Aplikasi

Tabel 6.11 menjelaskan skenario kasus uji untuk pengujian *end-to-end delay* pengiriman hasil satu siklus perekaman data ECG dari *node* sensor ke aplikasi.

**Tabel 6.11 Pengujian *end-to-end delay* pengiriman hasil satu siklus perekaman data ECG dari *node* sensor sampai ke aplikasi**

| Kode | PK_03 |
|---|---|
| Aspek | Kinerja |
| Nama Kasus Uji | *End-to-end delay* pengiriman hasil satu siklus perekaman data ECG dari *node* sensor sampai ke aplikasi pada *cloud.* |
| Tujuan Pengujian | Mengetahui *end-to-end delay* pengiriman hasil satu siklus perekaman data ECG dari *node* sensor sampai ke aplikasi. |
| Prosedur Pengujian | Pengujian dilakukan dengan cara mengirimkan data ECG sekunder yang berjumlah 1000 data ke LoRa *gateway* menggunakan LoRa dan LoRa *gateway* meneruskannya ke aplikasi pada *cloud*. Pada saat itu, *log* waktu mulainya pengiriman data pertama serta waktu sampainya data terakhir pada aplikasi akan direkam menggunakan *Network Time Protocol* dan disimpan dalam format *file* .csv. Pengujian dilakukan 60 kali dengan menggunakan data yang sama pada jarak 100 meter. Untuk memperoleh nilai *end-to-end delay*, dilakukan operasi pengurangan antara waktu penerimaan data terakhir pada aplikasi dan waktu pengiriman data pertama dari *node* sensor. |

Hasil pengujian *end-to-end delay* pengiriman satu siklus perekaman data ECG dari *node* sensor sampai ke aplikasi pada *cloud* dapat dilihat pada Tabel 6.12.

**Tabel 6.12 Hasil pengujian *end-to-end delay* pengiriman hasil satu siklus perekaman data ECG**

| Jumlah pengujian | *End-to-end delay* pengiriman satu siklus rekaman data ECG (detik) | | |
|---|---|---|---|
| | Rata-rata | Nilai tertinggi | Nilai terendah |
| 60 | 25,577779 | 26,68955 | 25,3399 |

Hasil pengujian pada Tabel 6.12 menunjukkan waktu yang diperlukan untuk mengirim satu siklus hasil perekaman data ECG dari *node* sensor sampai ke aplikasi pada *cloud*. Nilai yang ditampilkan pada tabel tersebut adalah jumlah dilakukannya pengujian, nilai rata-rata, nilai tertinggi, dan nilai terendah dari *delay* yang diperoleh. Nilai ditampilkan dengan format enam angka di belakang koma.

Dari hasil pengujian yang telah dijabarkan pada Tabel 6.12 dapat disimpulkan bahwa dari 60 kali dilakukannya pengujian, rata-rata *end-to-end delay* yang dihasilkan adalah 25,577779 detik dengan *end-to-end delay* tertinggi sebesar 26,68955 detik dan *end-to-end delay* terendah sebesar 25,3399. Hal yang mempengaruhi *end-to-end delay* tersebut adalah waktu yang diperlukan untuk pembagian data, proses pengiriman datanya, dan *end-to-end delay* per paketnya.

### 6.2.4 Pengujian *Packet Delivery Ratio* yang Dikirimkan oleh *Node* Sensor melalui LoRa

Tabel 6.13 berikut ini menjelaskan skenario kasus uji untuk pengujian *Packet Delivery Ratio* yang dikirimkan oleh *node* sensor melalui LoRa.

