TUGAS AKHIR – KI141502

IMPLEMENTASI KOMPRESI ADAPTIVE MENGGUNAKAN METODE HEATSHRINK UNTUK PENGIRIMAN DATA PADA WIRELESS SENSOR NETWORK BERBASIS ZIGBEE

MUHAMAD HENDRI FEBRIANSYAH
NRP 05111440000036

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

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom.

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

TUGAS AKHIR – KI141502

IMPLEMENTASI KOMPRESI ADAPTIVE MENGGUNAKAN METODE HEATSHRINK UNTUK PENGIRIMAN DATA PADA WIRELESS SENSOR NETWORK BERBASIS ZIGBEE

MUHAMAD HENDRI FEBRIANSYAH
NRP 05111440000036

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

Dosen Pembimbing II
Ir. Muchammad Husni, M.Kom.

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

IMPLEMENTATION OF ADAPTIVE COMPRESSION USING HEATSHRINK METHOD FOR DATA SHIPPING ON ZIGBEE BASED WIRELESS SENSOR NETWORK

MUHAMAD HENDRI FEBRIANSYAH
NRP 05111440000036

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

Second Advisor
Ir. Muchammad Husni, M.Kom.

INFORMATICS DEPARTEMENT
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI KOMPRESI ADAPTIVE MENGGUNAKAN METODE HEATSHRINK UNTUK PENGIRIMAN DATA PADA WIRELESS SENSOR NETWORK BERBASIS ZIGBEE

TUGAS AKHIR

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

Oleh:
MUHAMAD HENDRI FEBRIANSYAH
NRP: 051114410000036

Disetujui oleh Pembimbing Tugas Akhir:

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

2. Ir. Muchammad Husni M.Kom. .....
   (NIP. 196002211984031001) (Pembimbing 2)

SURABAYA
JULI, 2018

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI KOMPRESI ADAPTIVE MENGGUNAKAN METODE HEATSHRINK UNTUK PENGIRIMAN DATA PADA WIRELESS SENSOR NETWORK BERBASIS ZIGBEE

Nama Mahasiswa	**: MUHAMAD HENDRI FEBRIANSYAH**
NRP	**: 051114410000036**
Departemen	**: Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**: Ir. Muchammad Husni M.Kom.**

ABSTRAK

Kompresi data adalah suatu teknik untuk memampatkan data sehingga memerlukan ruang penyimpanan yang lebih kecil atau dapat mempersingkat waktu pertukaran data tersebut. Pada wireless sensor network kompresi data merupakan salah satu metode efektif untuk memperpanjang life time dan dapat digunakan pada sumber daya yang terbatas. Oleh karena itu, diperlukan adanya penerapan metode kompresi adaptive dalam pengiriman data pada wireless sensor network.

Pada Tugas Akhir ini, ZigBee berfungsi sebagai media pengiriman data dari node End device ke node Router kemudian ke node Coordinator. Pada node End device dilakukan kompresi menggunakan algoritma Heatshrink, node Router meneruskan data yang dikirimkan dari node End device ke node Coordinator dan node Coordinator akan melakukan dekompresi berdasarkan konfigurasi yang telah ditentukan.

Pada sistem dilakukan uji coba fungsionalitas dan uji coba performa dengan menggunakan beberapa skenario yang telah ditentukan. Untuk uji coba fungsionalitas, sebagian besar sistem berjalan dengan sebagaimana mestinya. Pada uji coba performa, efektifitas kompresi paling tinggi adalah 60.5% untuk data dengan

panjang 980 karakter menggunakan konfigurasi HS (9,8). Waktu yang dibutuhkan untuk proses kompresi data paling cepat adalah 0.02948 detik untuk data dengan panjang 584 karakter menggunakan konfigurasi HS (8,7). Sedangkan waktu yang dibutuhkan untuk proses dekompresi data paling cepat adalah 0.011856 detik untuk data dengan panjang 584 karakter menggunakan konfigurasi HS (9,8). Akurasi pengiriman data ZigBee pada jaringan single hop berturut – turut adalah 100% (±10 meter), 86,8% (±20 meter), dan 69,5% (±30 meter). Sedangkan akurasi pengiriman data ZigBee pada jaringan multi hop berturut – turut adalah 100% (±10 meter), 90,6% (±20 meter), dan 82,6% (±30 meter).

Kata kunci: Kompresi data, ZigBee, Heatshrink, Wireless Sensor Network.

IMPLEMENTATION OF ADAPTIVE COMPRESSION USING HEATSHRINK METHOD FOR DATA SHIPPING ON ZIGBEE WIRELESS SENSOR NETWORK BASED

Student's Name	**: MUHAMAD HENDRI FEBRIANSYAH**
Student's ID	**: 051114410000036**
Department	**: Informatics FTIK-ITS**
First Advisor	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Second Advisor	**: Ir. Muchammad Husni M.Kom.**

ABSTRACT

Data compression is a technique for compressing data so that it requires less storage space or can shorten the time of data exchange. In wireless sensor network data compression is one effective method to extend life time and can be used on limited resources. Therefore, it is necessary to apply adaptive compression method in data transmission on wireless sensor network.

In this final project, ZigBee serves as a data delivery medium from the End device node to the Router node and then to the Coordinator node. In the End device node compressed using the Heatshrink algorithm, the Router node forwards the data sent from the End device node to the Coordinator node and the Coordinator node will decompress according to the specified configuration.

The functionality and performance of the system was tested using some predetermined skenarios. For a test of functionality, most sistems work properly. In a performance test, the highest compression effectiveness was 60.5% for data with a length of 980 characters using the HS configuration (9,8). The time required for the fastest data compression process is 0.02948 seconds for data with a length of 584 characters using the HS configuration (8,7). While time required for the fastest data decompression process is 0.011856 seconds for data with a length of 584 characters using

HS configuration (9,8). The accuracy of ZigBee data transmission on a single hop network is 100% (± 10 meters), 86.8% (± 20 meters), and 69.5% (± 30 meters), respectively. While accuracy of ZigBee data transmission on multi-hop network are 100% (± 10 meter), 90,6% (± 20 meter), and 82,6% (± 30 meter) respectively.

Keywords : Data compression, ZigBee, Heatshrink, Wireless Sensor Network.

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah Swt yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

"**IMPLEMENTASI KOMPRESI ADAPTIVE MENGGUNAKAN METODE HEATSHRINK UNTUK PENGIRIMAN DATA PADA WIRELESS SENSOR NETWORK BERBASIS ZIGBEE**"

yang merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Allah –'Azz Wa Jalla– dan Nabi Muhammad –Shallallahu 'alaihi wa sallam–.
2. Bapak Jamaludin dan Ibu Suryati selaku orang tua penulis serta keluarga penulis yang selalu memberikan dukungan doa, moral, dan material yang tak terhingga kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku pembimbing I sekaligus dosen wali penulis yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Bapak Ir. Muchammad Husni, M.Kom. selaku pembimbing II yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala jurusan Teknik Informatika ITS.
6. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir.

7. Bapak Ir. F.X. Arunanto, M.Sc. sebagai dosen penguji I Tugas Akhir penulis.
8. Bapak Bagus Jati Santoso, S.Kom., Ph.D. sebagai dosen penguji II Tugas Akhir penulis.
9. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
10. Ibu Eva Mursidah dan Ibu Sri Budiati yang selalu mempermudah penulis dalam peminjaman buku di RBTC.
11. Para alumni Teknik Informatika ITS, terkhususnya mas Ade Ilham Fajri S.Kom yang terus memberikan bimbingan jarak jauh, mas Hanif Sudira S.Kom, mas Ghulam S.Kom, Mas Regin Iqbal S.Kom dan mas Randy Bastian S.Kom.
12. Teman-teman Keluarga Muslim Informatika, yang sudah banyak meluruskan penulis.
13. Teman-teman seperjuangan RMK NCC/KBJ, yang telah menemani dan menyemangati penulis.
14. Teman-teman administrator NCC/KBJ, yang telah menemani dan menyemangati penulis selama penulis menjadi administrator, menjadi rumah kedua penulis selama penulis berkuliah.
15. Teman-teman angkatan 2014, yang sudah mendukung saya selama perkuliahan.
16. Sahabat penulis yang tidak dapat disebutkan satu per satu yang selalu membantu, menghibur, menjadi tempat bertukar ilmu dan berjuang bersama-sama penulis.

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

Surabaya, Juli 2018

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Wireless Sensor Network (WSN) merupakan salah satu teknologi yang paling menjanjikan untuk kebutuhan di masa depan. Hal ini dikarenakan harganya yang murah, mudah dibangun, terdapat sensor cerdas, ukurannya kecil dan multi fungsionalitas sesuai kebutuhan. Berdasarkan penelitian dari IDTechEx, diprediksi bahwa pasar WSN akan mengalami pertumbuhan menjadi $ 1,8 miliar pada tahun 2024. Data ini mengacu pada WSN yang didefinisikan sebagai jaringan *mesh* nirkabel, yaitu *self-healing* dan *self-organising* [1].

Terdapat banyak permasalahan dan tantangan yang harus dihadapi dalam WSN untuk meningkatkan efisiensi, kelayakan dan manfaat. Tantang tersebut dapat dikategorikan kedalam empat kategori, yaitu efisiensi daya, pengumpulan data, jaringan dan strategi penyebaran [2]. WSN pada dasarnya adalah sistem yang berbasis *event*, *node* sensor akan mendeteksi keadaan di lingkungan sekitarnya untuk dikirim ke *sink*. Namun, karena *node* sensor sering mendeteksi fenomena umum, maka kemungkinan ada beberapa redundansi dalam data yang sumbernya beragam berkomunikasi dengan sink tertentu. Untuk itu pemfilteran dan pemrosesan dalam jaringan diperlukan agar dapat menghemat penggunaan energi yang terbatas.

Agregasi data merupakan proses pengumpulan data dari berbagai *node* untuk menghilangkan redundansi, meminimalkan jumlah transmisi dan memberikan informasi yang ringkas ke simpul utama [3]. Tujuan agregasi data adalah untuk memperpanjang umur jaringan dengan mengurangi transmisi waktu atau ukuran data yang dipancarkan *node* menggunakan algoritma cerdas. Secara umum, agregasi data dapat dikategorikan menjadi dua subsistem yang berbeda, yaitu protokol jaringan dan penggabungan data. Gagasan protokol agregasi data pada jaringan

adalah bekerjasama antar *node* spasial dan temporal berkorelasi [4] dalam menyebarkan data yang dikumpulkan. Pendekatan ini telah banyak diajukan dalam penelitian seperti LEECH, TEEN, HEED dan PEGASIS. Disisi lain penggabungan data bertujuan untuk mengurangi ukuran data yang ditransmisikan oleh *sink* atau *node* itu sendiri. Ukuran data dapat dikurangi menggunakan teknik operasi matematika [5] (*median, average, moving average*), kompresi, estimasi data dan pemodelan.

Kompresi data merupakan salah satu metode efektif untuk mengurangi penggunaan daya yang terbatas pada WSN. Diasumsikan bahwa beberapa kehilangan presisi atau kedetailan data pada saat kompresi dapat ditolerir jika hal tersebut dapat mengurangi komunikasi. Namun, perlu diperhatikan juga bahwa kualitas kedetailan data harus dipenuhi pada saat kondisi tertentu agar informasi yang didapatkan semakin banyak dan detail.

Disisi lain, menggunakan pendekatan agregasi data akan mempengaruhi perilaku komunikasi pada jaringan. Transmisi data yang harusnya kontinu diubah ke transmisi *buffer* berdasarkan kemampuan pemrosesan data lokal. Dengan menggabungkan agregasi data dengan komunikasi *buffer* maka ukuran paket data akan meningkat. Dengan demikian, slot waktu yang dibutuhkan untuk mentransmisikan paket data juga ikut meningkat. Jika terjadi suatu kondisi dimana beberapa *node* ingin mentransmisikan datanya secara bersamaan, maka *node forwarding* masing-masing lingkungan multihop akan menjadi hambatan, terutama pada jaringan yang berdaya rendah dan *bandwidth* yang terbatas. Maka *buffer overflow*s dan rasio paket *loss* yang meningkat akan menjadi masalah besar [6].

Pada kodisi nyata, jaringan komunikasi pada *wireless sensor network* memiliki *buffer* yang sangat kecil. Di nrf24l01+ payload data yang disediakan hanya 32 *byte* dan di IEEE 802.15.4 payload data sebesar 133 *byte*s. Belum lagi jika kita menggunakan modul tambahan. Pada XBee beban *buffer* yang dialokasikan untuk 802.15.4 dan ZigBee masing-masing adalah 100 dan 72 *byte* [7].

Oleh karena itu, dibutuhkan suatu managemen pengunaan paket yang baik untuk menghindari terjadinya *buffer overflows*. Dalam Tugas Akhir ini metode yang diusulkan adalah implementasi kompresi adaptive menggunakan metode Heatshrink untuk pengiriman data pada *wireless sensor network* berbasis Zigbee. Algoritma kompresi data yang digunakan adalah Heatshrink. Algoritma ini berbasis pada algoritma Lempel-Ziv-Storer-Szymanski (LZSS) yang merupakan lossless kompresi data yang cocok untuk kompresi data pada *embedded sistem*.

1.2 Rumusan Masalah

Tugas Akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana cara menentukan level kompresi data?
2. Bagaimana metode yang digunakan dalam pengiriman data pada protokol ZigBee?
3. Bagaimana tingkat efisiensi dari sistem yang dibangun dapat diukur?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki batasan sebagai berikut:

1. Mengunakan mikrokontroler Arduino Mega.
2. Komunikasi nirkabel menggunakan protokol ZigBee
3. Menggunakan algoritma Heatshrink untuk melakukan kompresi dan dekompresi data.
4. Algoritma Heatshrink dibuat dalam bahasa pemrograman C.
5. Data uji coba yang digunakan berupa data teks

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Membangun sebuah sistem kompresi data pada platform *wireless sensor network*.

2. Melakukan implementasi algoritma Heatshrink untuk kompresi *adaptive* pada pada platform *wireless sensor network*.

1.5 Manfaat

Dengan dibuatnya Tugas Akhir ini diharapkan dapat memberikan manfaat dalam penggunaan algoritma Heatshrink untuk kompresi adaptive pada pengiriman data di *platform wireless sensor network*.

Sedangkan bagi penulis, Tugas Akhir ini bermanfaat sebagai sarana untuk mengimplementasikan ilmu yang telah dipelajari selama kuliah agar berguna bagi masyarakat.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal

Proposal Tugas Akhir ini berisi tentang deskripsi pendahuluan dari Tugas Akhir yang akan dibuat. Pendahuluan terdiri atas hal yang menjadi latar belakang diajukannya usulan Tugas Akhir, rumusan masalah yang diangkat, batasan masalah untuk Tugas Akhir, tujuan dari pembuatan Tugas Akhir, dan manfaat dari hasil pembuatan Tugas Akhir. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Sub bab metodologi berisi penjelasan tahapan mengenai tahapan penyusunan Tugas Akhir mulai dari penyusunan proposal hingga penyususunan buku Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Tahap studi literatur merupakan tahap pembelajaran dan pengumpulan informasi yang digunakan untuk mengimplementasikan Tugas Akhir. Tahap ini diawali dengan pengumpulan literatur, diskusi, eksplorasi teknologi, dan pustaka, serta pemahaman dasar teori yang digunakan pada topik Tugas Akhir. Literatur-literatur yang dimaksud disebutkan yaitu mengenai Arduino, bahasa pemrograman C, algoritma Heatshrink, dan protokol ZigBee.

1.6.3 Analisis dan Desain Perangkat Lunak

Pada tahap ini akan dilakukan analisa, perancangan, dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang akan dihadapi pada tahap implementasi. Kemudian akan dijabarkan kebutuhan-kebutuhan tersebut ke dalam perancangan fitur sistem. Berikut langkah yang akan dilakukan perancangan proses perangkat lunak:

1. Perancangan rangkaian *node* yang akan dibuat
2. Uji coba komunikasi menggunakan protokol ZigBee pada rangkaian *node*
3. Implementasi kompresi dan dekompresi pada *node*
4. Implementasi kompresi *adaptive* pada *node*

1.6.4 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal Tugas Akhir. Sistem dibangun menggunakan Arduino IDE (*Integrated Development Environment*), dengan bahasa pemrograman C dan algoritma Heatshrink.

1.6.5 Pengujian dan Evaluasi

Tahap pengujian dan evaluasi berisi pengujian aplikasi dan evaluasi berdasarkan hasil pengujian. Pada tahap ini dilakukan pengujian dari fungsionalitas dan performa sistem WSN yang mana nanatinya akan dijalankan skenario yang sudah ditentukan. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian yang telah dilakukan. Pengujian fungsionalitas meliputi uji coba setiap bagian perangkat keras yang dirangkai pada Arduino dan juga uji coba keseluruhan sistem. Pengujian performa meliputi tingkat akurasi hasil kompresi data dan efisiensi data yang dapat di hemat.

1.6.6 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir yang mencangkup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini memaparkan hasil studi literatur yang digunakan sebagai dasar untuk menyelesaikan Tugas Akhir ini, terdiri atas deskripsi mengenai *wireless sensor network*, mikrokontroler, protokol ZigBee, bahasa pemrograman C, dan algoritma Heatshrink.
3. Bab III. Perancangan Perangkat Lunak

Bab ini berisi pembahasan mengenai perancangan komunikasi pada sistem, perancangan kompresi data, pengiriman data, perancangan perangkat keras dan lunak..

4. Bab IV. Implementasi
   Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Implementasi yang akan diterapkan berupa *Pseudocode*.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini menjelaskan pengujian pada sistem yang dibuat. Pengujian akan dilakukan berupa uji coba fungsionalitas dan uji coba performa.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran
   Lampiran yang ada berisikan kelengkapan – kelengkapan yang diperlukan dalam menyusun buku Tugas Akhir.

(Halaman ini sengaja dikosongkan)

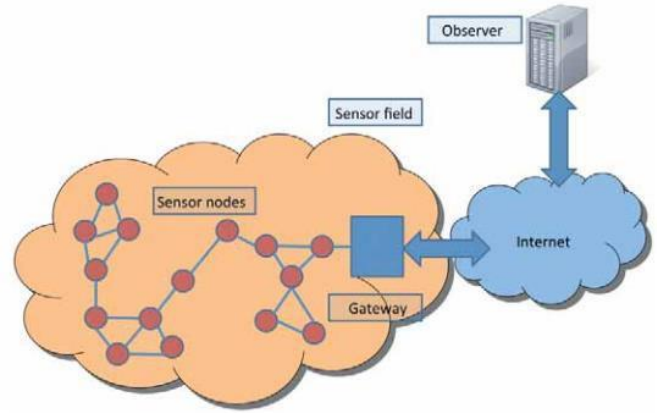
BAB II
TINJAUAN PUSTAKA

Bab ini membahas teori-teori dasar yang berkaitan dengan pokok bahasan Tugas Akhir. Bab ini juga menjelaskan modul dan alat yann nantinya akan digunakan pada tahap implementasi program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap alat yang digunakan dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 *Wireless Sensor Network*

Wireless Network Sensor (WSN) adalah jaringan yang dibentuk oleh sekumpulan banyak *node* sensor dimana masing-masing *node* dilengkapi dengan sensor untuk mendeteksi fenomena fisik seperti cahaya, panas, tekanan, getaran dan lain-lain. WSN dianggap sebagai metode pengumpulan informasi revolusioner untuk membangun sistem informasi dan komunikasi dalam meningkatkan kehandalan dan efisiensi sistem infrastruktur. Jika dibandingkan dengan solusi kabel, WSN lebih mudah dipasang dan memiliki fleksibilitas perangkat yang lebih baik [8].

WSN pada umumnya dapat digambarkan sebagai jaringan simpul yang secara kooperatif merasakan dan mengendalikan lingkungan, memungkinkan interaksi antar orang atau komputer dan lingkungan sekitar [9]. WSN saat ini biasanya mencakup *node* sensor, *node* akuator, *gateway* dan klien. Sejumlah besar *node* sensor diletakkan secara acak didalam atau di dekat area pemantauan (*sensor field*), kemudian membentuk jaringan melalui *self-organization*. *Node* sensor memonitor data yang terkumpul untuk dikirim bersama ke *node* sensor lainnya dengan melompat. Selama proses transmisi, data yang dipantau dapat ditanggani oleh beberapa *node* untuk sampai ke *node gateway* setelah *multihop* routing dan akhirnya mencapai *node management* melalui internet atau satelit. Gambar 2.1 merupakan contoh ilustrasi atau gambaran mengenai WSN.