**Tabel 6.13 Pengujian *Packet Delivery Ratio* yang dikirimkan *node* sensor melalui LoRa**

| Kode | PK_04 |
|---|---|
| Aspek | Ketersediaan |
| Nama Kasus Uji | *Packet Delivery Ratio* yang dikirimkan oleh *node* sensor melalui LoRa |
| Tujuan Pengujian | Mengetahui *Packet Delivery Ratio* yang dikirimkan oleh *node* sensor melalui LoRa |
| Prosedur Pengujian | Pengujian dilakukan dengan cara mengirimkan satu paket data ECG sekunder (39 data) ke LoRa *gateway* menggunakan LoRa dan menghitung jumlah paket yang diterima pada LoRa *gateway*. Perhitungan jumlah paket yang masuk ke LoRa *gateway* dilakukan dengan menghitung jumlah *log* waktu paket yang direkam pada saat menguji *delay* transmisi dan *delay end-to-end*. Pengiriman data dilakukan sebanyak 60 kali dengan interval 5 detik. Pengujian dilakukan 8 kali dengan jarak antara *node* sensor dan LoRa *gateway* sebagai pembeda. Jarak-jarak tersebut adalah 100 meter, 200 meter, 300 meter, 400 meter, 500 meter, 600 meter, 700 meter, dan 800 meter. Untuk memperoleh nilai *PDR*-nya, dilakukan operasi pembagian jumlah paket yang masuk ke LoRa *gateway* dengan jumlah paket yang dikirim oleh *node* sensor. |

Hasil pengujian PDR untuk pengiriman data dari *node* sensor sampai ke LoRa *gateway* dapat dilihat pada Tabel 6.14.

**Tabel 6.14 Hasil pengujian *Packet Delivery Ratio***

| No. | Jarak | Jumlah paket dikirim | Jumlah paket masuk | *Packet Delivery Ratio* (PDR) |
|---|---|---|---|---|
| 1. | 100 m | 60 | 60 | 100% |
| 2. | 200 m | 60 | 59 | 98,22% |
| 3. | 300 m | 60 | 60 | 100% |
| 4. | 400 m | 60 | 60 | 100% |
| 5. | 500 m | 60 | 60 | 100% |
| 6. | 600 m | 60 | 58 | 96,67% |
| 7. | 700 m | 60 | 60 | 100% |
| 8. | 800 m | 60 | 60 | 100% |

Hasil pengujian pada Tabel 6.12 menunjukkan jumlah paket yang dikirimkan dari *node* sensor, jumlah paket yang masuk ke LoRa *gateway*, dan *Packet Delivery Ratio*-nya (PDR). Tabel tersebut menampilkan semua parameter tersebut berdasarkan jarak ujinya. Nilai ditampilkan dengan format dua angka di belakang koma.

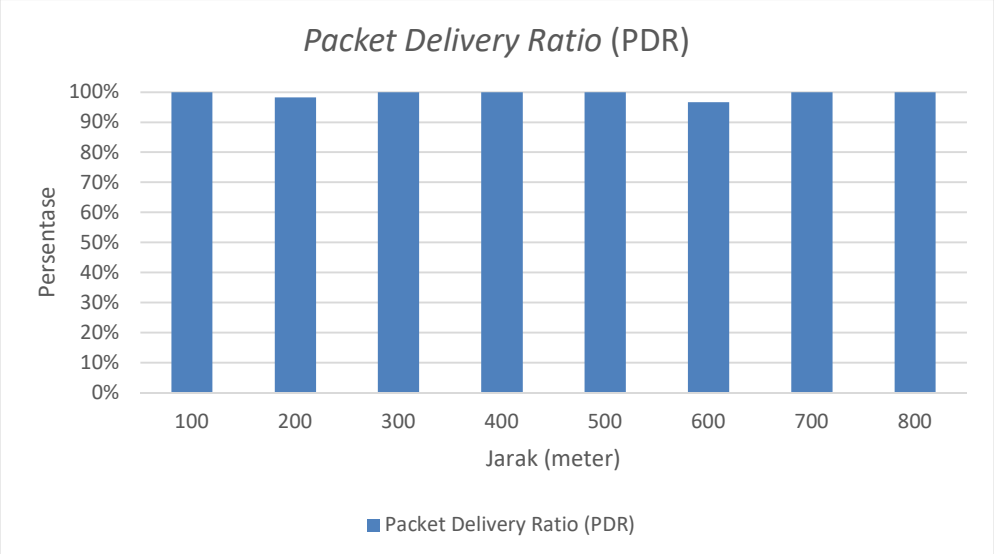
Pada pengujian di jarak 100 meter, jumlah paket yang dikirim oleh *node* sensor adalah 60 paket dan jumlah paket yang berhasil masuk ke LoRa *gateway* adalah