Gambar 2.1 Ilustrasi Wireless Sensor Network [8]

2.2 Arduino UNO

Arduino adalah pengendali mikro single-board yang bersifat open-source, diturunkan dari Wiring platform, dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. Perangkat kerasnya memiliki prosesor Atmel AVR dan perangkat lunaknya memiliki bahasa pemrograman sendiri. Arduino juga merupakan platform *hardware* terbuka yang ditujukan kepada siapa saja yang ingin membuat purwarupa peralatan elektronik interaktif berdasarkan *hardware* dan *software* yang fleksibel dan mudah digunakan. Mikrokontroler diprogram menggunakan bahasa pemrograman arduino yang memiliki kemiripan *syntax* dengan bahasa pemrograman C. Karena sifatnya yang terbuka maka siapa saja dapat mengunduh skema *hardware* arduino dan membangunnya. [10]

Terdapat tiga jenis memori yang ada pada mikrokontroler papan Arduino berbasis AVR, yaitu :

a. *Flash memory*, berfungsi sebagai tempat untuk menyimpan sketsa Arduino.

b. SRAM (*Static Random Access Memory*) adalah tempat sketsa menciptakan dan memanipulasi variable yang sedang dijalankan atau digunakan.
c. EEPROM (*Electrically Erasable Programmable Read-Only Memory*) adalah memori yang dapat digunakan untuk programmer menyimpan informasi jangka panjang.

Pada Flash memory dan EEPROM informasi yang disimpan tetap ada walaupun *power* telah dimatikan. Sedangkan untuk SRAM, informasi yang disimpan akan hilang ketika power dimatikan.

Arduino UNO merupakan salah satu jenis mikrokontroler Arduino yang ada di pasaran. Arduino UNO menggunakan chipset ATmega 328, sebagaimana yang ditunjukkan pada Gambar 2.2

Gambar 2.2 Arduino UNO

Berikut ini adalah data sheet yang ada pada Arduino UNO Chip ATmega 238 (ditunjukkan pada Table 2.1).

Tabel 2.1 Arduino UNO Data Sheet

Mikrokontroller	ATmega238
Operating Voltage	5V

Input Voltage (recommended)	7 - 12V
Input Voltage (limits)	6 – 20 V
Digital I/O Pins	14 (6 pin merupakan pulse width modulation output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash memory	32 KB (0.5KB digunakan untuk bootloader)
SRAM	2 KB
EPPROM	1 KB
Clock Speed	16 MHz

2.3 Arduino Mega 2560

Arduino Mega 2560 merupakan salah satu jenis mikrokontroler Arduino yang menggunakan chip ATmega2560. Mikrokontroler ini memiliki pin I/O yang cukup banyak yaitu sejumlah 54 buah pin digital. Berikut ini adalah gambar Arduino Mega 2560 sebagaimana yang ditunjukkan pada Gambar 2.3 beserta data sheet yang ada pada Arduino Mega2560 Chip ATmega 2560 (ditunjukkan pada Table 2.2).

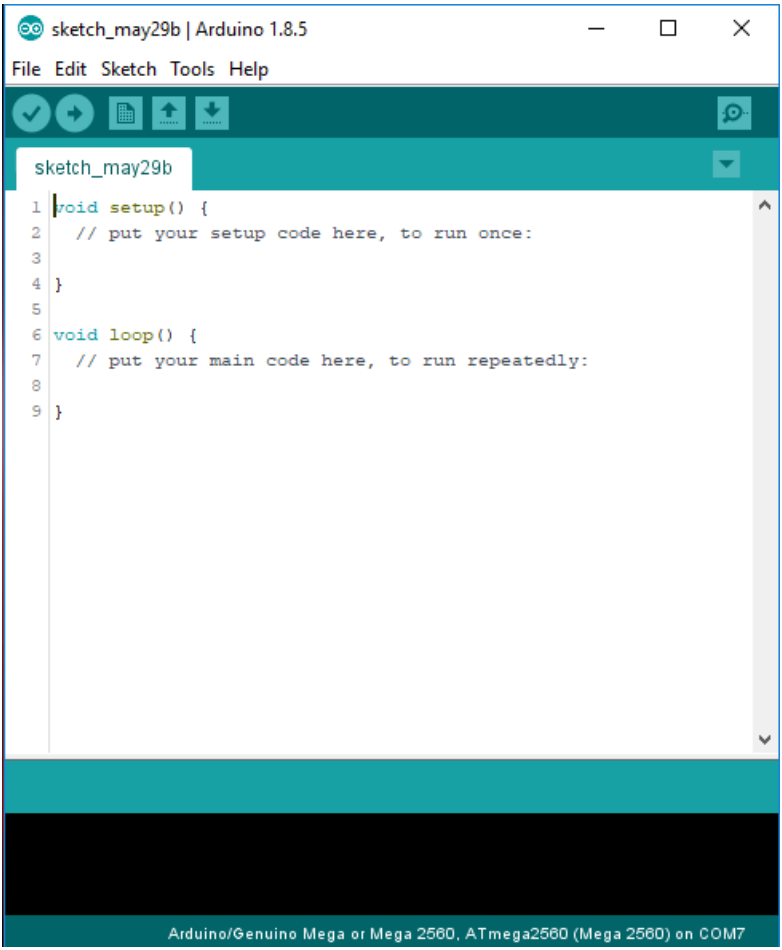
Gambar 2.3 Arduino Mega 2560

Tabel 2.2 Arduino Mega 2560 Data Sheet

Mikrokontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7 - 12V
Input Voltage (limits)	6 – 20 V
Digital I/O Pins	54 (14 pin merupakan pulse width modulation output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash memory	256 KB (8KB digunakan untuk bootloader)
SRAM	8 KB
EPPROM	4 KB
Clock Speed	16 MHz

2.4 *Arduino Integrated Development Environment*

Arduino Integrated Development Environment (IDE) merupakan editor teks untuk menulis kode, area pesan, konsol teks dan memiliki toolbar dengan tombol untuk fungsi umum dan serangkaian menu. Arduino IDE terhubung ke perangkat keras Arduino dan Genuino untuk mengunggah program dan dapat berkomunikasi dengan mereka. Program yang ditulis menggunakan Arduino Software (IDE) disebut sketsa. Sketsa ini ditulis dalam editor teks dan disimpan dengan ekstensi file .ino. Editor memiliki fitur untuk memotong / menempel dan mencari / mengganti teks. Area pesan memberi umpan balik saat menyimpan dan mengekspor dan menampilkan kesalahan. Konsol menampilkan *output* teks oleh Arduino Software (IDE), termasuk pesan kesalahan dan informasi lainnya yang lengkap [11]. Arduino IDE yang digunakan Arduino IDE versi 1.8.5 yang antar mukanya dapa dilihat pada Gambar 2.4 dibawah ini.



Gambar 2.4 Antarmuka Arduino IDE

2.5 XBee Shield

Shield adalah papan PCB atau lebih dikenal *board* yang dapat dihubungkan dengan papan Arduino untuk menambah fungsi dari arduino. XBee Shield ini dirancang dengan agar arduino dapat

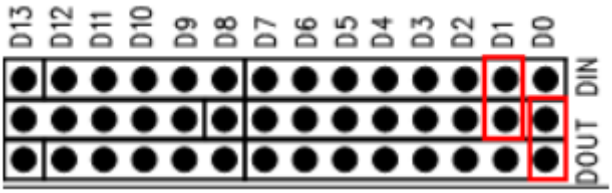
berkomunikasi secara nirkabel dengan modul XBee dari Maxstream. Dengan adanya modul ini, sebuah arduino akan mampu berkomunikasi secara nirkabel melebihi 30 meter di dalam ruangan dan 90 meter jika di luar ruangan. Dapat digunakan sebagai serial ataupun USB [12]. Pada Gambar 2.5 diperlihatkan bentuk fisik XBee Shield pabrikan Funduino dengan jumper bertipe *switch*.

Gambar 2.5 XBee Shield Pabrikan Funduino

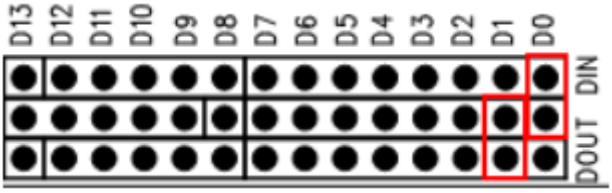
Pada penelitian ini, penulis menggunakan dua tipe XBee Shield. Pertama XBee Shield V03 keluaran dari perusahaan Funduino, dimana *jumper* yang digunakan bertipe switch, sehingga user lebih mudah untuk memilih mode *jumper*. Terdapat dua jenis jumper, yaitu mode XBee (berfungsi untuk pengiriman data) dan mode USB (berfungsi untuk programming). Perlu di perhatikan ketika memilih mode jumper, ketika ingin melakukan upload code ke arduino *jumper* harus berada dalam mode USB jika tidak maka code tidak akan bisa terupload. Begitu pula sebaliknya, ketika ingin mengirimkan data via *wireless jumper* harus berada dalam mode XBee.

Gambar 2.6 XBee Shield Pabrikan ITead Studio

Adapun XBee Shield jenis kedua merupakan XBee Shield V1.1 keluaran perusahaan ITead Studio. Terdapat dua jenis *jumper* yaitu mode USB dan mode XBee. Berikut posisi jumper shield tersebut



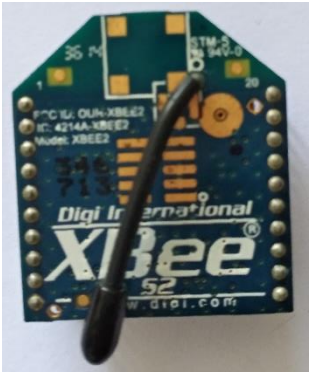
Gambar 2.7 Konfigurasi *Jumper* Mode USB



Gambar 2.8 Konfigurasi *Jumper* Mode XBee

2.6 XBee Modul S2

XBee modul seri 2 merupakan modul yang berfungsi untuk komunikasi antar jaringan nirkabel. XBee modul seri 2 menggunakan protokol ZigBee untuk saling berkomunikasi. Modul ini menyediakan transfer data yang sangat handal dengan kecepatan transfer mencapai 250 kbps. Setiap perangkat output serial dapat mengunakan modul untuk transfer data, *transmisi point to point* dan transmisi jaringan *multi-point*. Modul ini dirancang untuk aplikasi *high-throughput* (35kbps) yang membutuhkan latency rendah dan waktu komunikasi yang dapat diprediksi.



Gambar 2.9 XBee Modul S2

2.7 Protokol ZigBee

ZigBee merupakan standar komunikasi untuk perangkat nirkabel jarak pendek berdaya rendah yang berbasis pada standar IEEE 802.15.4 untuk jaringan area pribadi (PAN). Perangkat Zigbee mampu berkomunikasi *peer-to-peer*, *point-to-multipoint* dan *mesh*. Teknologi ini cocok untuk transfer data *rate* yang rendah, konsumsi daya yang rendah, biaya rendah, protocol jaringan nirkabel yang ditujukan untuk aplikasi otomasi dan *remote control*. Perangkat nirkabel yang sesuai dengan ZigBee

diperkirakan dapat melakukan transmisi 10 sampai 75 meter tergantung pada lingkungan RF dan konsumsi daya yang dikeluarkan untuk aplikasi tertentu. ZigBee memiliki tiga jenis tipe, yaitu 2.4GHz global (data rate 250kbps), 915MHz Americas (data rate 40kbps) dan 868 MHz Europe (data rate 20kbps). Jaringan ZigBee terdiri dari tiga jenis perangkat, yaitu: *coordinator*, *router*, dan *end device*. Setiap jaringan memiliki ID PAN 16bit. Semua perangkat dalam jaringan ZigBee diberi satu ID PAN.

Berikut ini adalah penjelasan lebih lanjut mengenai ketiga jenis perangkat ZigBee [13]:

a. ZigBee *Coordinator* (ZC) : ZC bertindak sebagai *coordinator* yang mengatur lalu lintas jaringan komunikasi. Harus ada satu ZC dalam setiap jaringan karena perangkat ini memulai jaringan dari awal. Koordinator memulai *Personal Area Network* (PAN) dengan memilih saluran RF dan PAN ID. ZC memungkinkan *router* dan *end-devices* untuk bergabung dengan PAN. Selain itu ZC, mampu menyimpan informasi tentang jaringan, termasuk bertindak sebagai *Trust Center* dan *repository* untuk kunci keamanan.
b. ZigBee *Router* (ZR) : ZR menjalanan fungsi aplikasi, selain itu *router* bertindak sebagai perantara, meneruskan data dari satu perangkat ke perangkat lain.
c. ZigBee *End device* (ZED) : ZED dapat melakukan komunikasi dengan koordinator dan *router*, akan tetapi tidak dapat menyampaikan data dari perangkat lain. Hubungan ini memungkinkan simpul untuk tidur dalam waktu yang cukup lama, sehingga dapat menghemat penggunaan baterai. ZED harus bergabung dengan PAN seperti *router* sebelum mengirimkan data sensor.

Kelebihan menggunakan ZigBee terutama terletak pada mode AT *default* nya, dimana lapisan PHY dan MAC frame transparan bagi pengguna. Artinya, pengguna biasa tidak akan melihat *frame acknowledgment* (ACK) atau transmisi ulang modul

frekuensi radio (RF) termasuk semua *byte* aktual yang dikirim. Pengguna hanya akan menyaksikan apakah data berhasil dikirim atau tidak, dengan semua teknis seperti *Carrier Sense Multiple Access - Collision Avoidance* (CSMA-CA) tersembunyi dari pandangan biasa dan karenanya menawarkan antarmuka yang lebih sederhana [7].

Beban maksimum yang dapat dialokasikan *buffer* XBee untuk 802.15.4 dan ZigBee masing-masing adalah 100 dan 72 *byte*. Manfaat *buffer* adalah pembacaan sensor ganda dapat dimasukkan ke dalam *frame* yang sama untuk satu sesi transmisi selama *buffer overflow* dihindari. [14] menunjukkan bahwa lonjakan arus untuk daya ZigBee menghabiskan 5 sampai 10 kali lebih besar daripada pada operasi normal. Oleh karena itu, *buffer* harus digunakan untuk mentransmisikan data sebanyak mungkin dalam interval yang dapat ditoleransi dengan aktivasi modul RF minimum.

2.8 Algoritma Heatshrink

Heatshrink merupakan algoritma kompresi lossless yang berbasis pada Lempel-Ziv-Storer-Szymanski (LZSS). Algoritma kompresi lossless memungkinkan untuk membentuk data asli yang tepat sama dari data yang sudah dikompresi. Algoritma ini cocok digunakan pada sistem *embedded* karena dapat berjalan dalam jumlah memori yang sangat kecil (dibawah 50 *byte* untuk dekompresi praktis). Selain itu, heatshrink dapat bekerja sedikit demi sedikit sambil menangani kebutuan lain dari sistem yang berjalan secara *real time* [15].

Heatshrink menggunakan algoritma Lempel-Ziv-Storer-Szymanski untuk melakukan kompresi dengan beberapa detail implementasi penting yang harus diperhatikan, yaitu [15] :

1. Proses kompresi dan dekompresi telah dirancang untuk berjalan secara bertahap, pemrosesan dapat bekerja beberapa *byte* setiap saat. Selain itu dapat juga menangguhkan dan melanjutkan proses sebagai data tambahan atau pada *buffer* yang tersedia.

2. Teknik *optional indexing* yang digunakan dapat mempercepat proses kompresi.
3. Secara umum *trade-off* implementasi banyak disukai pada penggunaan memori yang rendah.

Lempel-Ziv-Storer-Szymanski (LZSS) adalah salah satu jenis algoritma kompresi yang berbasis dictionary yang bersifat lossless (data dapat di rekonstruksi ulang menjadi data asli). LZSS merupakan salah satu varian dari LZ77 (Lempel Ziv 1977) yang dikembangkan oleh Storer dan Szymansky pada tahun 1982. Perbedaan yang mendasar antara kedua algoritma ini adalah jumlah token (tanda) yang terbentuk yakni dua token pada LZSS dan tiga token pada LZ77. Dua token yang dihasilkan oleh LZSS menunjukkan indeks dan panjang karakter yang sama pada dictionary. Sedangkan pada algoritma LZ77, dua token awal mempunyai fungsi sama dengan LZSS namun ada tambahan satu token yang berisi satu karakter yang mengikuti frasa yang sama tersebut [16].

Untuk proses kompresi dan dekompresi akan dijelaskan secara detail sebagai berikut [17].

1. Proses Kompresi

   Buffer dibagi menjadi dua, yakni *buffer* untuk pencarian dan *buffer look-ahead*. Setelah menginisialisasi *buffer*, karakter dibaca dari input data ke *buffer* data yang belum di kodekan. Untuk setiap karakter pada *buffer* yg belum di kodekan, dilakukan proses pencarian substring yang terpanjang di *buffer* pencarian sesuai dengan *buffer look-ahead* dimulai dengan karakter inputan pertama. Jika kecocokan substring sudah cukup, maka program akan mengkodekan indeks dan panjang substring ke dalam output. Jika tidak ada substring yang cocok dimulai dengan input pertama karakter masukan yang diberikan, maka karakter tersebut akan langsung ditulis ke output dengan *flag* yang menandakan tidak ada pengkodean yang dilakukan. Algoritma ini melakukan langkah-

langkah ini sampai tidak ada karakter yang tertinggal. Pengkodean dua karakter yang sesuai membutuhkan jumlah *byte* yang sama jika kita langsung menampilkan dua karakter.

Berikut ini adalah contoh ilustrasi dari proses kompresi data [18]. Pada Gambar 2.10 menampilkan teks asli yang masih belum di kompresi. Pada Gambar 2.11 akan menampilkan hasil kompresi yang telah dilakukan oleh algoritma LZSS.

```
  0: I am Sam
  9:
 10: Sam I am
 19:
 20: That Sam-I-am!
 35: That Sam-I-am!
 50: I do not like
 64: that Sam-I-am!
 79:
 80: Do you like green eggs and ham?
112:
113: I do not like them, Sam-I-am.
143: I do not like green eggs and ham.
```

Gambar 2.10 Contoh Teks Asli [18]

```
 0: I am Sam
 9:
10: (5,3) (0,4)
16:
17: That(4,4)-I-am!(19,16)I do not like
45: t(21,14)
49: Do you(58,5) green eggs and ham?
78: (49,14) them,(24,9).(112,15)(93,18).
```

Gambar 2.11 Teks yang Sudah Dikompresi [18]

Pada teks asli yang belum mengalami pengompresan, jumlah *byte* yang dihasilkan adalah 177 *byte* dari 177 karakter (termasuk spasi dan enter). Setelah dilakukan kompresi, jumlah *byte* berkurang menjadi 94 *byte*. Ini tidak termasuk 12 *byte* pada *flag* yang menunjukkan apakah potongan teks berikutnya adalah *pointer* atau *literal*. Jika ditambahkan dengan jumlah *flag* maka total ukurannya menjadi 106 *byte*, tentunya ini masih lebih pendek jika dibandingkan dengan ukuran aslinya 177 *byte*.

2. Proses Dekompresi

   Proses dekompresi dilakukan dengan menguraikan kode secara langsung dengan melibatkan membaca dan menulis ulang hasil tanpa melakukan pencarian apapun. *Flag* pengkodean dibaca untuk mengetahui karakter mana yang dikodekan. Jika flag menunjukkan bahwa karakter tersebut dikodekan, jumlah karakter dan posisi awal dikumpulkan dari bagian yang dikodekan. Kemudian jumlah karakter dengan indeks yang diberikan ditulis dari jendela geser ke *file output* atau memori. Jika tidak dikodekan, karakternya adalah *output* secara langsung. Dekompresi mengkonsumsi lebih sedikit sumber daya memori dan waktu komputasi jika dibandingkan dengan proses kompresi.

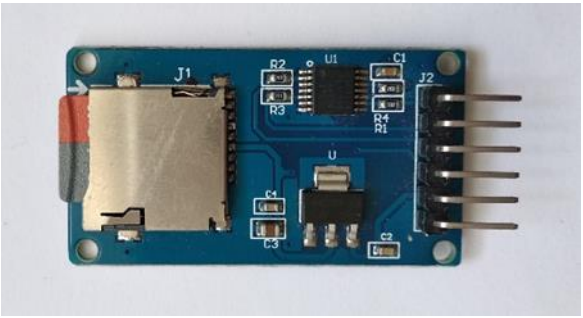
2.9 Bahasa Pemrograman C

Bahasa C merupakan salah satu bahasa pemrograman level tingkat menengah yang menjadi induk dari bahasa pemrograman modern seperti C++, C#, PHP, Javascript dan masih banyak lagi. Bahasa pemrograman C dibuat pertama kali oleh Dennis M Ritchie dengan tujuan untuk mengembangkan sistem operasi UNIX yang

sebelumnya menggunakan bahasa assembly. Adapun beberapa keunggulan bahasa C dibandingkan dengan bahasa pemrograman yang lain, yaitu: bahasa C termasuk bahasa pemrograman prosedural, bahasa C sangat cepat dan efisien, dan Bahasa C merupakan portable language.