60 paket. Sehingga setelah dilakukan pembagian antara keduanya, didapatkan nilai PDR yaitu 100%. Untuk pengujian di jarak 200 meter, jumlah paket yang dikirim *node* sensor adalah 60 paket dan jumlah paket yang berhasil masuk ke LoRa *gateway* adalah 59 paket. Sehingga setelah dilakukan pembagian antara keduanya, didapatkan nilai PDR yaitu 98,22%.

Pada pengujian di jarak 300 meter, jumlah paket yang dikirim oleh *node* sensor adalah 60 paket dan jumlah paket yang berhasil masuk ke LoRa *gateway* adalah 60 paket. Sehingga setelah dilakukan pembagian antara keduanya, didapatkan nilai PDR yaitu 100%. Untuk pengujian di jarak 400 meter, jumlah paket yang dikirim *node* sensor adalah 60 paket dan jumlah paket yang berhasil masuk ke LoRa *gateway* adalah 60 paket. Sehingga setelah dilakukan pembagian antara keduanya, didapatkan nilai PDR yaitu 100%.

Pada pengujian di jarak 500 meter, jumlah paket yang dikirim oleh *node* sensor adalah 60 paket dan jumlah paket yang berhasil masuk ke LoRa *gateway* adalah 60 paket. Sehingga setelah dilakukan pembagian antara keduanya, didapatkan nilai PDR yaitu 100%. Untuk pengujian di jarak 600 meter, jumlah paket yang dikirim *node* sensor adalah 60 paket dan jumlah paket yang berhasil masuk ke LoRa *gateway* adalah 58 paket. Sehingga setelah dilakukan pembagian antara keduanya, didapatkan nilai PDR yaitu 96,67%.

Pada pengujian di jarak 700 meter, jumlah paket yang dikirim oleh *node* sensor adalah 60 paket dan jumlah paket yang berhasil masuk ke LoRa *gateway* adalah 60 paket. Sehingga setelah dilakukan pembagian antara keduanya, didapatkan nilai PDR yaitu 100%. Untuk pengujian di jarak 800 meter, jumlah paket yang dikirim *node* sensor adalah 60 paket dan jumlah paket yang berhasil masuk ke LoRa *gateway* adalah 60 paket. Sehingga setelah dilakukan pembagian antara keduanya, didapatkan nilai PDR yaitu 100%.




**Gambar 6.26 Grafik *Packet Delivery Ratio* (PDR)**

Dari hasil pengujian yang telah dijabarkan pada Tabel 6.14 dan digambarkan pada Gambar 6.26, dapat disimpulkan bahwa pengiriman data menggunakan LoRa sampai dengan jarak 800 meter masih menghasilkan PDR di atas 95%. Sedangkan, PDR yang paling kecil didapat dari pengujian pada jarak 600 meter. Hal ini dapat terjadi karena pengujian dilakukan pada kondisi *line-of-sight* sehingga memungkinkan rambatan sinyal LoRa di udara tidak terhambat oleh objek-objek tertentu yang berpotensi mengganggu rambatan sinyal.

# BAB 7 KESIMPULAN DAN SARAN

Bagian ini memuat kesimpulan dan saran terhadap skripsi. Kesimpulan diperoleh dari hasil perancangan, implementasi, dan pengujian. Sedangkan saran memuat masukan bagi peneliti-peneliti yang tertarik untuk mengembangkan penelitian ini lebih lanjut atau menganalisis penelitian ini.