2.10 MicroSD Card Adapter

MicroSD Card Adapter ini merupakan modul pembaca kartu MicroSD melalui sistem *file* dan SPI antarmuka *driver*, MCU untuk membaca dan menulis pada kartu microSD. Dengan menggunakan Arduino IDE dan libarary SD card, pengguna dapat menginisialisasi kartu SD card, membaca dan menulisnya. Gambar 2.12 merupakan bentuk fisik dari *microsd card adapter*.

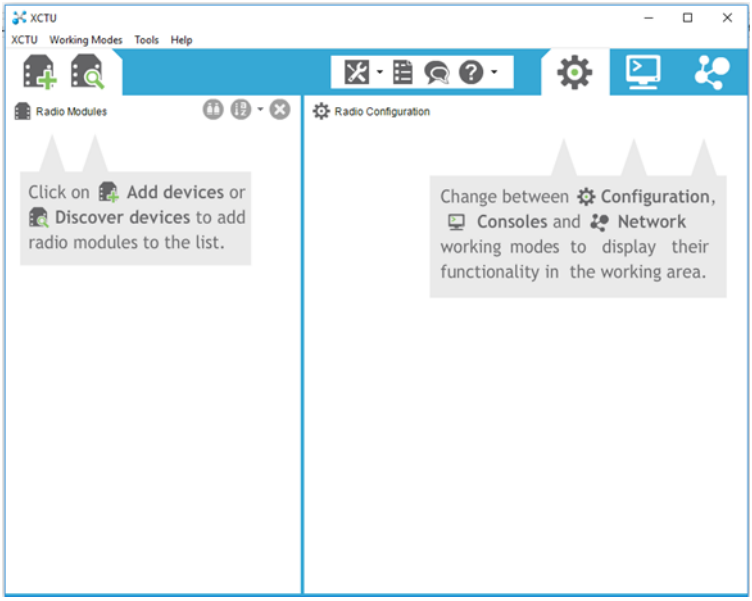


Gambar 2.12 MicroSD Card Adapter

2.11 DIGI XCTU

XCTU merupakan aplikasi *multi-platform* gratis yang dirancang untuk memungkinkan pengembangan modul Digi RF melalui antarmuka grafis yang mudah digunakan. Aplikasi ini dapat digunakan untuk melakukan konfigurasi dan pengujian pada modul XBee® RF. XCTU memiliki semua tools yang dibutuhkan pengembang untuk melakukan pengembangan dengan XBee. Terdapat fitur unik didalamnya, seperti tampilan jaringan grafis,

yang secara grafis mewakili jaringan XBee yang ada bersama dengan kekuatan sinyal setiap sambungan. Selain itu kita dapat menggunakan XBee API yang secara intuitif membantu dalam membangun dan menafsirkan API frame untuk XBee yang menggunakan mode API. Dengan menggunakan aplikasi ini, kita lebih mudah dalam melakukan pengembangan pada platform *wireless sensor network* yang menggunakan XBee. Gambar 2.13 merupakan antarmuka pada DIGI XCTU.



Gambar 2.13 Antarmuka XCTU

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai dasar dari perancanga sistem yang akan dibangun pada Tugas Akhir. Perancangan yang dibahas meliputi deskripsi umum sistem, proses perancangan, alur dan implementasinya.

3.1 Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibangun suatu sistem kompresi dan dekompresi data pada platform wireless sensor network dengan menggunakan algorima heatshrink. Teknologi *wireless sensor network* menggunakan mikrokontroler Arduino dan protokol ZigBee sebagai jalur komunikasi.

Gambar 3.1 Deskripsi Umum Sistem

Terdapat tiga *node* yang akan digunakan, yaitu *Coordinator*, *Router* dan *End device*. Setiap *node* memiliki komponen yang saling terhubung. Penjelasan komponen pendukung pada ketiga *node* tersebut sebagai berikut.

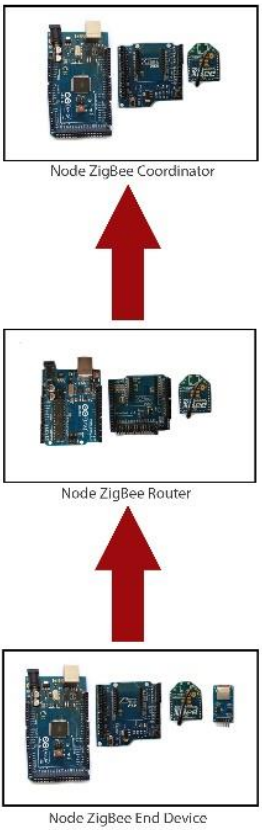
- *Node Coordinator*
  Node ZigBee *coordinator* berfungsi untuk menerima data dari *node Router*. Data tersebut kemudian akan dikompresi sesuai konfigurasi yang ditentukan. Berikut ini adalah komponen pendukung sistem pada *node Coordinator*.
  1. Mikrokontroler Arduino Mega sebagai computer yang mengendalikan sistem pada *node Coordinator*

2. XBee Shield
3. XBee Modul S2 sebagai perangkat radio komunikasi

- *Node Router*
  Node ZigBee *Router* berfungsi untuk menerima data hasil kompresi dari *node End device* kemudian meneruskannya ke *node Coordinator*. Berikut ini adalah komponen pendukung sistem pada *node Router*.
  1. Mikrokontroler Arduino UNO sebagai computer yang mengendalikan sistem pada *node Router*
  2. XBee Shield
  3. XBee Modul S2 sebagai perangkat radio komunikasi
- *Node End device*
  Node ZigBee *End device* bertugas melakukan kompresi data dan mengirimkan hasilnya beserta konfigurasi yang dikunakan ke *node Router*. Berikut ini adalah komponen pendukung sistem pada *node End device*.
  1. Mikrokontroler Arduino Mega sebagai computer yang mengendalikan sistem pada *node End device*
  2. XBee Shield
  3. XBee Modul S2 sebagai perangkat radio komunikasi
  4. MicroSd card adapter yang berfungsi untuk menampung data yang akan di kompresi

3.2 Arsitektur Umum Sistem

Teknologi wireless sensor network yang dikembangkan pada Tugas Akhir ini, menggunakan perangkat XBee S2 yang akan berperan sebagai perangkat yang membantu komunikasi antar *node* dengan protokol ZigBee sebagai jalur komunikasinya. Arsitektur jaringan yang digunakan adalah *cluster tree*, dimana terdapat tiga jenis perangkat ZigBee yang dibutuhkan, yaitu *node* ZigBee *Coordinator*, *node* ZigBee *Router* dan *node* ZigBee *End device*.



Gambar 3.2 Arsitektur Detail Sistem

Berdasarkan Gambar 3.2 sistem kompresi dan dekompresi data pada platform wireless sensor network memiliki alur proses yang akan dijabarkan sebagai berikut :

1. Pada ZigBee *Coordinator*, XBee Shield dipasang diatas Arduino Mega 2560, kemudian diatas XBee Shield tersebut dipasang modul XBee S2

2. Pada ZigBee *Router*, XBee Shield dipasang diatas Arduino Mega UNO, kemudian diatas XBee Shield tersebut dipasang modul XBee S2
3. Pada ZigBee *End device*, XBee Shield dipasang diatas Arduino Mega 2560, kemudian diatas XBee Shield tersebut dipasang modul XBee S2. Selain itu dipasang juga modul MicroSD Card Adapter dihubungkan menggunakan kabel jumper ke pin Arduino Mega.

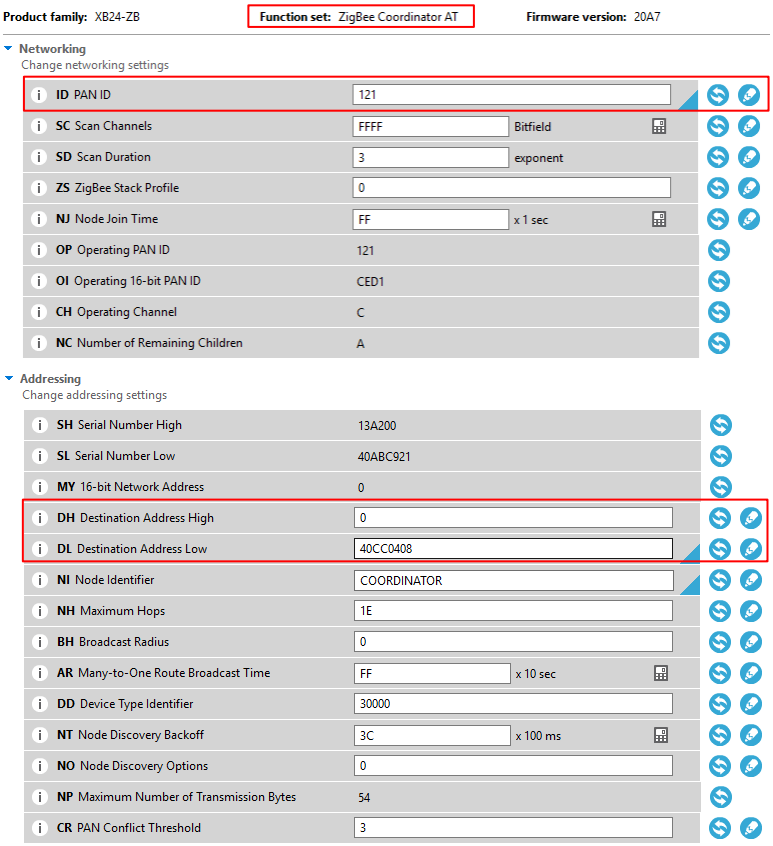
3.3 Perancangan Komunikasi Sistem

Perancangan komunikasi sistem merupakan salah satu proses penting yang harus dilakukan dalam membangun sistem agar *node* pada jaringan dapat saling berkomunikasi. Protokol jaringan komunikasi yang digunakan adalah protokol ZigBee, dimana pada jaringan tersebut minimal terdapat sebuah *node* yang bertindak sebagai *Coordinator*.

Pada penelitian Tugas Akhir ini, akan menggunakan satu buah *node* yang berfungsi sebagai ZigBee *End device*, satu buah *node* yang berfungsi sebagai ZigBee *Router* dan satu *node* sebagai ZigBee *Coordinator*. Agar setiap *node* dapat berkomunikasi, harus dilakukan konfigurasi terlebih dahulu terhadap *node Coordinator* dan *Router* dengan memanfaatkan aplikasi XCTU yang disediakan oleh Digi International Inc. Beberapa hal yang harus di perhatikan pada saat melakukan konfigurasi antara lain :

- **Function Set**
- **ID** PAN ID
- **DH** *Destination Address High*
- **DL** *Destination Address Low*

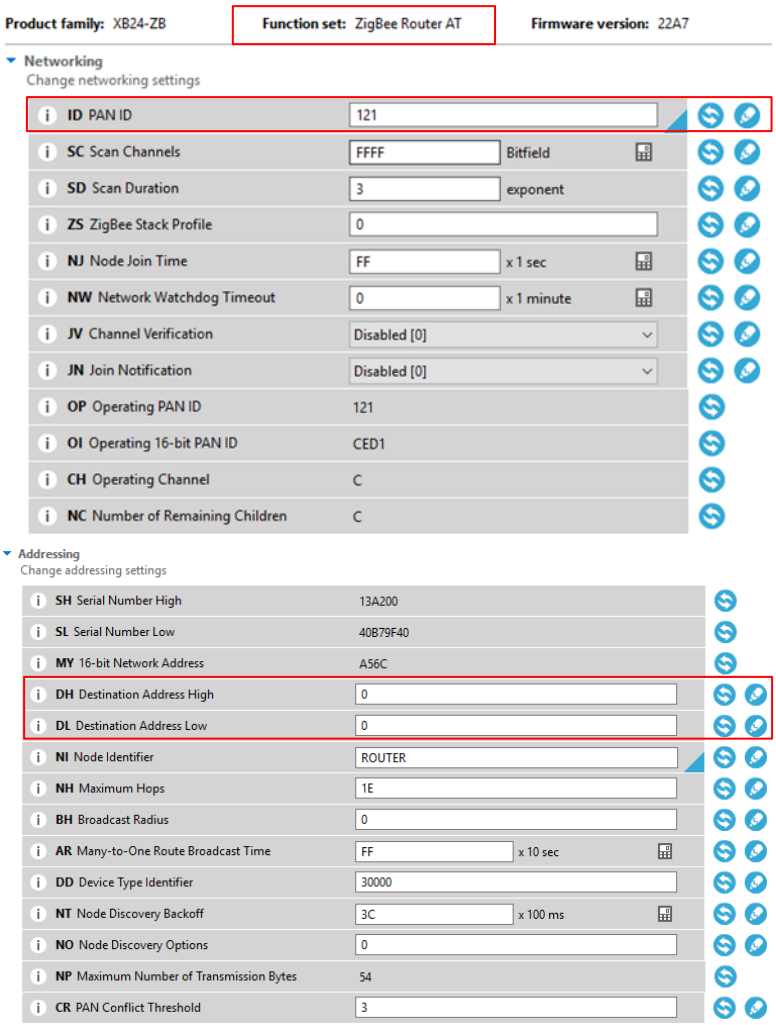
Berikut ini adalah konfigurasi secara detail yang digunakan pada ZigBee *Coordinator* ditunjukkan pada Gambar 3.3, konfigurasi ZigBee *Router* ditunjukkan pada Gambar 3.4 dan konfigurasi ZigBee *End device* ditunjukkan pada Gambar 3.5.



Gambar 3.3 Konfigurasi pada *Node* ZigBee *Coordinator*

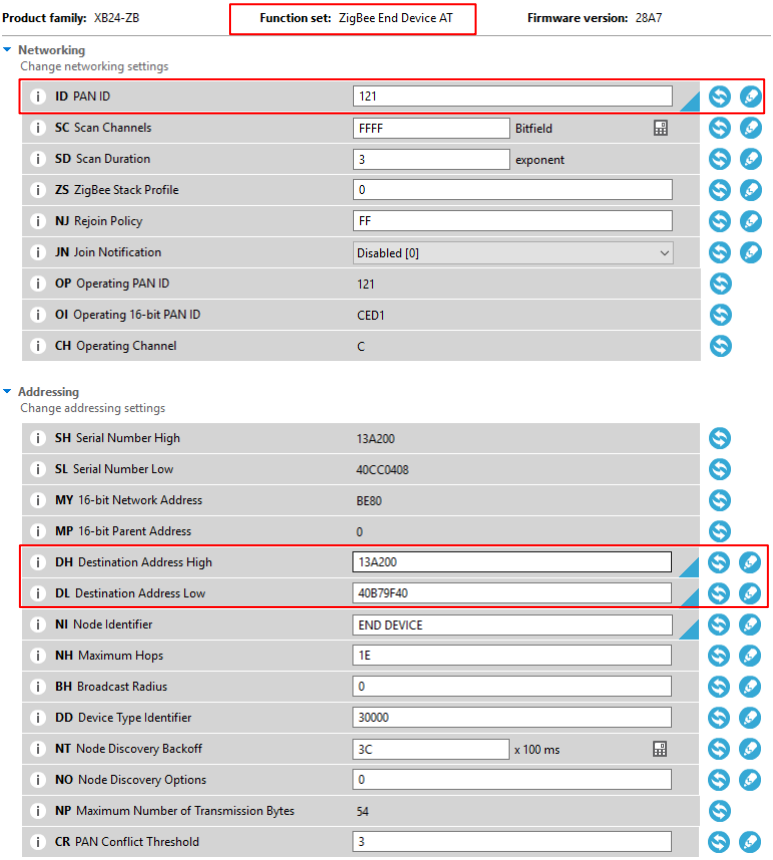
Function Set yang digunakan pada konfigurasi *node* ZigBee *Coordinator* adalah **ZigBee *Coordinator* AT**. Untuk **PAN ID (ID)** nya **121**. PAN ID ini memiliki fungsi yang hampir sama dengan subnet, dimana semua *node* yang ada pada jaringan ZigBee nilainya harus sama agar dapat saling berkomunikasi. ***Destination Address High* (DH)** yang digunakan yaitu **0** dan ***Destination Address Low* (DL)** yaitu **40CC0408**. Konfigurasi yang digunakan

bertujuan agar *node* ZigBee *Coordinator* hanya dapat menerima pesan dari *node* ZigBee *Router*.



Gambar 3.4 Konfigurasi pada *Node* ZigBee *Router*

Setelah mensetting *node Coordinator*, langkah selanjutnya adalah mensetting *node Router*. **Function Set** yang digunakan pada konfigurasi *node* ZigBee *Router* adalah **ZigBee *Router* AT**. Untuk **PAN ID (ID)** nilainya **121**. **Destination Address High (DH)** yang digunakan yaitu **0** dan **Destination Address Low (DL)** yaitu **0.** Konfigurasi yang digunakan bertujuan agar *node Router* dapat menerima pesan dari *node* ZigBee *End device* dan dapat meneruskannya ke *node* ZigBee *Coordinator*.



Gambar 3.5 Konfigurasi pada *Node* ZigBee *End device*

Function Set yang digunakan pada konfigurasi *node* ZigBee *Router* adalah **ZigBee *End device* AT**. Untuk **PAN ID (ID)** nilainya **121**. **Destination Address High (DH)** yang digunakan adalah **13A200**, dimana nilai ini didapat dari **Serial Number High (SH)** pada *node* ZigBee *Router*. Selainnya itu **Destination Address Low (DL)** nilainya **40B79F40** yang merupakan nilai **Serial Number Low (SL)** pada *node* ZigBee *Router*. Konfigurasi ini bertujuan agar *node* ZigBee *End device* dapat megirimkan pesan pada *node* ZigBee *Router*.

3.4 Perancangan Kompresi dan Dekompresi Data

Pada bagian ini akan dijelaskan bagaimana perancangan yang dilakukan sebelum data di kompresi. Untuk melakukan proses kompresi dan dekompresi, pada penelitian ini menggunakan algoritma Heatshrink, dimana algoritma ini berbasis pada algoritma Lempel-Ziv-Storer-Szymanski (LZSS).

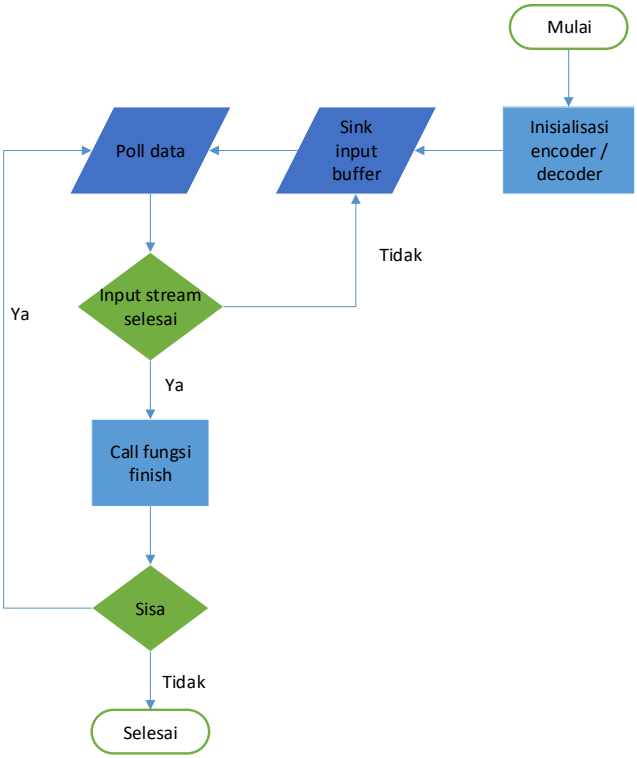
Kebutuhan memori merupakan suatu hal yang mendasar dalam melakukan perancangan kompresi dan dekompresi data, sebab jumlah memori yang tersedia pada Arduino sangat terbatas. Pada algoritma heatshrink ukuran *buffer* telah ditetapkan untuk memungkinkan terjadinya *trade-off* antara efektivitas kompresi dengan memori kerja. Persyaratan yang dibutuhkan untuk penggunaan *buffer* IO adalah sebagai berkut [15]:

a. Encoding
   $16 + 2 * 2^N$ *byte* untuk encoding, ditambah lagi untuk indeks pencarian optional yakni $2 * 2^N$ *byte* untuk mempercepat pengkodean

b. Decoding
   $16 + 2^N$ *byte* untuk decoding, dimana N dapat di atur pada saat pengodean (encoding)

3.4.1 Cara Kerja Algoritma Heatshrink

Lempel-Ziv-Storer-Szymanski (LZSS) membutuhkan sedikit ruang kerja dimana *cache* yang diperlukan adalah 2^N *byte* dari data terakhir (N dapat dikonfigurasi) dan proses dekompresi yang dilakukan cukup sederhana. Sistem yang dijalankan juga cukup sederhana, *dump* data kedalam, terus putar engkol sampai tidak ada lagi data yang keluar, *dump* data lebih banyak, ulangi proses tersebut. Notifikasi *encoder* / *decoder* ketika akhir input telah tercapai, lakukan sampai selesai. Pada Gambar 3.6 menjelaskan diagram alir cara kerja algoritma Heatshrink.