## 7.1 Kesimpulan

Berdasarkan perancangan, implementasi, dan pengujian yang telah dilakukan, dapat ditarik kesimpulan sebagai berikut:

1. Data ECG yang ditransmisikan oleh *node* sensor memuat informasi topik, id sensor, dan data hasil rekam ECG pengguna. Ketiga informasi tersebut disimpan ke dalam bentuk *list* dan diformat dengan menggunakan format JSON. Dikarenakan ukuran maksimum untuk *payload* LoRa adalah 255 *byte*, maka data hasil rekam ECG yang dimasukkan ke dalam *payload* hanya sejumlah 39 data yang berjumlah 234 *bytes* (251 *bytes* apabila sudah diformat dengan topik dan id sensor) dalam sekali transmisi.
2. Mekanisme transmisi data ECG dari *node* sensor ke aplikasi pada *cloud* melalui LoRa *gateway* telah berhasil diimplementasikan. Pengiriman data ECG dari *node* sensor ke LoRa *gateway* dilakukan dengan menggunakan media transmisi LoRa dan dengan format data seperti yang telah dijelaskan pada poin 1. Sedangkan penerusan data ke aplikasi dilakukan dengan cara mengirimkan *request* HTTP *post* yang memuat id sensor dan data ECG dari *node* sensor beserta *token* yang sudah terautentikasi ke aplikasi dengan topik yang sesuai menggunakan media transmisi WiFi.
3. Kinerja untuk transmisi data ECG dari *node* sensor ke LoRa *gateway* via media transmisi LoRa lalu ke *cloud* diukur menggunakan empat parameter yaitu *delay* pengiriman, *end-to-end delay*, *end-to-end delay* pengiriman hasil satu siklus perekaman, dan *Packet Delivery Ratio* (PDR). Hasil dari pengujian yang menggambarkan kinerjanya adalah sebagai berikut:
   - Dari hasil pengujian *delay* pengiriman data dari *node* sensor ke LoRa *gateway* melalui LoRa, didapatkan kesimpulan bahwa rata-rata *delay* yang terjadi masih di bawah 0,5 detik. Namun tetap tidak menutup kemungkinan bahwa dapat terjadi *delay* di bawah ataupun di atas angka tersebut. Seperti yang terlihat saat pengujian pada jarak 600 meter, terjadi satu kali lonjakan *delay* yang mencapai 1,19 detik.
   - Dari hasil pengujian *end-to-end delay* pengiriman data dari *node* sensor sampai ke aplikasi pada *cloud*, didapatkan kesimpulan bahwa rata-rata *delay* yang terjadi berada di rentang antara 0,56 sampai 1,01. Namun, dalam kasus ini, hal yang memiliki pengaruh besar terhadap nilai *end-to-end delay* adalah *delay* pengiriman data dari LoRa *gateway* ke aplikasi pada *cloud*. Karena saat pengujian

dilakukan, *delay* yang terjadi saat pengiriman data dari LoRa *gateway* ke aplikasi pada *cloud* cenderung tidak stabil.

- Dari hasil pengujian *end-to-end delay* pengiriman hasil satu siklus perekaman data ECG dari *node* sensor sampai ke aplikasi pada *cloud*, didapatkan kesimpulan bahwa rata-rata *delay* yang terjadi masih berada di kisaran 25,5 detik. Namun tetap tidak menutup kemungkinan bahwa dapat terjadi *delay* di bawah ataupun di atas angka tersebut seperti yang terlihat pada nilai tertinggi dan nilai terendah yaitu 26,7 detik dan 25,3 detik.
- Dari hasil pengujian PDR untuk pengiriman data dari *node* sensor ke LoRa *gateway* melalui LoRa, didapatkan kesimpulan bahwa untuk transmisi sampai dengan jarak 800 meter, LoRa masih dapat memberikan nilai PDR di atas 95% dengan nilai PDR yang terkecil terjadi pada saat pengujian pada jarak 600 meter. Namun, yang mempengaruhi nilai PDR bukan hanya jarak, melainkan faktor lingkungan juga berpengaruh. Terlihat pada hasil pengujian PDR, pada jarak 800 meter yang termasuk jarak terjauh pada pengujian ini namun mendapatkan hasil nilai PDR yang sempurna yaitu 100%.