Gambar 3.6 Diagram Alir Cara Kerja Algoritma Heatshrink

Berikut ini akan dijelaskan cara kerja pada algoritma Heatshrink [19] secara detail :

1. Alokasikan **heatshrink_encoder** atau **heatshrink_decoder** pada *state machine* menggunakan fungsi **alloc** atau dapat mengunakan **static alloc** dan panggil fungsi reset untuk memulai inisialisasi
2. Gunakan **sink** untuk memasukkan input *buffer* kedalam state machine. Pointer pada input_size digunakan untuk menunjukkan seberapa banyak *byte* dari *buffer* input yang digunakan (jika nilainya 0 maka *buffer*-nya penuh)
3. Gunakan **poll** untuk memindahkan output dari state machine ke *buffer* output. Pointer pada output_size menunjukkan berapa banyak *byte* yang dihasilkan dan fungsi return menunjukkan apakah *output* selanjutnya tersedia (*state mechine* tidak boleh mengeluarkan data sampai ia menerima *input* yang cukup)
4. Ulangi langkah 2 dan 3 untuk melakukan *stream data* melalui *state machine*. Pada saat kompresi data, ukuran *input* dan *output* dapat bervariasi secara signifikan. *Looping* diperlukan untuk *buffer input* dan *output* dalam pemrosesan data.
5. Ketika input stream selesai, panggil fungsi **finish** untuk memberitahu bahwa *state machine* tidak lagi bisa menerima *input*. Nilai kembalian dari proses yang terlah selesai menunjukkan apakah ada output yang tersisa. Jika ada, panggil fungsi **poll** lagi.
6. Kemudian panggil fungsi **finish** dan *flush* sisa *output* hingga selesai, sampai sisa *output* habis.

3.4.2 Konfigurasi Algoritma Heatshrink

Heatshrink memiliki beberapa opsi konfigurasi yang dapat mempengaruhi penggunaan sumberdaya dan seberapa efektif ia dapat memampatkan data. Konfigurasi ini dapat diatur secara dinamis pada saat akan melakukan kompresi dan dekompresi atau dapat pula di setting statis pada file heatshrink_config.h. Adapun konfigurasi yang dimaksudkan adalah sebagi berikut :

- ***window_sz2***

  Ukuran *window* menentukan seberapa panjang input yang dapat dicari untuk pola yang berulang. Semakin besar ukuran *window* maka akan menggunakan memori semakin banyak, tetapi dapat melakukan kompresi lebih efektif dalam mendeteksi pengulangan yang lebih banyak. Sebuah window_sz2 = 8 akan menggunakan memori 256 *byte* (2^8), sedangkan window_sz2 = 10 akan menggunakan memori 1024 *byte* (2^10). Pengaturan window_sz yang tersedia adalah antara **4 sampai 15.**

- ***lookahead_sz2***

  Ukuran *lookahead* menentukan panjang maksimal unttuk pola berulang yang ditemukan. Jika lookahead_sz2 adalah 4, ‘a’ 50-bit dari karakter ‘a’ akan direpresentasikan sebagai pola 16-*byte* berulang (2^4). Jumlah bit yang digunakan unutk ukuran lookahead bersifat tetap, sehingga ukuran lookahead yang besar dapat mengurangi kompresi dengan menambahkan bit yang tidak digunakan ke pola-pola kecil. Pengaturan lookahead_sz2 yang ada saat ini adalah antara **3 sampai *window_sz* – 1.**

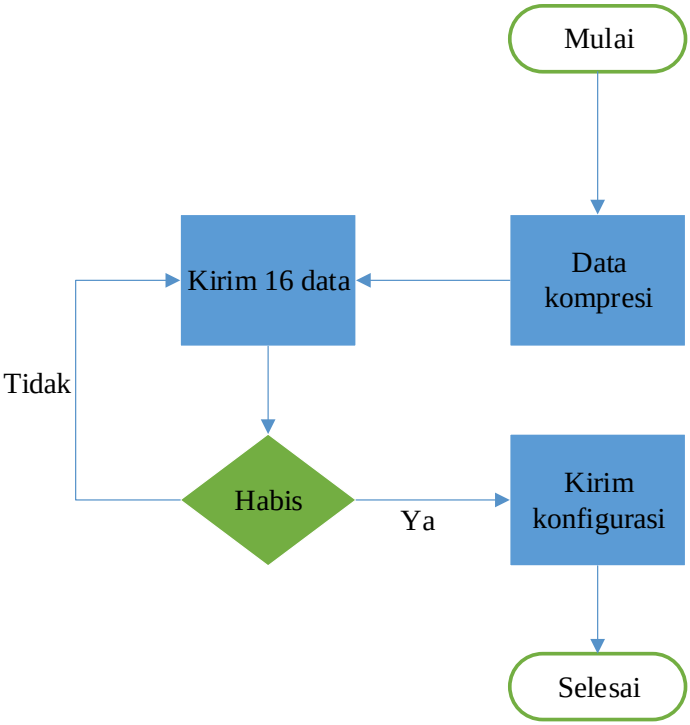
- ***input_buffer_size***

Besar atau kecilnya *buffer input* yang digunakan untuk decoder ditentukan oleh **input_*buffer*_size**. Ukuran *buffer* input berdampak pada seberapa banya pekerjaan yang dapat dilakukan decoder dalam satu langkah, dan semakin besar *buffer* maka memori yang dibutuhkan semakin banyak. *Buffer* yang sangat kecil (misalnya 1 *byte*) akan menambah *overhead* karena banyak banyak melakukan pemanggilan fungsi *suspend* / *resume*, akan tetapi ***input_buffer_size*** tidak mempengaruhi seberapa baik dalam melakukan kompresi data.

3.5 Perancangan Pengiriman Data

Pengiriman data adalah suatu hal yang perlu diperhatikan dalam perancangan sistem ini, sebab keterbatasan *buffer* yang ada pada protokol ZigBee yaitu sebesar 72 *byte* menjadi landasan dasar untuk melakukan perancangan pengiriman data yang tepat.

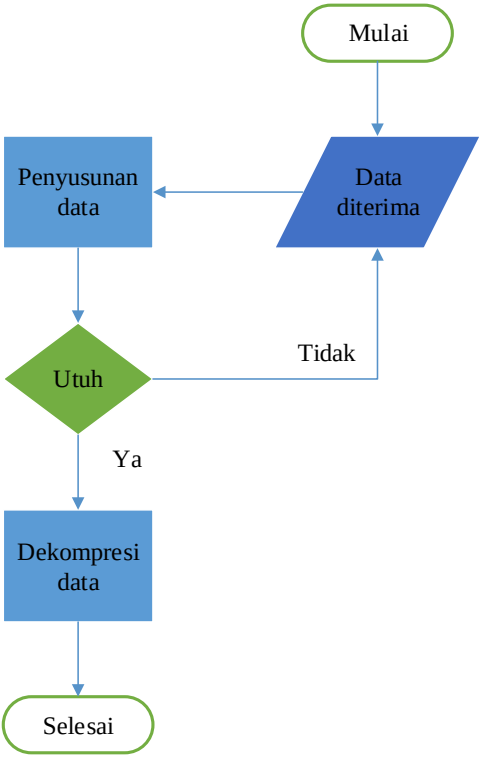
Pada penelitian ini protokol ZigBee menggunakan mode AT atau lebih dikenal mode "Transparan". Dalam mode AT, data akan segera dikirim ke modul jarak jauh yang diidentifikasi melalui alamat tujuan yang ada pada memori modul XBee. Alamat tujuan dapat di konfigurasi oleh pengguna pada mode Command. Jika XBee menggirimkan data ke *Coordinator* maka akan di broadcast pada PAN ID. Informasi paket tidak diperlukan, tetapi prosesnya lebih sederhana, dimana Serial data dikirimkan ke Tx dari satu XBee dan akan diterima oleh Rx tujuan XBee. Mode AT cocok digunakan pada jaringan yang sangat sederhana, karena tidak perlu untuk mengubah alamat tujuan terlalu sering. Pada Gambar 3.7 menjelaskan bagaimana proses pengiriman data dari *node* ZigBee *End device* menuju *node* ZigBee *Router*.



Gambar 3.7 Diagram Alir Pengiriman Data

3.6 Perancangan Dekompresi Data

Data yang dikirimkan oleh *node* ZigBee *Router* akan di terima oleh *node* ZigBee *Coordinator* melalui jaringan ZigBee. Potongan – potongan data tersebut akan ditampung terlebih dahulu dan kemudian akan di satukan kembali untuk di proses kembali. Proses dekompresi akan dilakukan ketika data dekompresi yang diterima sudah utuh beserta konfigurasi *encoder / decoder* yang digunakan. Pada Gambar 3.8 menjelaskan bagaimana proses dekompresi data.



Gambar 3.8 Diagram Alir Dekompresi Data

3.7 Perancangan Perangkat Keras

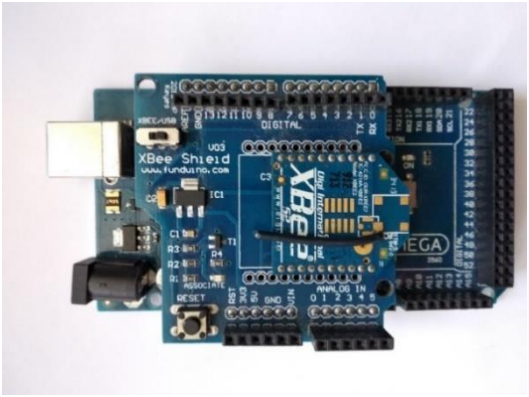
Perancangan perangkat keras secara umum menjelaskan mengenai penempatan perangkat keras yang digunakan dalam membangun sistem, yang mana terdiri dari rangkaian ZigBee *Coordinator*, ZigBee *Router* dan ZigBee *End device*. Rangkaian perangkat keras pada sistem dapat dilihat pada Gambar 3.9 dan Gambar 3.10. Agar sistem dapat berjalan sebagai mestinya, terdapat beberapa komponen yang dibutuhkan antara lain:

1. Dua buah Arduino Mega

2. Satu buah Arduino UNO
3. Satu buah baterai 9V
4. Tiga buah XBee Shield
5. Tiga buah XBee S2
6. Satu buah MicroSD Card Adapter
7. Satu buah SD Card

3.8 Perancangan Perangkat ZigBee *Coordinator*

Pada rangkaian ZigBee *Coordinator*, XBee Shield V03 pabrikan Funduino menempati tepat di bagian atas Arduino Mega 2560 dengan posisi pin Tx dan Rx yang sama dengan pin Tx dan Rx pada XBee Shield. Kemudian modul XBee S2 Pro di letakkan pada space yang telah disediakan pada XBee Shield. Pastikan kepala modul XBee S2 menghadap arah yang berlawanan dengan konektor USB.



Gambar 3.9 *Node* ZigBee *Coordinator*

3.9 Perancangan Perangkat ZigBee *Router*

Pada rangkaian ZigBee *Router*, XBee Shield V1.1 pabrikan ITead Studio menempati tepat di bagian atas Arduino UNO dengan

posisi pin Tx dan Rx yang sama dengan pin Tx dan Rx pada XBee Shield. Kemudian modul XBee S2 Pro di letakkan pada space yang telah disediakan pada XBee Shield. Pastikan kepala modul XBee S2 menghadap arah yang berlawanan dengan konektor USB.

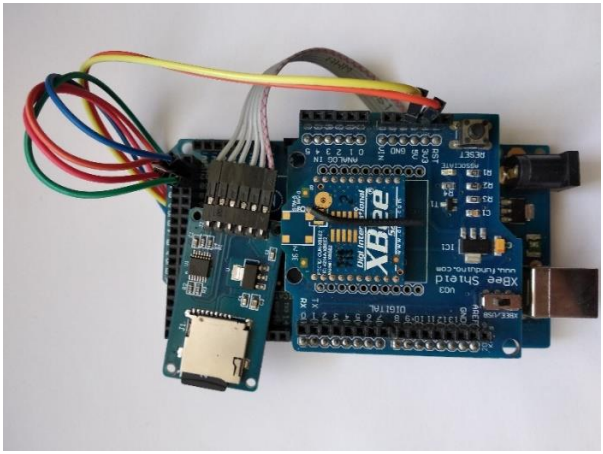
Gambar 3.10 *Node* ZigBee *Router*

3.10 Perancangan Perangkat ZigBee *End device*

Pada rangkaian ZigBee *End device* posisinya hampir sama dengan rangkaian ZigBee *Coordinator*. Xbee Shield diletakkan diatas Arduino Mega dengan posisi pin Tx dan Rx yang sama. Selain itu letakkan modul XBee S2 pada tempat yang telah disediakan. Pada perangkat ZigBee *Router* membutuhkan MicroSD Card Adapter yang berfungsi untuk menampung data yang akan di kompresi. Pastikan sudah terdapat SD Card pada modul tersebut, kemudian sambungkan pin yang terdapat pada MicroSD Card Adapter kepada pin yang sudah ditentukan, berikut ini rincian pinnya :

- Pin **CS** dihubungkan dengan pin **digital 53**
- Pin **SCK** dihubungkan dengan pin **digital 52**
- Pin **MOSI** dihubungkan dengan pin **digital 51**
- Pin **MISO** dihubungkan dengan pin **digital 50**

- Pin **VCC** dihubungkan dengan pin **tegangan 5V**
- Pin **GND** dihubungkan dengan pin **GND**



Gambar 3.11 *Node* ZigBee *End device* disertai MicroSD

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi berupa pseudocode untuk membangun program. Cakupan implementasi dari perancangan sistem ini meliputi perangkat *node* ZigBee *Router* yang bertugas untuk melakukan kompresi data dan mengirimkan hasilnya ke perangkat *node* ZigBee *Coordinator*, kemudian akan di dekompresi berdasarkan konfigurasi yang telah diterima sebelumnya. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan suatu lingkungan dimana sistem akan dibangun. Untuk mempermudah penjelasan, lingkungan implementasi akan terbagi menjadi dua bagian. Pembahasan pertama mengenai lingkungan implementasi perangkat keras dan pembahasan kedua mengenai lingkungan implementasi perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Pada bagian ini akan dibahas mengenai perangkat keras apa saja yang dibutuhkan untuk membangun sistem. Lingkungan implementasi perangkat keras dari sistem yang akan dibangun secara lebih lengkap dijelaskan pada Tabel 4.1 dibawah ini

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

Perangkat	Detail Perangkat
Perangkat Komputer	Model : • Lenovo Y410P
	Manufaktur :

	• Lenovo
	Processor : • Intel® Core™ i7-4700MQ (2.40GHz 1600MHz 6MB)
	Memori : • 8GB PC3-12800 DDR3L SDRAM 1600 MH
Perangkat Mikrokontroler	Mikrokontroler : • ATmega2560
	Model : • Arduino Mega 2560 (a) • Ardino UNO (b)
	Tegangan : • 5 V (a) • 5 V (b)
	Memori Flash : • 256 KB (8KB digunakan untuk bootloader) (a) • 32 KB (0.5KB digunakan untuk bootloader) (b)
	SRAM : • 8 KB (a) • 2 KB (b)
Perangkat XBee Shield	Model: • XBee Shield V0.3 (a) • XBee Shield V1.1 (b)
	Manufaktur : • Funduino (a) • ITead Studio (b)

	Tipe jumper : • Switch (a) • Pasang lepas (b)
Perangkat Modul XBee	Model : • XBee S2
	Manufaktur : • Digi International Inc.

4.1.2 Lingkungan Implementasi Perangkat Lunak

Pada bagian ini akan dibahas mengenai perangkat lunak apa saja yang dibutuhkan untuk membangun sistem. Lingkungan impementasi perangkat lunak dari sistem yang akan dibangun secara lebih detail akan dijelaskan pada Tabel 4.2 dibawah ini.

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak

Perangkat	Detail Perangkat
Perangkat Lunak	Sistem Operasi : • Microsoft Windows 10 Pro 64-bit
	Software Arduino : • Arduino IDE 1.8.5
	Software XBee : • DIGI XCTU 6.3.13

4.2 Implementasi Perangkat Keras

Implementasi perangkat keras untuk penelitian Tugas Akhir ini adalah sebagai berikut :

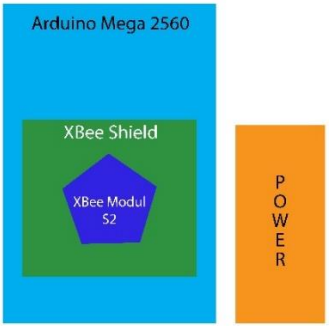
- 2 (dua) buah Arduino Mega
- 1 (satu) buah Arduino UNO
- 3 (tiga) buah XBee Shield

- 3 (tiga) buah XBee S2
- 1 (satu) buah MicroSD Card Adapter
- 1 (satu) buah SD Card
- 1 (satu) buah baterai 9 volt

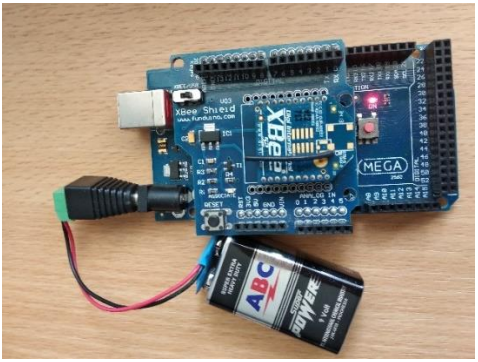
Pada *node End device*, terdapat sdcard yang menampung data yang akan dikompresi. Setelah data selesai dikompresi, data kemudian dikirimkan ke *node* ZigBee *Router* beserta konfigurasi yang telah digunakan. Kemudian *node* ZigBee *Router* meneruskan data yang diterima ke *node* ZigBee *Coordinator* untuk dilakukan dekompresi berdasarkan konfigurasi yang ditetapkan. Perancangan serta implementasi perangkat *node* ZigBee *Coordinator*, *node* ZigBee *Router* dan *node* ZigBee *End device* akan dijabarkan lebih mendetail pada Gambar 4.2, Gambar 4.4 dan Gambar 4.6

4.2.1 Perangkat *Node* ZigBee *Coordinator*

Perangkat *node* ZigBee *Coordinator* tersusun dari Arduino Mega, XBee Shield, dan modul XBee S2. Zigbee *Coordinator* memiliki peranan dalam menerima data dari ZigBee *Router* dan akan mendekompresi data yang telah diterima. Pada Gambar 4.1 memperlihatkan perancangan dari *node* ZigBee *Coordinator*, sedangkan pada Gambar 4.2 merupakan implementasi dari *node* ZigBee *Coordinator*.



Gambar 4.1 Perancangan *Node* ZigBee *Coordinator*

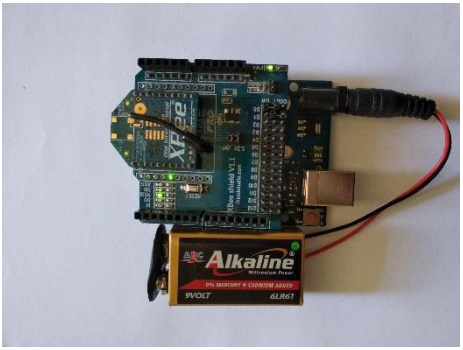


Gambar 4.2 Implementasi *Node* ZigBee *Coordinator*

4.2.2 Perangkat *Node* ZigBee *Router*

Perangkat *node* ZigBee *Router* tersusun dari Arduino UNO, XBee Shield, dan modul XBee S2 dan MicroSD Card Adapter. ZigBee *Router* memiliki peranan sebagai penghubung antara *node* ZigBee *Router* dengan *node* ZigBee *End device*. Pada Gambar 4.3 memperlihatkan perancangan dari *node* ZigBee *Router* dan Gambar 4.4 merupakan implementasi dari *node* ZigBee *Router*.

Gambar 4.3 Perancangan *Node* ZigBee *Router*

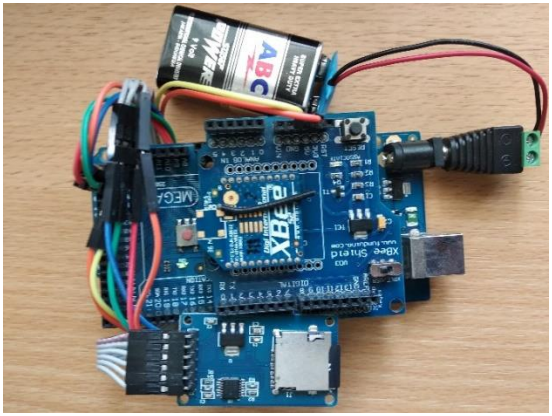


Gambar 4.4 Implementasi *Node* ZigBee *Router*

4.2.3 Perangkat *Node* ZigBee *End device*

Perangkat *node* ZigBee *Router* tersusun dari Arduino Mega, XBee Shield, dan modul XBee S2 dan MicroSD Card Adapter. ZigBee *End device* memiliki peranan dalam mengambil data dari *sd card* melakukan kompresi data dan mengirimkan data tersebut menuju *node* ZigBee *Router*. Pada Gambar 4.5 memperlihatkan perancangan dari *node* ZigBee *End device* dan Gambar 4.6 merupakan implementasi dari *node* ZigBee *End device*.

Gambar 4.5 Perancangan *Node* ZigBee *End device*



Gambar 4.6 Implementasi *Node* ZigBee *End device*

4.3 Implementasi Inisialisasi Data Pada SD Card

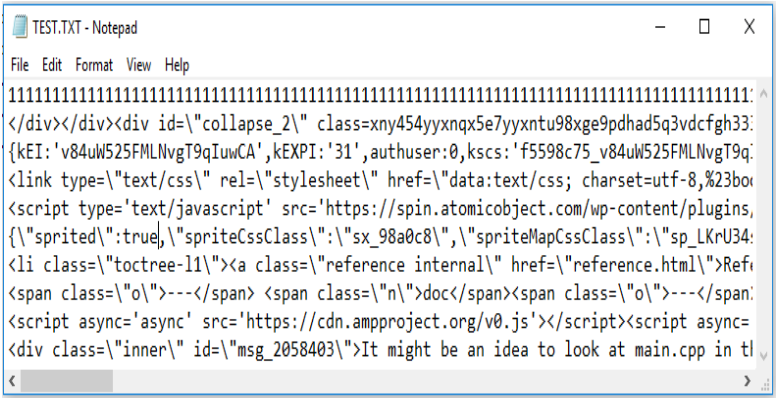
Data yang akan dikompresi berupa string diletakkan pada SD Card. Sebelum melakukan kompresi data, terlebih dahulu kita harus menginisialisasi data. Pada penelitian ini terdapat lima jenis data yang akan dikompresi, dimana masing – masing data memiliki panjang yang berbeda. Data tersebut akan disimpan dalam bentuk file txt. Berikut ini adalah kelima data tersebut :

- Data string dengan panjang 584 karakter
- Data string dengan panjang 980 karakter
- Data string dengan panjang 1280 karakter
- Data string dengan panjang 1345 karakter
- Data string dengan panjang 1870 karakter

Setiap file menyimpan data dengan panjang yang sama. Sebagai contohnya file data_584.txt menyimpan data dengan panjang 584 karakter, dimana setiap baris data yang ada pada file tersebut panjangnya sama yaitu 584 karakter.

Pada Gambar 4.7 merupakan contoh file txt dengan panjang string 584 karakter per baris. Pada Gambar 4.8 merupakan contoh file txt dengan panjang string 980 karakter per baris. Pada Gambar

4.9 merupakan contoh file txt dengan panjang string 1280 karakter per baris. Pada Gambar 4.10 merupakan contoh file txt dengan panjang string 1345 karakter per baris. Pada Gambar 4.11 merupakan contoh file txt dengan panjang string 1870 karakter per baris.

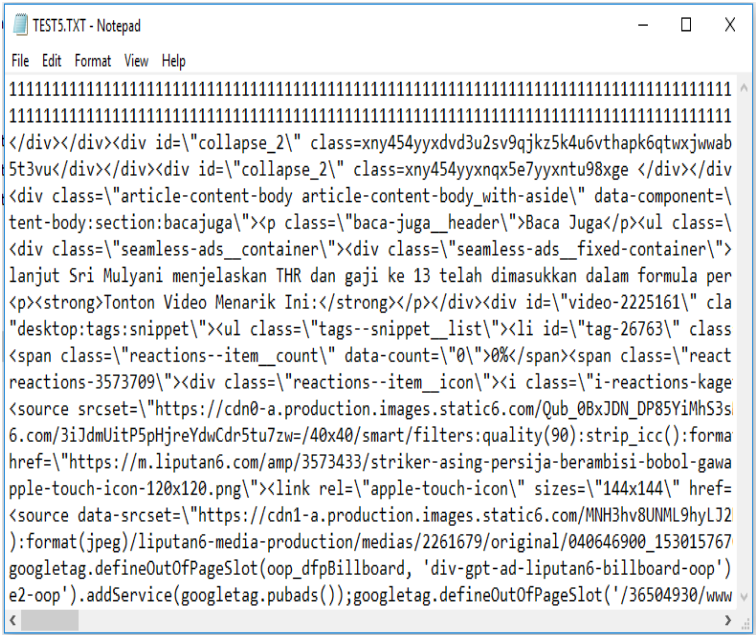


Gambar 4.7 Data dengan Panjang 584 Karakter per Baris

Gambar 4.8 Data dengan Panjang 980 Karakter per Baris

Gambar 4.9 Data dengan Panjang 1280 Karakter per Baris

Gambar 4.10 Data dengan Panjang 1345 Karakter per Baris



Gambar 4.11 Data dengan Panjang 1870 Karakter per Baris

4.4 Implementasi Membaca Data dari SD Card

Setelah data dibagi kedalam masing-masing file, maka data akan siap di kompresi. Pada *node Router* data akan diambil dari *sd card* per baris. Kemudian data tersebut di akan di ubah kedalam array lalu siap untuk kompresi dengan konfigurasi yang telah ditentukan. Pada Kode Sumber 4.1 diperlihatkan bagaimana proses *read* data perbaris dari *sdcard* memori.

```
1  function readData()
2      myFile ← load DataFile
3      while(myFile.available()) do
4          data ← myFile.readByLine
5          data.toArray(data_test, data_sz)
6      endwhile
7  endfunction
```

Kode Sumber 4.1 *Pseuducode* Membaca Data dari sdcard

4.5 Implementasi Setting Konfigurasi *Encoder / Decoder*

Implementasi setting konfigurasi *encoder / decoder* bertujuan untuk melakukan setting konfigurasi yang akan digunakan *encoder* dalam proses kompresi data dan decoder dalam dekompresi data. Pada Kode Sumber 4.2 diperlihatkan cara melakukan konfigurasi pada *encoder / decoder*. Adapun konfigurasi yang akan digunakan yaitu sebagai berikut :

- window_sz menggunakan konfigurasi antara 4 sampai 8
- lookahead_sz menggunakan konfigurasi nilai 3 sampai window_sz – 1
- input_*buffer*_size yang digunakan adalah 64

```
1  function setConfiguration()
2      Configuration cfg := new Configuration()
3      cfg.log_level ← value
4      cfg.window_sz ← value
5      cfg.lookahead_sz ← value
6      cfg.decoder_input_buffer_sz ← value
7  endfunction
```

Kode Sumber 4.2 *Pseuducode* Setting Konfigurasi *Encoder / Decoder*

4.6 Implementasi Kompresi Data

Implementasi kompresi data yang dilakukan pada penelitian ini menggunakan *library* Heatshrink yang merupakan salah satu algoritma kompresi data yang berbasis pada algoritma Lempel-Ziv-Storer-Szymanski (LZSS). Pada Kode Sumber 4.3 diperlihatkan mekanisme kompresi data yang dilakukan oleh sistem.

```
1  function compressData(input, input_sz, cfg)
2      heatshrink_encoder hse ←
.      heatshrink_encoder_alloc(cfg)
3      comp ← malloc
4      while(sunk < input_sz) do
```

```
5           HSE_sink_res esres ←
.           heatshrink_encoder_sink(hse,
.           input[sunk], input_sz – sunk, count)
6           sunk ← sunk + count
7           if(sunk == input_sz) then
8              heatshrink_encoder_finish(hse)
9           endif
10          HSE_poll_res pres
11          do
12             pres ←  heatshrink_encoder_poll(hse,
.              comp[polled], comp_sz – polled,
.              count)
13             polled ← polled + count
14          while (pres == HSER_POLL_MORE)
15          if(sunk == input_sz) then
16              heatshrink_encoder_finish(hse)
17        endif
18     endwhile
19     free(comp)
20     heasthrink_encoder_free(hse)
21     return polled
22  endfunction
```

Kode Sumber 4.3 *Pseudocode* Kompresi Data

4.7 Implementasi Mekanisme Pengiriman Data

Pada penelitian ini, pengiriman data yang dilakukan oleh *node* ZigBee *Router* menggunakan mode *Router* AT. Proses pengiriman data memanfaatkan kelas Serial yang ada pada Arduino. Data hasil kompresi yang akan dikirimkan akan dipecah menjadi beberapa bagian kecil. Dalam satu kali pengiriman jumlah data yang dapat di kirimkan adalah sebanyak 16 buah, dimana terdapat delay 3 detik untuk setiap pengiriman data. Setelah data kompresi dikirimkan semua, konfigurasi *encoder* / *decoder* yang digunakan juga akan dikirimkan ke *node* ZigBee *Coordinator*. Pada Kode Sumber 4.4 akan diperlihatkan mekanisme pengiriman data yang akan dilakukan melalui protokol ZigBee dengan memanfaatkan kelas Serial pada Arduino.

```
1   function sendData(comp, cfg, polled, length_data)
2       Serial(comp[0])
3       for i=1 to polled do
4         if(i mod 16 == 0) then
5           Serial(comp[i])
6           Serial("\n")
7         else
8           Serial(comp[i])
9           Serial("\n")
10        endif
11      endfor
12      delay()
13      Serial(length_data)
14      Serial(cfg.window_sz)
15      Serial(cfg.lookahead_sz)
16      Serial(cfg.decoder_input_buffer_sz)
17     Serial(polled)
18  endfunction
```

Kode Sumber 4.4 *Pseuducode* Mekanisme Pengiriman Data

4.8 Implementasi Dekompresi Data

Potongan – potongan data yang dikirimkan oleh *node* ZigBee *Router* akan disatukan kembali pada *Node* ZigBee *Coordinator*. Setelah data hasil kompresi utuh, maka proses dekompresi data akan siap dilakukan dengan konfigurasi *encoder* / *decoder* yang dikirimkan juga oleh *node* ZigBee *Router*. Pada Kode Sumber 4.5 akan dilakukan pembentukan ulang data dari data yang telah diterima, kemudian akan dilakukan dekompresi data sesuai konfigurasi encoder / decoder yang digunakan yang akan dipreliihatkan pada Kode Sumber 4.6

```
1   function reformingData()
2       if(Serial.available() > 0) then
3          incoming Byte ← Serial.read()
4          if(incoming Byte != "\n") then
5             stringData ← stringData + incoming Byte
```

```
6           elseif( incomingByte == "\n")) then
7              num[idx] ← stringData.toInt()
8              idx++
9           endif
10      endif
11  endfunction
```

Kode Sumber 4.5 *Pseuducode* Pembentukan Ulang Data

```
1   function decompressData(input, input_sz, cfg,
.            output, output_sz, polled)
2       heatshrink_decoder hsd ←
.       heatshrink_decoder_alloc(cfg)
3       comp_sz ← polled
4       while(sunk < comp_sz) do
5              heatshrink_decoder_sink(hsd,
.              input[sunk], comp_sz - sunk, count)
6           sunk ← sunk + count
7           if(sunk == comp_sz) then
8              heatshrink_decoder_finish(hsd)
9           endif
10          HSE_poll_res pres
11          do
12             pres ←  heatshrink_decoder_poll(hsd,
.              output[polled], output_sz - polled,
.              count)
13             polled ← polled + count
14          while (pres == HSDR_POLL_MORE)
15          if(sunk == comp_sz) then
16              heatshrink_dencoder_finish(hsd)
17          endif
18      endwhile
19  endfunction
```

Kode Sumber 4.6 *Pseuducode* Dekompresi Data

BAB V
HASIL UJI COBA DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada Tugas Akhir yang telah dikerjakan. Uji coba yang akan dilakukan secara garis besar terdiri dari uji coba fungsionalitas dan uji coba performa. Mekanisme uji coba dilakukan dengan menjalankan serangkaian skenario yang telah ditentukan. Pengujian fungsionalitas meliputi uji coba setiap bagian perangkat keras yang dirangkai pada Arduino dan uji coba keseluruhan sistem. Sedangkan pengujian performa meliputi efektifitas hasil kompresi, waktu yang dibutuhkan untuk kompresi dan dekompresi serta energi yang dapat dihemat oleh sistem. Bagian akhir dari bab ini akan membahas mengenai evaluasi dari serangkaian uji coba yang telah dilakukan.

5.1 Lingkungan Uji Coba

Lingkungan pelaksanaan uji coba meliputi perangkat keras dan perangkat lunak yang akan digunakan pada sistem ini. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam rangka uji coba perangkat lunak ini dicantumkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Uji Coba

Perangkat Keras	
Node ZigBee *Coordinator*	Rangkaian dari :
	- Arduino Mega 2560
	- XBee Shield
	- XBee Modul S2
Node ZigBee *Router*	Rangkaian dari :
	- Arduino UNO
	- XBee Shield

	- XBee Modul S2
Node ZigBee *End device*	Rangkaian dari :
	- Arduino Mega 2560
	- XBee Shield
	- XBee Modul S2
	- MicroSD Card Adapter
	- Baterai 9 Volt
	- SD Card
Perangkat Lunak	
Sistem Operasi	Microsoft Windows 10 Pro 64-bit
Software Arduino	Arduino IDE 1.8.5
Software XBee	DIGI XCTU 6.3.13

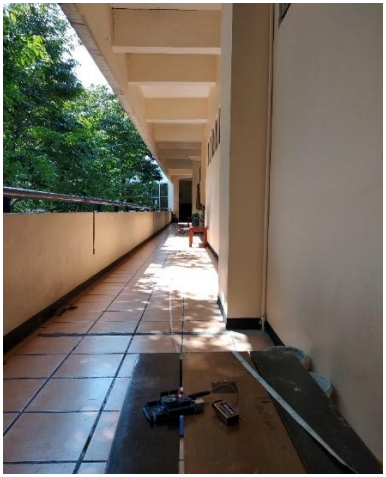
Terdapat tiga lokasi berbeda yang dijadikan tempat untuk uji coba. Lokasi pertama seperti Gambar 5.1 berada di Laboratorium KBJ Departemen Informatika ITS

Gambar 5.1 Lokasi Pertama

Lokasi kedua yang dijadikan tempat untuk uji coba seperti Gambar 5.2 berada di Lapangan Departemen Informatika ITS

Gambar 5.2 Lokasi Kedua

Sedangkan untuk lokasi ketiga seperti Gambar 5.3 berada di Lantai 3 Departemen Informatika ITS.



Gambar 5.3 Lokasi Ketiga

5.2 Data Pengujian

Subbab ini menjelaskan mengenai data yang digunakan pada uji coba. Seperti yang telah dijelaskan sebelumnya, data yang akan digunakan untuk melakukan pengujian harus di buat terlebih dahulu. Data yang digunakan untuk mengujian berupa data string dimana data akan terbagi menjadi lima jenis data yang berbeda berdasarkan panjangnya. Berikut ini adalah kelima data tersebut :

- Data string dengan panjang 584 karakter
- Data string dengan panjang 980 karakter
- Data string dengan panjang 1280 karakter
- Data string dengan panjang 1345 karakter
- Data string dengan panjang 1870 karakter

5.3 Skenario Uji Coba Fungsionalitas

Pengujian ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi benar-benar diimplementasikan dan bekerja sebagaimana seharusnya. Pengujian juga dilakukan untuk mengetahui kesesuaian setiap tahapan atau langkah penggunaan fitur terhadap skenario yang dipersiapkan.

5.3.1 Skenario Uji Coba Membaca Data dari SD Card

Pada skenario uji coba ini, *node* ZigBee *End device* akan membaca data yang ada pada sd card. Data tersebut dibaca per baris dengan jumlah total 80 baris, kemudian data akan ditampung kedalam sebuah array dan siap untuk di kompresi. Tabel 5.2 merupakan tabel skenario uji coba membaca data dari *sdcard* memori.

Tabel 5.2 Skenario Uji Coba Membaca Data dari SD Card

ID	UJ – F01
Nama	Uji Coba Membaca Data dari SD Card

Tujuan Uji Coba	Menguji fungsionalitas sistem untuk membaca data dari sd card
Kondisi Awal	*Node* ZigBee *End device* diaktifkan
Skenario	1. *Node* ZigBee *End device* membaca data dari *sdcard* 2. Data dibaca per baris 3. Kegiatan ini akan terus berulang hingga data pada baris terakhir. 4. Mengulangi langkah 1 – 3 dengan data yang panjang karakternya 980 5. Mengulangi langkah 1 – 3 dengan data yang panjang karakternya 1280 6. Mengulangi langkah 1 – 3 dengan data yang panjang karakternya 1345 7. Mengulangi langkah 1 – 3 dengan data yang panjang karakternya 1870
Masukan	Data yang ada pada sd card
Keluaran	Menampilkan data dibaca pada serial monitor
Hasil yang Diharapkan	Data dapat terbaca dengan baik

5.3.2 Skenario Uji Coba Kompresi Data

Pada skenario uji coba ini, data yang telah di baca akan mengalami proses kompresi. Kompresi data akan dilakukan dengan berbagai jenis konfigurasi *encoder / decoder*. Tabel 5.3 adalah tabel skenario uji coba kompresi data pada sistem.

Tabel 5.3 Skenario Uji Coba Kompresi Data

ID	UJ – F02
Nama	Uji Coba Kompresi Data

Tujuan Uji Coba	Menguji fungsionalitas sistem untuk melakukan kompresi dengan berbagai konfigurasi dan berbagai data yang memiliki panjang berbeda - beda
Kondisi Awal	*Node* ZigBee *End device* diaktifkan
Skenario	1. *Node* ZigBee *End device* membaca data dengan panjang 584 karakter 2. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (4,3) 3. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (5,4) 4. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (6,5) 5. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (7,6) 6. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (8,7) 7. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (9,8) 8. Mengulangi langkah 1 – 7 dengan data uji coba yang memiliki panjang 980 karakter 9. Mengulangi langkah 1 – 7 dengan data uji coba yang memiliki panjang 1280 karakter 10. Mengulangi langkah 1 – 7 dengan data uji coba yang memiliki panjang 1345 karakter 11. Mengulangi langkah 1 – 7 dengan data uji coba yang memiliki panjang 1870 karakter
Masukan	Data string yang akan dikompresi
Keluaran	Data hasil kompresi
Hasil yang Diharapkan	*Node* ZigBee *End device* berhasil mengkompresi data

5.3.3 Skenario Uji Coba Komunikasi Pada Topologi Cluster Tree

Pada skenario uji coba ini, *node* ZigBee *Coordinator* akan terkoneksi dengan *node* ZigBee *Router* dan *node* ZigBee *Router* terkoneksi dengan *node* ZigBee *End device*. Uji coba ini bertujuan untuk mengetahui apakah data yang dikirimkan dari *node* ZigBee *End device* diterima oleh *node* ZigBee *Coordinator*. Data yang akan dikirimkan berupa string dan hasilnya dapat dilihat melalui Serial monitor. Tabel 5.4 adalah tabel mengenai skenario uji coba komunikasi yang dilakukan pada topologi jaringan cluster tree yang digunakan.