Dari hasil pengujian yang diperoleh, dapat disimpulkan bahwa mekanisme pengiriman data ECG yang diterapkan pada penelitian ini masih belum dapat memenuhi salah satu kebutuhan yang harus dipenuhi untuk keperluan pemantauan ECG, yaitu nilai 4 detik sebagai *delay* maksimum yang masih dapat ditoleransi pada pengiriman datanya. Sedangkan, nilai *end-to-end delay* yang diperoleh pada pengiriman satu siklus data ECG pada penelitian ini masih berada di atas angka tersebut, yaitu rata-rata sebesar 25,5 detik. Sehingga, mekanisme pengiriman data ECG yang diterapkan pada penelitian ini masih perlu pengembangan lebih lanjut agar dapat memenuhi kebutuhan tersebut.

## 7.2 Saran

Berikut adalah saran yang dapat dijadikan acuan untuk pelaksanaan penelitian lanjutan atau penelitian lain untuk mengembangkan penelitian ini.

1. Dapat dilakukan pengembangan lebih lanjut pada mekanisme pengiriman data ECG yang diterapkan pada penelitian ini. Hal ini dapat dilakukan karena mekanisme yang dihasilkan pada penelitian ini masih belum dapat memenuhi salah satu kebutuhan untuk diterapkan pada pemantauan ECG pengguna yaitu *delay* yang dihasilkan masih berada di atas *delay* maksimum yang dapat ditoleransi.
2. Dapat dikembangkan algoritma keamanan data saat transmisi data dilakukan. Hal ini dapat dilakukan karena pada penelitian ini data yang dikirimkan tidak memiliki mekanisme pengamanan sama sekali.

3. Dapat dilakukan analisis perbandingan kinerja penggunaan media transmisi LoRa pada bidang *monitoring* kesehatan dengan media transmisi lain seperti Zigbee, BLE, 6LowPAN, atau media transmisi lainnya yang dinilai tepat digunakan pada IoT.

# DAFTAR REFERENSI


Alesanco, Á. dan García, J., 2010. Clinical assessment of wireless ECG transmission in real-time cardiac telemonitoring. *IEEE Transactions on Information Technology in Biomedicine*, 14(5), hal.1144–1152.

Ali, Z.H., Ali, H.A. dan Badawy, M.M., 2015. Internet of Things (IoT): Definitions, Challenges and Recent Research Directions. *International Journal of Computer Applications*, 128(1), hal.37–47.

American Heart Association, 2015. *Electrocardiogram (ECG or EKG) | American Heart Association*. *Heart.org*. Tersedia pada: <https://www.heart.org/en/health-topics/heart-attack/diagnosing-a-heart-attack/electrocardiogram-ecg-or-ekg>.

Ashley, E.A. dan Niebauer, J., 2004. Conquering the ECG. In: *Cardiology Explained*. London: Remedica.

Atzori, L., Iera, A. dan Morabito, G., 2010. The Internet of Things: A survey. *Computer Networks*, [daring] 54(15), hal.2787–2805. Tersedia pada: <http://dx.doi.org/10.1016/j.comnet.2010.05.010>.

Chen, H., Jia, X. dan Li, H., 2011. A Brief Introduction to IoT Gateway. *Proceeding of ICCTA2011*, 7, hal.5–8.

Curry, S.J., Krist, A.H., Owens, D.K., Barry, M.J., Caughey, A.B., Davidson, K.W., Doubeni, C.A., Epling, J.W., Kemper, A.R., Kubik, M., Seth Landefeld, C., Mangione, C.M., Silverstein, M., Simon, M.A., Tseng, C.W. dan Wong, J.B., 2018. Screening for cardiovascular disease risk with electrocardiography us preventive services task force recommendation statement. *JAMA - Journal of the American Medical Association*, 319(22), hal.2308–2314.

Devalal, S. dan Karthikeyan, A., 2018. LoRa technology-an overview. *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, (Iceca), hal.284–290.

Diehl, T.S., 2011. *ECG made incredibly easy*. 5th ed. [daring] Lippincott Williams Wilkins. Tersedia pada: <https://www.theheartcheck.com/documents/ECG Interpretation Made Incredibly Easy! (5th edition).pdf>.