Tabel 5.4 Skenario Uji Coba Komunikasi pada Topologi *Cluster Tree*

ID	UJ – F03
Nama	Uji Coba Komunikasi pada Topologi *Cluster Tree.*
Tujuan Uji Coba	Menguji fungsionalitas *node* ZigBee *Coordinator* untuk dapat terhubung dengan *node End device.*
Kondisi Awal	Ketiga *node* ZigBee diaktifkan. Ketiga *node* berada dalam mode XBee. Data yang akan dikirimkan sudah disiapkan.
Skenario	1. *Node* ZigBee *End device* mengirimkan data hasil kompresi ke *node* ZigBee *Router* 2. *Node* ZigBee *Router* menerima data dari *node End device* 3. Melakukan Pemantauan data yang diterima melalui *serial monitor*

	4. *Node* ZigBee *Router* mengirimkan data hasil kompresi ke *node* ZigBee *Coordinator* 5. Melakukan Pemantauan data yang diterima melalui *serial monitor node Coordinator*
Masukan	Data hasil kompresi
Keluaran	Tampilan data yang diterima pada serial monitor
Hasil yang Diharapkan	*Node* ZigBee *Coordinator* dapat menerima data yang dikirimkan oleh *node* ZigBee *End device*

5.3.4 Skenario Uji Coba Dekompresi Data

Pada skenario uji coba ini, potongan – potongan data hasil kompresi yang diterima oleh *node* ZigBee *Coordinator* akan di bentuk kembali sampai utuh. Kemudian data di dekompresi sesuai konfigurasi yang diterima. Tabel 5.5 adalah tabel skenario uji coba dekompresi data pada sistem.

Tabel 5.5 Skenario Uji Coba Dekompresi Data

ID	UJ – F04
Nama	Uji Coba Dekompresi Data
Tujuan Uji Coba	Menguji fungsionalitas sistem untuk melakukan dekompresi data dari potongan – potongan data yang diterima oleh *node* ZigBee *Coordinator*
Kondisi Awal	Ketiga *node* ZigBee diaktifkan. Ketiga *node* berada dalam mode XBee.
Skenario	1. *Node* ZigBee *Coordinator* menerima data hasil kompresi

	2. *Node* ZigBee *Coordinator* menyusun potongan – potongan data yang diterima 3. *Node* ZigBee *Coordinator* melakukan proses dekompresi 4. Melakukan pemantauan hasil dekompresi melalui *serial monitor*
Masukan	Data hasil kompresi
Keluaran	Tampilan data yang hasil dekompresi data pada serial monitor *node* ZigBee *Coordinator*
Hasil yang Diharapkan	*Node* ZigBee *Coordinator* dapat melakukan penyusunan potongan – potongan data kemudian di lakukan proses dekompresi

5.3.5 Skenario Uji Coba Kompresi *Adaptive*

Pada skenario uji coba ini, akan dilakukan kompresi data dengan settingan konfigurasi window_sz dan lookahead_sz sesuai dengan skenario yang digunakan. Konfigurasi window_sz dan lookahead_sz bersifat adaptive tergantung panjang data yang akan dikompresi. Jika panjang data **kurang dari 980** maka konfigurasi yang digunakan adalah **window_sz = 9 dan lookahead_sz = 8**. Jika panjang datanya **antara 980 sampai 1345** maka konfigurasi yang digunakan adalah **window_sz = 8 dan lookahead_sz = 7.** Jika panjang datanya **antara 1346 sampai 1870** maka konfigurasi yang digunakan adalah **window_sz = 7 dan lookahead_sz = 6.** Tabel 5.6 merupakan tabel skenario uji coba kompresi *adaptive* pada sistem.

Tabel 5.6 Skenario Uji Coba Kompresi *Adaptive*

ID	UJ – F05
Nama	Uji Coba Kompresi *Adaptive*

Tujuan Uji Coba	Menguji fungsionalitas sistem untuk melakukan kompresi *adaptive* berdasarkan panjang data
Kondisi Awal	*Node* ZigBee *End device* diaktifkan
Skenario	1. *Node* ZigBee *End device* membaca data satu baris pada sdcard 2. *Node* ZigBee *End device* melakukan kompresi data dengan konfigurasi window_sz dan lookahead_sz yang telah ditentukan sebelumnya 3. Mengulangi langkah 1 dan 2 sampai data terakhir pada sd card
Masukan	Data string yang akan dikompresi
Keluaran	Tampilan hasil kompresi data pada serial monitor *node* ZigBee *End device*
Hasil yang Diharapkan	*Node* ZigBee *End device* dapat melakukan kompresi adaptive

5.4 Hasil Uji Coba Fungsionalitas

Telah dijabarkan pada bagian sebelumnya mengenai skenario dari keseluruhan uji coba fungsionalitas. Skenario uji coba yang telah dijabarkan terdiri dari beberapa hal. Secara garis besar, pengujian fungsionalitas terdiri dari empat bagian, yaitu fungsionalitas membaca data, komunikasi *node*, kompresi data dan dekompresi data.

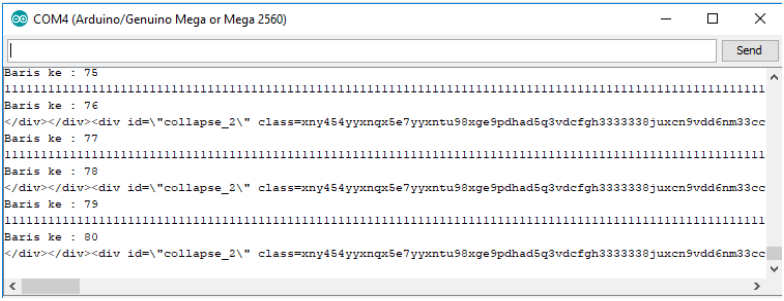
5.4.1 Hasil Uji Coba (UJ-F01) – Membaca Data dari SD Card

Pengujian dilakukan berdasarkan skenario uji coba pada bab 5.3.1. Langkah pertama yang dilakukan *node* ZigBee *Router*

membaca data yang ada pada *sdcard*, data dibaca baris per baris dengan jumlah total baris sebanyak 80 baris. Pada pengujian yang dilakukan data yang dibaca akan di tampilkan pada serial monitor Arduino. Data berhasil dibaca untuk semua jenis panjang karakter yang ada pada skenario. Pada Gambar 5.4 merupakan hasil uji coba membaca data dengan panjang 584 karakter. Pada Gambar 5.5 merupakan hasil uji coba membaca data dengan panjang 980 karakter. Pada Gambar 5.6 merupakan hasil uji coba membaca data dengan panjang 1280 karakter. Pada Gambar 5.7 merupakan hasil uji coba membaca data dengan panjang 1345 karakter. Dan pada Gambar 5.8 merupakan hasil uji coba membaca data dengan panjang 1870 karakter.

Gambar 5.4 Hasil Uji Coba UJ-F01 pada Panjang Karakter 584

Gambar 5.5 Hasil Uji Coba UJ-F01 pada Panjang Karakter 980



Gambar 5.6 Hasil Uji Coba UJ-F01 pada Panjang Karakter 1280

Gambar 5.7 Hasil Uji Coba UJ-F01 pada Panjang Karakter 1345

Gambar 5.8 Hasil Uji Coba UJ-F01 pada Panjang Karakter 1870

5.4.2 Hasil Uji Coba (UJ-F02) – Kompresi Data

Uji coba dilakukan berdasarkan skenario uji coba pada bab 5.3.2. Pada skenario dilakukan kompresi data berdasarkan konfigurasi tertentu. Konfigurasi encoder pada Heatshrink yang perlu diperhatikan adalah ukuran *window*, *lookahead* dan ukuran *input buffer*. Ukuran input buffer di setting *default* 64, sedangkan ukuran window di setting 4 sampai 9 dan ukuran lookahead disetting ukuran 3 sampai window – 1. Untuk penulisan konfigurasinya adalah Heatshrink(ukuran window, ukuran lookahead) atau biasa disingkat menjadi **HS(window_sz, lookahead_sz)**. Berikut ini adalah hasil uji coba kompresi data yang di peroleh pada Tabel 5.7.

Tabel 5.7 Hasi Uji Coba UJ – F02 Kompresi Data

	Panjang Data	**HS (4,3)**	**HS (5,4)**	**HS (6,5)**	**HS (7,6)**	**HS (8,7)**	**HS (9,8)**
Kompresi	584	berhasil	berhasil	berhasil	berhasil	berhasil	berhasil
	980	berhasil	berhasil	berhasil	berhasil	berhasil	gagal
	1280	berhasil	berhasil	berhasil	berhasil	berhasil	gagal
	1435	berhasil	berhasil	berhasil	berhasil	berhasil	gagal
	1870	berhasil	berhasil	berhasil	berhasil	gagal	gagal

Menurut hasil uji coba pada table diatas, sebagian besar proses konfigurasi menggunakan settingan konfigurasi **HS (9,8) gagal**, kecuali pada data dengan panjang 584. Kegagalan tersebut dikarenakan memori yang tidak mecukupi, sehingga tidak dapat

melakukan proses kompresi data. Selain itu pada settingan HS (8,7) proses kompresi tidak dapat berjalan pada data dengan panjang 1870 karakter.

5.4.3 Hasil Uji Coba (UJ-F03) – Komunikasi Pada Topologi Cluster Tree

Uji coba yang dilakukan pad bab 5.3.3 bertujuan untuk mengetahui fungsionalitas dari komunikasi pada topologi *cluster tree* yang digunakan. Berikut ini adalah hasil uji coba yang di peroleh berdasarkan skenario yang dijalankan yang akan diperlihatkan pada Tabel 5.8

Tabel 5.8 Hasi Uji Coba UJ – F03 Komunikasi Pada Topologi Cluster Tree

Sumber	Tujuan	Hasil
Node ZigBee *End device*	*Node* ZigBee *Router*	Berhasil
Node ZigBee *Router*	*Node* ZigBee *Coordinator*	Berhasil

Menurut hasil uji coba yang ada pada table diatas *node* ZigBee *Coordinator* berhasil menerima data yang dikirimkan oleh *node* ZigBee *End device*.

5.4.4 Hasil Uji Coba (UJ-F04) – Dekompresi Data

Uji coba dilakukan berdasarkan skenario uji coba pada bab 5.3.4. Pada skenario dilakukan dekompresi data berdasarkan konfigurasi tertentu. Pada Tabel 5.9 diperlihatkan hasil uji coba dekompresi data dengan konfigurasi **HS (window_sz, lookahead_sz)** telah ditentukan sebelumnya yaitu ukuran window adalah 4 sampai 9 dan ukuran lookahead adalah 3 sampai ukuran window – 1 dan ukuran input buffer adalah 64.

Tabel 5.9 Hasi Uji Coba UJ – F04 Dekompresi Data

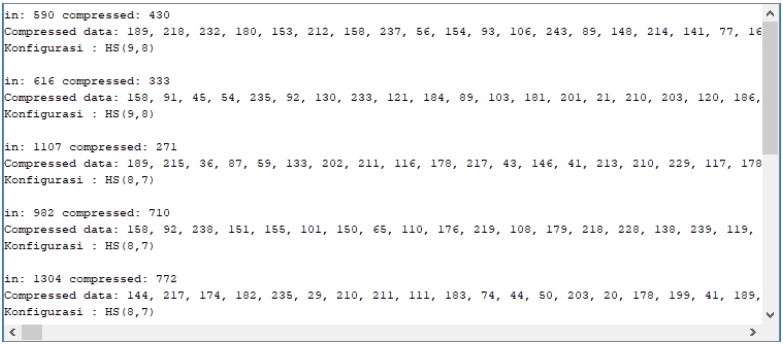
Dekompresi	Panjang Data	HS (4,3)	HS (5,4)	HS (6,5)	HS (7,6)	HS (8,7)	HS (9,8)
	584	berhasil	berhasil	berhasil	berhasil	berhasil	berhasil
	980	berhasil	berhasil	berhasil	berhasil	berhasil	gagal
	1280	berhasil	berhasil	berhasil	berhasil	berhasil	gagal
	1435	berhasil	berhasil	berhasil	berhasil	berhasil	gagal
	1870	berhasil	berhasil	berhasil	berhasil	gagal	gagal

Menurut hasil uji coba pada table diatas, sebagian besar proses konfigurasi menggunakan settingan konfigurasi **HS (9,8) gagal**, kecuali pada data dengan panjang 584 karakter. Kegagalan tersebut dikarenakan memori yang tidak mecukupi, sehingga tidak dapat melakukan proses kompresi data. Selain itu pada settingan HS (8,7) proses kompresi tidak dapat berjalan pada data dengan penjang 1640.

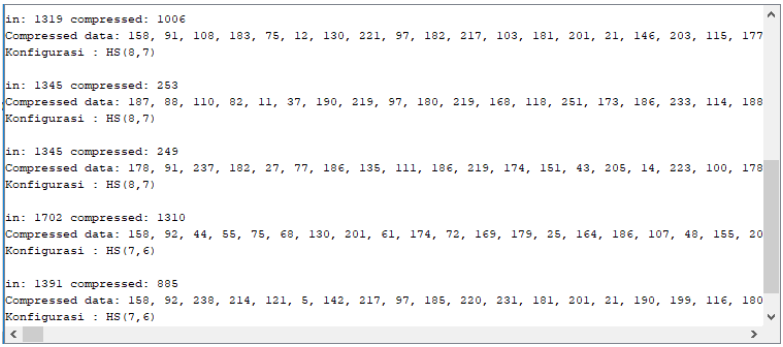
5.4.5 Hasil Uji Coba (UJ-F05) – Kompresi Adaptive

Uji coba dilakukan berdasarkan skenario uji coba pada bab 5.3.5. Pada skenario dilakukan kompresi adaptive berdasarkan konfigurasi yang telah ditetapkan. Skenario uji coba menggunakan 10 buah data string dengan panjang data yang masing-masing berbeda. Hasil pengujian akan ditampilkan pada serial monitor beserta konfigurasi encoder/decoder yang digunakan. Berikut ini

adalah hasil uji coba kompresi *adaptive* yang diperoleh ditunjukkan pada Gambar 5.9 dan Gambar 5.10.



Gambar 5.9 Hasil Uji Coba UJ-F05 Kompresi Adaptive



Gambar 5.10 Hasil Uji Coba UJ-F05 Kompresi Adaptive

5.5 Skenario Uji Coba Performa

Uji coba performa sistem dilakukan untuk mengetahui performa dari sistem yang dibangun berdasarkan skenario yang akan dibuat. Uji coba performa sistem yang akan dilakukan meliputi efektifitas hasil kompresi dalam, akurasi pengiriman, dan waktu yang dibutuhkan untuk proses kompresi serta dekompresi.

5.5.1 Skenario Uji Coba Efektifitas Kompresi

Uji coba efektifitas kompresi bertujuan untuk mengetahui efektifitas kompresi data dari berbagai konfigurasi encoder / decoder yang digunakan. Terdapat 10 buah data string beragam dengan panjang data yang sama yang akan digunakan untuk uji coba. Tabel 5.10 adalah tabel skenario uji coba efektifitas kompresi.

Tabel 5.10 Uji Coba Efektifitas Kompresi

ID	UJ – P01
Nama	Uji Coba Efektifitas Kompresi
Tujuan Uji Coba	Menguji performa sistem untuk mengetahui efektifitas kompresi yang didapatkan dari dari berbagai konfigurasi encoder / decoder yang digunakan dan berbagai panjang data yang digunakan.
Kondisi Awal	*Node* ZigBee *End device* diaktifkan
Skenario	1. *Node* ZigBee *End device* membaca data pada *sdcard* dengan panjang 584 karakter 2. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (4,3) 3. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (5,4) 4. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (6,5) 5. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (7,6)

	6. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (8,7) 7. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (9,8) 8. Mengulangi pengujian dengan 10 data string yang berbeda 9. Mengulangi langkah 1 – 8 dengan data uji coba yang memiliki panjang 980 karakter 10. Mengulangi langkah 1 – 8 dengan data uji coba yang memiliki panjang 1280 karakter 11. Mengulangi langkah 1 – 8 dengan data uji coba yang memiliki panjang 1345 karakter 12. Mengulangi langkah 1 – 8 dengan data uji coba yang memiliki panjang 1870 karakter
Masukan	Data raw yang akan dikompresi
Keluaran	Tampilan jumlah data hasil kompresi data pada serial monitor *node* ZigBee *End device*
Hasil yang Diharapkan	Tercatatnya efektifitas kompresi pada setiap skenario uji coba

5.5.2 Skenario Uji Coba Akurasi Pengiriman Data ZigBee pada Jaringan *Single Hop*

Tujuan dari skenario uji coba performa ini adalah untuk mengetahui seberapa besar akurasi sistem dalam mengirimkan paket data melalui ZigBee pada jaringan *single hop*. Uji coba dilakukan menggunakan *node* ZigBee *Router* dan *node* ZigBee *Coordinator* dengan jarak antar kedua *node* ±10 meter, ±20 meter

dan ±30 meter. Percobaan diulangi sebanyak sepuluh kali. Berikut ini menjelaskan skenario uji cobanya yang akan diperlihatkan pada Tabel 5.11.

Tabel 5.11 Uji Coba Akurasi Pengiriman Data ZigBee pada Jaringan *Single Hop*

ID	UJ – P02
Nama	Uji Coba Akurasi Pengiriman Data ZigBee pada Jaringan *Single hop*
Tujuan Uji Coba	Menguji performa sistem untuk mengetahui akurasi pengiriman data menggunakan ZigBee pada *single hop* dari jarak yang berbeda – beda (±10 meter, ±20 meter, ±30 meter)
Kondisi Awal	*Node* ZigBee *Router* dan *node* ZigBee Coordiantor diaktifkan. Keduanya berada dalam mode XBee.
Skenario	1. Menyalakan *node* ZigBee *Router* dan *node* ZigBee *Coordinator* dalam mode XBee selama kurang lebih 2 menit pada jarak ±10 meter 2. Mengulangi pengujian hingga 10 kali 3. Memantau dan mencatat hasil pada serial motor *node* ZigBee *Coordinator* 4. Mengulangi pengujian pada jarak ±20 meter 5. Mengulangi pengujian pada jarak ±30 meter
Masukan	Data yang akan dikirim
Keluaran	Akurasi pengiriman data pada jarak berbeda - beda
Hasil yang Diharapkan	Mendatpakan akurasi pengiriman data menggunakan ZigBee dari jarak

	berbeda – beda dan membandingkan hasilnya.

5.5.3 Skenario Uji Coba Akurasi Pengiriman Data ZigBee pada Jaringan *Multi Hop*

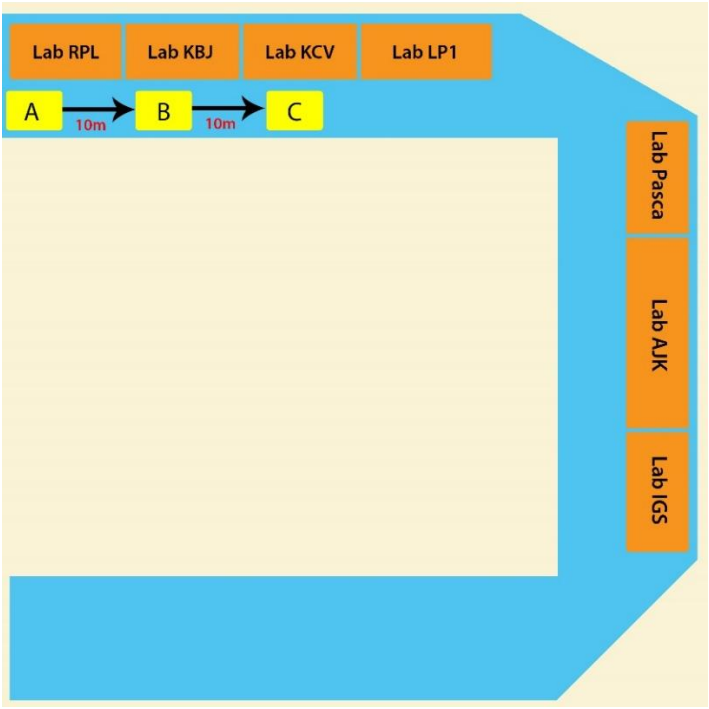
Hampir sama dengan uji coba sebelumnya, pengujian selanjutnya yang dilakukan adalah pengujian akurasi pengiriman data ZigBee pada jarigan *multi hop*. Tujuan dari uji coba ini adalah untuk mengetahui seberapa besar akurasi sistem dalam mengirimkan paket data melalui ZigBee pada jaringan *multi hop*. Uji coba dilakukan menggunakan tiga *node*, yaitu *node* ZigBee *End device*, *node* ZigBee *Router* dan *node* ZigBee *Coordinator*. Lokasi yang digunakan untuk uji coba adalah lokasi ketiga yang berada pada lantai 3 Departemen Informatika. Masing-masing *node* dipisahkan pada jarak tertentu. Panjang jarak yang digunakan untuk uji coba adalah ±10 meter, ±20 meter dan ±30 meter. Percobaan diulangi sebanyak sepuluh kali. Berikut ini Tabel 5.12 menjelaskan skenario uji cobanya.

Tabel 5.12 Uji Coba Akurasi Pengiriman Data ZigBee pada Jaringan *Multi Hop*

ID	UJ – P03
Nama	Uji Coba Akurasi Pengiriman Data ZigBee pada Jaringan *Multi Hop*
Tujuan Uji Coba	Menguji performa sistem untuk mengetahui akurasi pengiriman data menggunakan ZigBee pada *multi hop* dari jarak yang berbeda – beda (±10 meter, ±20 meter, ±30 meter)
Kondisi Awal	*Node* ZigBee *End device*, *node* ZigBee *Router* dan *node* ZigBee Coordiantor diaktifkan. Ketiga *node* berada dalam mode XBee.

Skenario	1. Menyalakan ketiga *node* dalam mode XBee selama kurang lebih 2 menit pada jarak ±10 meter 2. Mengulangi pengujian hingga 10 kali 3. Memantau dan mencatat hasil pada serial motor *node* ZigBee *Coordinator* 4. Mengulangi pengujian pada jarak ±20 meter 5. Mengulangi pengujian pada jarak ±30 meter
Masukan	Data yang akan dikirim
Keluaran	Akurasi pengiriman data pada jarak berbeda - beda
Hasil yang Diharapkan	Mendapatkan akurasi pengiriman data menggunakan ZigBee dari jarak berbeda – beda dan membandingkan hasilnya.

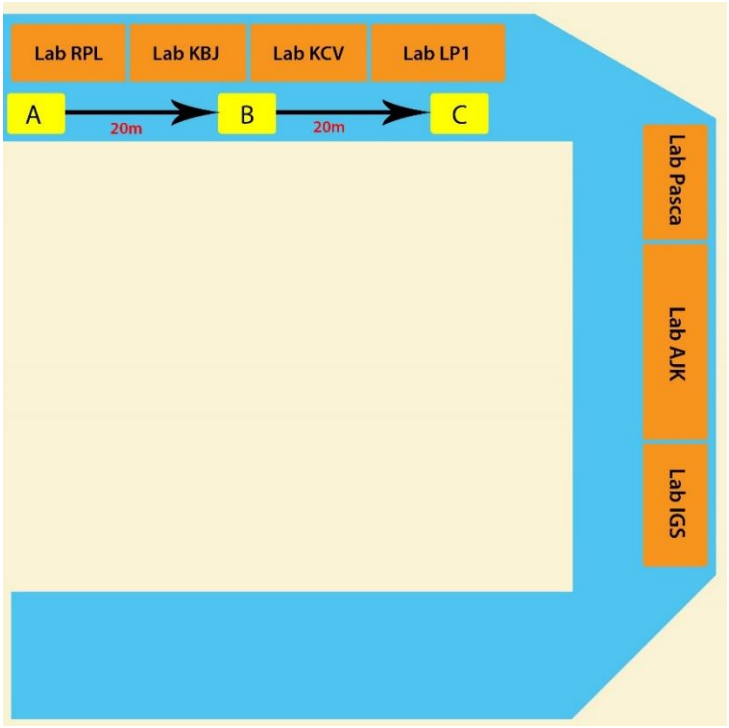
Pada uji coba pertama, jarak masing-masing *node* kurang lebih 10 meter. Berikut ini adalah peta uji coba akurasi pengiriman data ZigBee pada *multi hop* dengan jarak ±10 meter yang ditunjukkan oleh Gambar 5.11.



Gambar 5.11 Peta Lokasi Skenario Uji Coba dengan Jarak ±10 meter

Pada Gambar 5.11 terdapat tiga kotak berwarna kuning dengan nama masing-masing A, B, dan C. Kotak A menunjukkan posisi dari *node* ZigBee *Coordinator*. Kotak B menunjukkan posisi dari *node* ZigBee *Router*. Dan kotak C menunjukkan posisi dari *node* ZigBee *End device*.

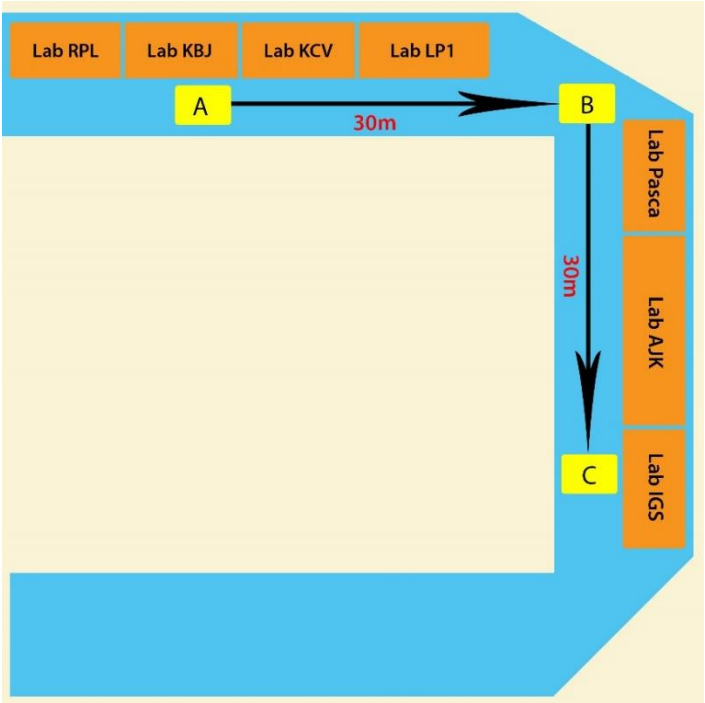
Pada uji coba kedua, jarak masing-masing *node* kurang lebih 20 meter. Berikut ini adalah peta uji coba akurasi pengiriman data ZigBee pada *multi hop* dengan jarak ±20 meter yang ditunjukkan oleh Gambar 5.12.



Gambar 5.12 Peta Lokasi Skenario Uji Coba dengan Jarak ±20 meter

Pada Gambar 5.12 terdapat tiga kotak berwarna kuning dengan nama masing-masing A, B, dan C. Kotak A menunjukkan posisi dari *node* ZigBee *Coordinator*. Kotak B menunjukkan posisi dari *node* ZigBee *Router*. Dan kotak C menunjukkan posisi dari *node* ZigBee *End device*.

Pada uji coba ketiga, jarak masing-masing *node* kurang lebih 30 meter. Berikut ini adalah peta uji coba akurasi pengiriman data ZigBee pada *multi hop* dengan jarak ±30 meter yang ditunjukkan oleh Gambar 5.13.



Gambar 5.13 Peta Lokasi Skenario Uji Coba dengan Jarak ±30 meter

Pada Gambar 5.13 terdapat tiga kotak berwarna kuning dengan nama masing-masing A, B, dan C. Kotak A menunjukkan posisi dari *node* ZigBee *Coordinator*. Kotak B menunjukkan posisi dari *node* ZigBee *Router*. Dan kotak C menunjukkan posisi dari *node* ZigBee *End device*.

5.5.4 Skenario Uji Waktu Kompresi

Uji coba waktu kompresi, bertujuan untuk mengetahui waktu yang dibutuhkan untuk proses kompresi data. Terdapat 10 buah data string beragam dengan panjang data yang sama yang akan digunakan untuk uji coba. Tabel 5.13 akan menjelaskan

skenario uji coba waktu yang dibutuhkan untuk kompresi data secara detail.

Tabel 5.13 Uji Coba Waktu Kompresi

ID	UJ – P04
Nama	Uji Coba Waktu Kompresi
Tujuan Uji Coba	Menguji performa sistem untuk mengetahui waktu yang dibutuhkan untuk proses kompresi
Kondisi Awal	Data raw telah di siapkan pada file tersendiri. *Node* ZigBee *End device* diaktifkan.
Skenario	1. *Node* ZigBee *End device* membaca data pada *sdcard* dengan panjang 584 karakter 2. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (4,3) 3. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (5,4) 4. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (6,5) 5. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (7,6) 6. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (8,7) 7. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (9,8) 8. Mengulangi pengujian dengan 10 data string yang berbeda

	9. Mengulangi langkah 1 – 8 dengan data uji coba yang memiliki panjang 980 karakter 10. Mengulangi langkah 1 – 8 dengan data uji coba yang memiliki panjang 1280 karakter 11. Mengulangi langkah 1 – 8 dengan data uji coba yang memiliki panjang 1345 karakter 12. Mengulangi langkah 1 – 8 dengan data uji coba yang memiliki panjang 1870 karakter
Masukan	Data raw yang akan dikompresi
Keluaran	Tampilan waktu yang dibutuhkan untuk proses kompresi pada serial monitor *node* ZigBee *End device*
Hasil yang Diharapkan	Tercatatnya waktu yang di butuhkan untuk proses kompresi data pada setiap skenario uji coba

5.5.5 Skenario Uji Waktu Dekompresi

Uji coba waktu dekompresi, bertujuan untuk mengetahui waktu yang dibutuhkan proses dekompresi data. Terdapat 10 buah data string beragam dari hasil kompresi yang telah dilakukan sebelumnya. Data tersebut akan digunakan untuk skenario uji coba. Tabel 5.14 akan menjelaskan skenario uji coba waktu yang dibutuhkan untuk dekompresi data secara detail.

Tabel 5.14 Uji Coba Waktu Dekompresi

ID	UJ – P05
Nama	Uji Coba Waktu Dekompresi
Tujuan Uji Coba	Menguji performa sistem untuk mengetahui waktu yang dibutuhkan untuk proses dekompresi

Kondisi Awal	Data hasil kompresi telah di siapkan. *Node* ZigBee *End device* diaktifkan.
Skenario	1. Menyiapkan data hasil kompresi dari panjang data 584 karakter sebelumnya 2. *Node* ZigBee melakukan dekompresi data dengan konfigurasi HS (4,3) 3. *Node* ZigBee melakukan dekompresi data dengan konfigurasi HS (5,4) 4. *Node* ZigBee melakukan dekompresi data dengan konfigurasi HS (6,5) 5. *Node* ZigBee melakukan dekompresi data dengan konfigurasi HS (7,6) 6. *Node* ZigBee melakukan dekompresi data dengan konfigurasi HS (8,7) 7. *Node* ZigBee melakukan kompresi data dengan konfigurasi HS (9,8) 8. Mengulangi pengujian dengan 10 data hasil kompresi sebelumnya 9. Mengulangi langkah 1 – 8 dengan hasil kompresi data dari 980 karakter 10. Mengulangi langkah 1 – 8 dengan hasil kompresi data dari 1280 karakter 11. Mengulangi langkah 1 – 8 dengan hasil kompresi data dari 1345 karakter

	12. Mengulangi langkah 1 – 8 dengan hasil kompresi data dari 1870 karakter
Masukan	Data hasil kompresi
Keluaran	Tampilan waktu yang dibutuhkan untuk proses dekompresi pada serial monitor *node* ZigBee *End device*
Hasil yang Diharapkan	Tercatatnya waktu yang di butuhkan untuk proses dekompresi data pada setiap skenario uji coba

5.6 Hasil Uji Coba Performa

Telah dijabarkan pada bagian sebelumnya mengenai skenario dari keseluruhan uji coba performa. Skenario uji coba yang telah dijabarkan terdiri dari beberapa hal. Secara garis besar, pengujian performa terdiri dari tiga bagian, yaitu performa efektifitas, akurasi pengiriman, dan waktu.

5.6.1 Hasil Uji Coba (UJ-P01) – Efektifitas Kompresi

Sesuai skenario pada bab 5.5.1, yang di ujikan adalah efektifitas kompresi data dari berbagai konfigurasi *encoder / decoder* yang digunakan dengan 10 buah data string yang berbeda -beda. Untuk mengetahui nilai efektifitas digunakan rumus :

$$\text{Efektifitas} = \left(1 - \frac{panjang\ data\ setelah\ kompresi}{panjang\ data\ sebelum\ kompresi}\right) \times 100\%$$

Berikut ini adalah hasil yang diperoleh dari uji coba efektifitas kompresi dengan mengambil rata – rata dari 10 kali percobaan untuk setiap konfigurasi dan panjang data, yang ditunjukkan pada Tabel 5.15.

Tabel 5.15 Hasil Uji Coba UJ – P01 Efektifitas Kompresi

Panjang Data	Uji Coba	Konfigurasi					
		HS (4,3)	**HS (5,4)**	**HS (6,5)**	**HS (7,6)**	**HS (8,7)**	**HS (9,8)**
584	Efektifitas	9.3 %	24.4 %	31.8 %	49.4 %	59.8 %	56.7 %
980	Efektifitas	9.6 %	19.9 %	32.3 %	49.1 %	54.5 %	60.5 %
1280	Efektifitas	10.0 %	16.8 %	37.4 %	45.9 %	50.2 %	gagal
1345	Efektifitas	10.2 %	17.0 %	37.7 %	45.9 %	54.5 %	gagal
1870	Efektifitas	5.1 %	11.2 %	20.4 %	27.8 %	gagal	gagal

Berdasarkan hasil uji coba efektifitas kompresi pada panjang data **584 karakter,** didapatkan persentase efektifitas kompresi paling kecil ketika menggunakan konfigurasi HS (4,3) sebesar **9.3%**. Sedangkan efektifitas kompresi paling besar adalah **56.7%** pada konfigurasi **HS (9,8)**. Tabel 5.15 menjabarkan hasil uji coba dari efektifitas kompresi pada panjang data **584 karakter** yang telah dijabarkan lebih rinci pada Table 8.1 di lampiran.

Berdasarkan hasil uji coba efektifitas kompresi pada panjang data **980 karakter,** didapatkan persentase efektifitas kompresi paling kecil ketika menggunakan konfigurasi HS(4,3) sebesar **9.6%**. Sedangkan efektifitas kompresi paling besar adalah **60.5%** pada konfigurasi **HS (9,8)**. Tabel 5.15 menjabarkan hasil uji coba dari efektifitas kompresi pada panjang data **980 karakter** yang telah dijabarkan lebih rinci pada Table 8.2 di lampiran.

Berdasarkan hasil uji coba efektifitas kompresi pada panjang data **1280 karakter,** didapatkan persentase efektifitas kompresi paling kecil ketika menggunakan konfigurasi HS (4,3) sebesar **10.0%**. Sedangkan efektifitas kompresi paling besar adalah **50.2%** pada konfigurasi **HS (8,7**). Tabel 5.15 menjabarkan hasil uji coba dari efektifitas kompresi pada panjang data **1280**

karakter yang telah dijabarkan lebih rinci pada Table 8.3 di lampiran.

Berdasarkan hasil uji coba efektifitas kompresi pada panjang data **1345 karakter,** didapatkan persentase efektifitas kompresi paling kecil ketika menggunakan konfigurasi HS (4,3) sebesar **10.2%**. Sedangkan efektifitas kompresi paling besar adalah **54.5%** pada konfigurasi **HS (8,7)**. Tabel 5.15 menjabarkan hasil uji coba dari efektifitas kompresi pada panjang data **1345 karakter** yang telah dijabarkan lebih rinci pada Table 8.4 di lampiran.

Berdasarkan hasil uji coba efektifitas kompresi pada panjang data **1870 karakter,** didapatkan persentase efektifitas kompresi paling kecil ketika menggunakan konfigurasi HS (4,3) sebesar **5.1%**. Sedangkan efektifitas kompresi paling besar adalah **27.8%** pada konfigurasi **HS (8,7)**. Tabel 5.15 menjabarkan hasil uji coba dari efektifitas kompresi pada panjang data **1870 karakter** yang telah dijabarkan lebih rinci pada Table 8.5 di lampiran.

Pada konfigurasi **HS (9,8)** untuk data dengan panjang lebih dari 1219 karakter tidak dapat dilakukan. Selain itu, pada konfigurasi **HS (8,7)** tidak dapat berjalan pada data dengan panjang lebih dari 1869 karakter, keadaan tersebut disebabkan karena keterbatasan kapasitas memori yang ada pada Arduino.

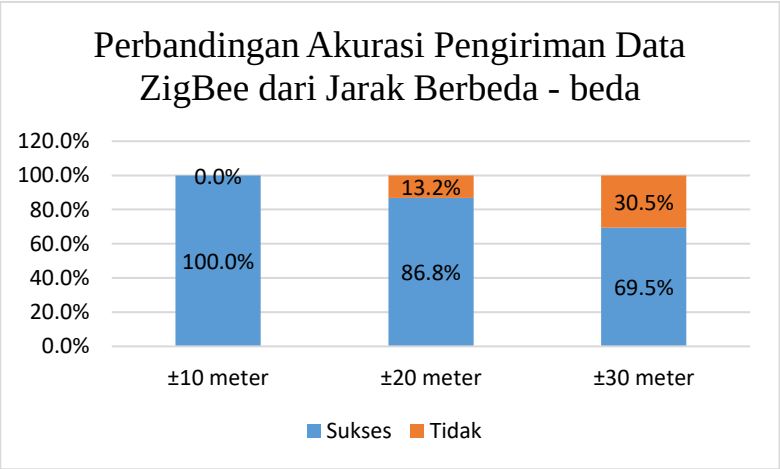
5.6.2 Hasil Uji Coba (UJ-P02) – Akurasi Pengiriman Data ZigBee pada Jaringan *Single Hop*

Mengacu pada skenario bab 5.5.2, dilakukan pengujian untuk mengetahui akurasi pengiriman data pada ZigBee. Setiap data yang diterima akan di print pada serial monitor *node* ZigBee *Coordinator*. Berikut ini Tabel 5.16 menjelaskan hasil uji coba akurasi pengiriman data yang didapatkan pada jaringan single hop menggunakan protokol ZigBee.

Tabel 5.16 Hasil Uji Coba UJ – P02 Pengiriman Data ZigBee pada Jaringan *Single Hop*

Akurasi Pengiriman ZigBee		
±10 meter	**±20 meter**	**±30 meter**
100%	86,8%	69,5%

Berdasarkan hasil uji coba akurasi pada jarak **±10 meter,** didapatkan akurasi pengiriman data menggunakan ZigBee sebesar **100%.** Untuk detail lebih rincinya dapat dilihat pada Table 8.6 di lampiran. Pada jarak **±20 meter** akurasi yang didapatkan sebesar **86,8%** dan untuk detail lebih rincinya dapat dilihat pada Table 8.7 di lampiran. Pada jarak **±30 meter** akurasi yang didapatkan sebesar **69,5%** dan untuk detail lebih rincinya dapat dilihat pada Table 8.8 di lampiran. Pada Gambar 5.14 akan di sajikan grafik perbandingan akurasi pengiriman dari jarak berbeda – beda.



Gambar 5.14 Grafik Perbandingan Akurasi Pengiriman Data ZigBee pada Jaringan *Single Hop*

Berdasarkan grafik diatas, dapat diketahui bahwa semakin jauh jarak komunikasi maka akurasi pengiriman data ZigBee akan semakin menurun.

5.6.3 Hasil Uji Coba (UJ-P03) – Akurasi Pengiriman Data ZigBee pada Jaringan *Multi Hop*

Mengacu pada skenario bab 5.5.3, dilakukan pengujian untuk mengetahui akurasi pengiriman data pada ZigBee. Setiap data yang diterima akan di print pada serial monitor *node* ZigBee *Router* dan monitor *node* ZigBee Coordiantor. Berikut ini tabel 5.17 menjelaskan hasil uji coba yang didapatkan

Tabel 5.17 Hasil Uji Coba UJ – P03 Akurasi Pengiriman Data ZigBee pada Jaringan *Multi Hop*

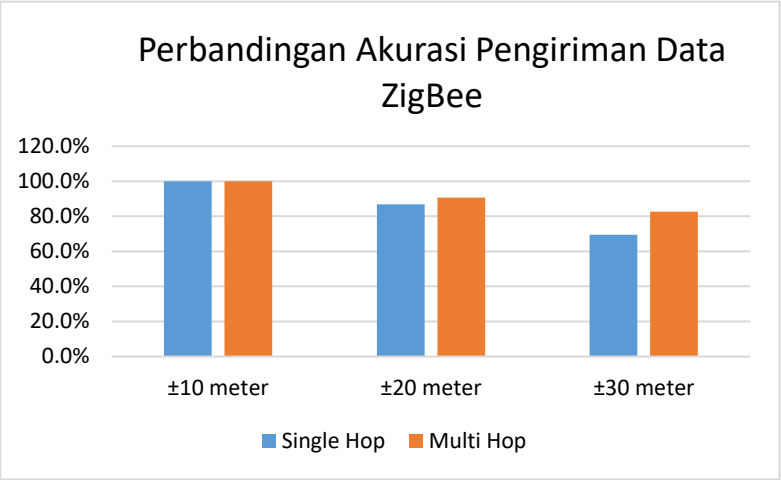
Akurasi Pengiriman ZigBee		
±10 meter	**±20 meter**	**±30 meter**
100%	90,6%	82,6 %

Berdasarkan hasil uji coba akurasi pada jarak **±10 meter,** didapatkan akurasi pengiriman data menggunakan ZigBee sebesar **100%.** Tabel 5.17 menjabarkan hasil uji coba dari jarak **±10 meter** yang telah dijabarkan lebih rinci pada Table 8.9 di lampiran

Berdasarkan hasil uji coba akurasi pada jarak **±20 meter,** didapatkan akurasi pengiriman data menggunakan ZigBee sebesar **90,6%.** Tabel 5.17 menjabarkan hasil uji coba dari jarak **±20 meter** yang telah dijabarkan lebih rinci pada Table 8.10 di lampiran

Berdasarkan hasil uji coba akurasi pada jarak **±30 meter,** didapatkan akurasi pengiriman data menggunakan ZigBee sebesar **82,6%.** Tabel 5.17 menjabarkan hasil uji coba dari jarak **±30 meter** yang telah dijabarkan lebih rinci pada Table 8.11 di lampiran

Untuk lebih rincinya, berikut ini adalah perbandingan data akurasi pengiriman data ZigBee pada jarak berbeda - beda.