Espressif Systems, 2019a. ESP32-WROOM-32.

Espressif Systems, 2019b. ESP32 Series Datasheet. *Espressif Systems*, [daring] hal.1–61. Tersedia pada: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>.

Hodgart, E. dan Macfarlane, P.W., 2004. 10 Second heart rate variability. *Computers in Cardiology*, 31, hal.217–220.

Islam, M.S., Islam, M.T. dan Almutairi, A.F., 2019. applied sciences Monitoring of the Human Body Signal through the Internet of Things ( IoT ) Based LoRa

Wireless Network System.

de Jong, Y.L.C., Camire, D. dan Rogers, D. V., 2011. 700 AND 2 , 500 MHz PERTAINING TO MACROCELLULAR COVERAGE. hal.1–32.

Karray, F., Jmal, M.W., Abid, M., Bensaleh, M.S. dan Obeid, A.M., 2014. A review on wireless sensor node architectures. *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip, ReCoSoC 2014*, (May).

Kwon, O., Jeong, J. dan Kim, H. Bin, 2018. ECG Sampling Frequency for HRV Analysis. *Healthcare Informatics Research*, [daring] 24(3), hal.198–206. Tersedia pada: <www.e-hir.org>.

Minerva, R., Biru, A. dan Rotondi, D., 2015. Towards a definition of the Internet of Things (IoT). *IEEE IoT Initiative white paper*, [daring] (1), hal.86. Tersedia pada: <https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf>.

Nogueira, V. dan Carnaz, G., 2019. An Overview of IoT and Healthcare. (February).

Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D., 2014. Context aware computing for the internet of things: A survey. Communications Surveys Tutorials. *Ieee*, 16(1), hal.414 – 454.

Pramukantoro, E.S., Luckies, M. dan Bakhtiar, F.A., 2019. Bridging IoT infrastructure and cloud application using cellular-based internet gateway device. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 17(3), hal.1439.

Pramukantoro, E.S., Yahya, W., Arganata, G., Bhawiyuga, A. dan Basuki, A., 2017. Topic Based IoT Data Storage Framework for Heterogeneous Sensor Data. In: *11th International Conference on Telecommunication Systems Services and Applications (TSSA)*. Lombok: IEEE.

Raspberry Pi Foundation, 2016. Raspberry Pi 3 Model B+ Datasheet. *Datasheet*, [daring] hal.5. Tersedia pada: <https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf>.

Salman, T. dan Jain, R., 2017. Advanced Computing and Communications , Vol . 1 , No . 1 , March 2017 . 1(1).

Sarkar, N.I. dan Member, S., 2011. The Impact of Transmission Overheads on IEEE 802 . 11 Throughput : Analysis and Simulation. (May), hal.49–55.

Saville, R., 2019. *What is a Raspberry Pi?* [daring] Raspberrypi.Org. Tersedia pada: <https://www.raspberrypi.org/help/what- is-a-raspberry-pi/%0Ahttp://www.raspberrypi.org> [Diakses 15 Des 2019].

Semtech, 2015. LoRa™Modulation Basics Semtech. AN1200.22(May), hal.1–26.

Semtech, 2018. *What is LoRa? | Semtech LoRa Technology | Semtech*. Tersedia pada: <https://www.semtech.com/lora/what-is-lora>.

Sonnenberg, J., 2019. *The LoRa Protocol*.

WEMOS Electronics, 2018. *D32 [WEMOS Electronics]*. [daring] Tersedia pada: <https://wiki.wemos.cc/products:d32:d32>.

World Heart Federation, 2016. The World's Most Common Cause Of Death. Cardiovascular Diseases CVDs Global Facts and Figures. hal.863.

Yang, Z., Zhou, Q., Lei, L., Zheng, K. dan Xiang, W., 2016. An IoT-cloud Based Wearable ECG Monitoring System for Smart Healthcare. *Journal of Medical Systems*, 40(12).

Yi, J., Clausen, T.H., Townsley, W.M. dan Systems, C., 2016. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. (September).