Gambar 5.15 Grafik Perbandingan Akurasi Pengiriman Data ZigBee pada Jaringan *Single Hop* dengan Jaringan *Multi Hop*

Lokasi uji coba akurasi pengiriman data ZigBee pada jaringan *single hop* dilakukan di lokasi kedua yang berada di lapangan Departemen Informatika ITS. Tempat tersebut merupakan ruangan terbuka dengan terdapat banyak meja kursi dan tiang – tiang di area tersebut beserta tanaman disekitarnya. Sedangkan lokasi uji coba akurasi pengiriman data ZigBee pada jaringan *multi hop* dilakukan di lokasi ketiga yang berada di lantai 3 gedung Departemen Informatika ITS. Tempat tersebut merupakan koridor semi terbuka.

Berdasarkan grafik diatas, terdapat perbedaan akurasi yang didapatkan. Pada jaringan *single hop* dan multihop untuk jarak uji coba ±10 meter akurasi yang didapatkan sama-sama **100%**. Kemudian pada jarak ±20 meter akurasi jaringan *single hop* menurun menjadi **86,8%** sedangkan akurasi jaringan *multi hop*

adalah **90,6%.** Selanjutnya pada jarak ±30 meter akurasi jaringan *single hop* menurun menjadi **69,5%** sedangkan akurasi jaringan *multi hop* adalah **82,6%.** Dari hasil perbandingan diatas dapat disimpulkan bahwa jarak komunikasi dan lokasi tempat uji coba mempengaruhi penurunan akurasi pengiriman. Semakin jauh jarak jarak komunikasi maka akurasi semakin menurun. Selain itu, semakin banyak penghalang antara pemancar sinyal dengan penerima maka akurasinya akan semakin menurun.

5.6.4 Hasil Uji Coba (UJ-P04) – Waktu Kompresi Data

Pada skenario bab 5.5.4 dijabarkan tentang uji coba waktu yang dibutuhkan dalam melakukan proses kompresi data. Pengujian dilakukan pada 10 buah data string yang berbeda - beda untuk masing – masing panjang data pada skenario. Berikut ini adalah hasil uji coba yang didapatkan yang dapat dilihat pada tabel 5.18.

Tabel 5.18 Hasil Uji Coba UJ – P04 Waktu Kompresi Data

Panjang Data	Uji Coba	Konfigurasi					
		HS (4,3)	HS (5,4)	HS (6,5)	HS (7,6)	HS (8,7)	HS (9,8)
584	Waktu	0.05804	0.04692	0.04142	0.03231	0.02948	0.03005
980	Waktu	0.09688	0.08148	0.0688	0.05472	0.0529	0.0499
1280	Waktu	0.12733	0.11091	0.0877	0.07781	0.0784	gagal
1345	Waktu	0.13365	0.11642	0.09174	0.08168	0.08238	gagal
1870	Waktu	0.19124	0.16759	0.14966	0.14012	gagal	gagal

Berdasarkan hasil uji coba waktu kompresi data pada panjang data **584 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.02948 detik** ketika menggunakan konfigurasi **HS (8,7).** Tabel 5.18 menjabarkan hasil uji coba dari waktu kompresi pada panjang data **584 karakter** yang telah dijabarkan lebih rinci pada Table 8.12 di lampiran.

Berdasarkan hasil uji coba waktu kompresi data pada panjang data **980 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.0499 detik** ketika menggunakan konfigurasi **HS (9,8).** Tabel 5.18 menjabarkan hasil uji coba dari waktu kompresi pada panjang data **980 karakter** yang telah dijabarkan lebih rinci pada Table 8.13 di lampiran.

Berdasarkan hasil uji coba waktu kompresi data pada panjang data **1280 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.07781 detik** ketika menggunakan konfigurasi **HS (7,6**). Tabel 5.18 menjabarkan hasil uji coba dari waktu kompresi pada panjang data **1280 karakter** yang telah dijabarkan lebih rinci pada Table 8.14 di lampiran.

Berdasarkan hasil uji coba waktu kompresi data pada panjang data **1345 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.08168 detik** ketika menggunakan konfigurasi **HS (7,6)**. Tabel 5.18 menjabarkan hasil uji coba dari waktu kompresi pada panjang data **1345 karakter** yang telah dijabarkan lebih rinci pada Table 8.15 di lampiran.

Berdasarkan hasil uji coba waktu kompresi data pada panjang data **1870 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.14012 detik** ketika menggunakan konfigurasi **HS (7,6)**. Tabel 5.18 menjabarkan hasil uji coba dari waktu kompresi pada panjang data **1345 karakter** yang telah dijabarkan lebih rinci pada Table 8.16 di lampiran.

Pada konfigurasi **HS (9,8)** untuk data dengan panjang lebih dari 1219 karakter tidak dapat dilakukan. Selain itu, pada konfigurasi **HS (8,7)** tidak dapat berjalan pada data dengan panjang lebih dari 1869 karakter, keadaan tersebut disebabkan karena keterbatasan kapasitas memori yang ada pada Arduino.

.

5.6.5 Hasil Uji Coba (UJ-P05) – Waktu Dekompresi Data

Pada skenario bab 5.5.5 dijabarkan tentang uji coba waktu yang dibutuhkan dalam melakukan proses dekompresi data. Pengujian dilakukan pada 10 buah data string yang berbeda - beda dari hasil kompresi data sebelumnya. Berikut ini adalah hasil uji coba waktu dekompresi data yang didapatkan pada Tabel 5.19.

Tabel 5.19 Hasil Uji Coba UJ – P05 Waktu Dekompresi Data

Panjang Data	Uji Coba	Konfigurasi					
		HS (4,3)	HS (5,4)	HS (6,5)	HS (7,6)	HS (8,7)	HS (9,8)
584	Waktu	0.020436	0.018062	0.016648	0.013286	0.011669	0.011856
980	Waktu	0.033833	0.031182	0.027464	0.022094	0.019 83	0.018288
1280	Waktu	0.044415	0.042452	0.034629	0.030599	0.027731	gagal
1345	Waktu	0.046591	0.044518	0.036244	0.032141	0.028978	gagal
1870	Waktu	0.066589	0.064135	0.059676	0.055481	gagal	gagal

Berdasarkan hasil uji coba waktu dekompresi data pada panjang data **584 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.011669 detik** ketika menggunakan konfigurasi **HS (8,7).** Tabel 5.19 menjabarkan hasil uji coba dari waktu kompresi pada panjang data **584 karakter** yang telah dijabarkan lebih rinci pada Table 8.17 di lampiran.

Berdasarkan hasil uji coba waktu kompresi data pada panjang data **980 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.018288 detik** ketika menggunakan konfigurasi **HS (9,8).** Tabel 5.19 menjabarkan hasil uji coba dari

waktu kompresi pada panjang data **980 karakter** yang telah dijabarkan lebih rinci pada Table 8.18 di lampiran.

Berdasarkan hasil uji coba waktu kompresi data pada panjang data **1280 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.027731 detik** ketika menggunakan konfigurasi **HS (8,7**). Tabel 5.19 menjabarkan hasil uji coba dari waktu kompresi pada panjang data **1280 karakter** yang telah dijabarkan lebih rinci pada Table 8.19 di lampiran.

Berdasarkan hasil uji coba waktu kompresi data pada panjang data **1345 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.028978 detik** ketika menggunakan konfigurasi **HS (8,7)**. Tabel 5.19 menjabarkan hasil uji coba dari waktu kompresi pada panjang data **1345 karakter** yang telah dijabarkan lebih rinci pada Table 8.20 di lampiran.

Berdasarkan hasil uji coba waktu kompresi data pada panjang data **1870 karakter,** waktu tercepat yang dibutuhkan untuk kompresi data adalah **0.055481 detik** ketika menggunakan konfigurasi **HS (8,7)**. Tabel 5.19 menjabarkan hasil uji coba dari waktu kompresi pada panjang data **1870 karakter** yang telah dijabarkan lebih rinci pada Table 8.21 di lampiran.

Pada konfigurasi **HS (9,8)** untuk data dengan panjang lebih dari 1219 karakter tidak dapat dilakukan. Selain itu, pada konfigurasi **HS (8,7)** tidak dapat berjalan pada data dengan panjang lebih dari 1869 karakter, keadaan tersebut disebabkan karena keterbatasan kapasitas memori yang ada pada Arduino.

5.7 Evaluasi Hasil Uji Coba

Pada bagian sebelumnya, telah dilakukan uji coba terhadap sistem yang dibuat. Uji coba yang telah dilakukan berkenaan dengan uji coba fungsionalitas dan uji coba performa. Berikut ini Tabel 5.20 merangkum evaluasi hasil uji coba fungsionalitas dan Tabel 5.21 merangkum evaluasi hasil uji coba performa yang telah dilakukan.

Tabel 5.20 Evaluasi Hasil Uji Coba Fungsionalitas

No	Kode Uji Coba	Evaluasi
1	UJ – F01	Dapat membaca data dari sd card berdasarkan skenario yang ada
2	UJ – F02	Sebagian besar data berhasil di kompresi, akan tetapi ketika menggunakan konfigurasi HS(9,8) pada panjang data lebih dari 1219 karakter kompresi data mengalami kegagalan dikarenakan keterbatasan memori yang ada pada Arduino.
3	UJ – F03	Komunikasi pada topologi cluster tree berjalan dengan baik, hal ini dapat dilihat dari *node* ZigBee *Coordinator* dapat menerima data yang dikirimkan oleh *node* ZigBee *End device*
4	UJ – F04	Sebagian besar data berhasil di dekompresi, akan tetapi ketika menggunakan konfigurasi HS (9,8) pada panjang data lebih dari 1219 karakter kompresi data mengalami kegagalan dikarenakan keterbatasan memori yang ada pada Arduino.
5	UJ – F05	Kompresi adaptive berhasil dilakukan dengan konfigurasi encoder/decoder yang telah ditetapkan.

Tabel 5.21 Evaluasi Hasil Uji Coba Performa

No	Kode Uji Coba	Evaluasi
1	UJ – P01	Persentase efektifitas kompresi yang didapatkan semakin besar seiring dengan dengan besarnya konfigurasi window_sz dan lookahead_sz yang digunakan.
2	UJ – P02	Akurasi pengiriman data ZigBee pada jaringan *single hop* berturut – turut adalah 100% (±10 meter), 86,8% (±20 meter), dan 69,5% (±30 meter). Sehingga dapat disimpulkan, bahwa akurasi ZigBee menurun seiring dengan jarak yang semakin meningkat.
3	UJ – P03	Akurasi pengiriman data ZigBee berturut – turut pada jaringan *multi hop* adalah 100% (±10 meter), 90,6% (±20 meter), dan 82,6% (±30 meter). Sehingga dapat disimpulkan, bahwa akurasi ZigBee menurun seiring dengan jarak yang semakin meningkat.
4	UJ – P04	Rata – rata waktu yang dibutuhkan untuk proses kompresi data semakin cepat seiring dengan besarnya konfigurasi window_sz dan lookahead_sz yang digunakan.
5	UJ – P05	Rata – rata waktu yang dibutuhkan untuk proses dekompresi data semakin cepat seiring dengan besarnya konfigurasi window_sz dan lookahead_sz yang digunakan.

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak selanjutnya.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba kompresi adaptive menggunakan metode Heatshrink untuk pengiriman data pada *wireless sensor network* berbasis ZigBee adalah sebagai berikut:

1. Implementasi kompresi adaptive menggunakan metode Heatshrink dapat digunakan untuk menggurangi pengiriman data sehingga dapat menghemat penggunaan daya pada *wireless sensor network.*
2. Pada bebeberapa kondisi kompresi data tidak dapat digunakan karena keterbatasan sumberdaya yang ada pada *wireless sensor network.*
3. Performa dari sistem yang telah dibangun adalah sebagai berikut:
    a. Persentase efektifitas kompresi paling tinggi adalah **60.5%** pada panjang data 980 karakter dengan menggunakan konfigurasi HS (9,8). Persentasi efektifitas kompresi pada masing-masing data sesuai skenario uji coba rata-rata akan terus meningkat seiring dengan besar konfigurasi yang digunakan.
    b. Akurasi pengiriman data ZigBee pada jaringan *single hop* berturut – turut adalah **100%** (±10 meter), **86,8%** (±20 meter), dan **69,5%** (±30 meter).
    c. Akurasi pengiriman data ZigBee pada jaringan *multi hop* berturut – turut adalah **100%** (±10 meter), **90,6%** (±20 meter), dan **82,6%** (±30 meter).

Sehingga dapat disimpulkan, bahwa akurasi pengiriman data pada ZigBee menurun seiring dengan jarak yang semakin meningkat.

d. Waktu yang dibutuhkan untuk kompresi data paling cepat adalah **0.02948 detik** pada panjang data 584 karakter dengan menggunakan konfigurasi HS (8,7). Waktu kompresi pada masing-masing data sesuai skenario uji coba rata-rata akan mengalami percepatan seiring dengan besar konfigurasi yang digunakan.
e. Waktu yang dibutuhkan untuk dekompresi data paling cepat adalah **0.011856 detik** pada panjang data 584 karakter dengan menggunakan konfigurasi HS (9,8). Waktu dekompresi pada masing-masing data sesuai skenario uji coba rata-rata akan mengalami percepatan seiring dengan besar konfigurasi yang digunakan.

6.2 Saran

Saran yang diberikan terkait pengembangan pada Tugas Akhir ini adalah:

1. Menambah jumlah data untuk uji coba berdasarkan panjang datanya.
2. Menambahkan jumlah *node* pengirim untuk uji coba

DAFTAR PUSTAKA

[1] D. P. Harrop dan R. Das, Wireless Sensor Networks (WSN) 2014-2024: Forecasts, Technologies, Players, IDTechEx, 2014.

[2] S. Rhee, D. Seetharam dan S. Liu, "Techniques for Minimizing Power Consumption in Low Data-Rate Wireless Sensor Networks," *IEEE Wireless Communications and Networking Conference,* vol. 3, pp. 1727-1731, 2004.

[3] R. Rajagopalan dan P. K. Varshney, "Data-Aggregation Techniques in Sensor Networks: A survey," *IEEE Communications Surveys & Tutorials,* vol. 8, no. 4, pp. 48-63, 2006.

[4] C. Wang, H. Ma, Y. He dan X. Shuguang, "Adaptive Approximate Data Collection for Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Sistems,* vol. 23, no. 6, pp. 1004 - 1016, 2012.

[5] P. Ghaffariyan, "An Effective Data Aggregation Mechanism for Wireless Sensor Networks," *2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM),* pp. 1-4, 2010.

[6] "Data Aggregation and Data Fusion Techniques in WSN/SANET Topologies - A Critical Discussion," *TENCON 2012 IEEE Region 10 Conference,* pp. 1-6, 2012.

[7] Y. F. Solahuddin dan W. Ismail, "Data Fusion for Reducing Power Consumption in Arduino-Xbee Wireless Sensor Network Platform," *2014 International Conference on Computer and Information Sciences (ICCOINS),* pp. 1-6, 2014.

[8] S. Yinbiao dan K. Lee, "Internet of Things: Wireless Sensor Network," *International Electrotechnical Commission, White Paper*, 2014.

[9] B. Arne, "New Generation Sensor Web Enablement," PubMed Central (PMC), 1 March 2011. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3231615/. [Diakses 2 January 2018].

[10] Wikipedia, "Arduino," [Online]. Available: https://id.wikipedia.org/wiki/Arduino. [Diakses 2 January 2018].

[11] "Arduino Software (IDE)," [Online]. Available: https://www.arduino.cc/en/Guide/Environment. [Diakses 3 January 2018].

[12] belajararduino.net, "Shield Arduino," 2018. [Online]. Available: http://www.belajararduino.net/shield-arduino. [Diakses 3 January 2018].

[13] "Zigbee Networking with XBee Series 2 and Seeed's Products," [Online]. Available: http://wiki.seeed.cc/Zigbee_Networking_with_XBee_Series. [Diakses 3 January 2018].

[14] J. Song dan Y. K. Tan, "Energy Consumption Analysis of ZigBee-Based Energy Harvesting Wireless Sensor Networks," *2012 IEEE International Conference on Communication Sistems (ICCS)*, pp. 468-472, 2012.

[15] S. Vokes, "Heatshrink: An Embedded Data Compression Library," 14 March 2013. [Online]. Available: https://spin.atomicobject.com/2013/03/14/heatshrink-embedded-data-compression/. [Diakses 3 January 2018].

[16] Mahfud, “Algoritma Kompresi LZSS,” 14 June 2012. [Online]. Available: https://mahfudharun.wordpress.com/2012/06/14/algoritma-kompresi-lzss/. [Diakses 6 January 2018].

[17] A. Ozsoy dan M. Swany, “CULZSS: LZSS Lossless Data Compression on CUDA,” *2011 IEEE International Conference on Cluster Computing*, pp. 403-411, 2011.

[18] “Lempel–Ziv–Storer–Szymanski,” [Online]. Available: https://en.wikipedia.org/wiki/Lempel–Ziv–Storer–Szymanski. [Diakses 6 January 2018].

[19] atomicobject, “Heatshrink,” [Online]. Available: https://github.com/atomicobject/heatshrink. [Diakses 10 May 2018].

(Halaman ini sengaja dikosongkan)

LAMPIRAN

Tabel 8.1 Rincian Hasil Uji Coba UJ-P01 pada Panjang Data 584 karakter

Efektifitas Kompresi Pada Panjang Data 584		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	87.0 %	2.7 %	-4.6 %	-10.4 %	-2.2 %	1.7 %	-1.4 %	2.4 %	-0.2 %	17.6 %	9.3%
	HS (5,4)	91.6 %	7.0 %	9.8 %	-1.0 %	4.3 %	23.8 %	16.3 %	58.9 %	4.3 %	29.3 %	24.4%
	HS (6,5)	94.7 %	10.3 %	24.1 %	5.1 %	4.8 %	37.5 %	15.2 %	75.7 %	12.2 %	38.9 %	31.8%
	HS (7,6)	96.6 %	9.4 %	29.3 %	7.7 %	12.0 %	76.7 %	67.5 %	80.0 %	72.9 %	41.8 %	49.4%
	HS (8,7)	97.8 %	23.5 %	26.7 %	44.2 %	38.5 %	76.0 %	66.8 %	79.5 %	71.7 %	73.8 %	59.8%
	HS (9,8)	98.5 %	26.4 %	26.9 %	43.3 %	42.1 %	75.7 %	66.1 %	78.8 %	71.1 %	38.7 %	56.7%

Tabel 8.2 Rincian Hasil Uji Coba UJ-P01 pada Panjang Data 980 karakter

Efektifitas Kompresi Pada Panjang Data 980		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	87.1 %	3.5 %	5.6 %	0.8 %	-3.1 %	-1.5 %	10.5 %	-3.8 %	-1.0 %	-2.2 %	9.6%
	HS (5,4)	91.8 %	6.1 %	10.9 %	24.0 %	13.0 %	18.3 %	10.0 %	0.9 %	11.7 %	12.4 %	19.9%
	HS (6,5)	95.0 %	8.0 %	19.9 %	38.6 %	17.8 %	50.1 %	9.0 %	26.9 %	29.0 %	29.3 %	32.3%
	HS (7,6)	96.8 %	7.6 %	31.6 %	74.1 %	18.3 %	73.0 %	67.3 %	49.1 %	35.5 %	37.7 %	49.1%
	HS (8,7)	98.1 %	40.9 %	39.1 %	73.1 %	27.8 %	77.9 %	68.0 %	47.3 %	35.7 %	37.2 %	54.5%
	HS (9,8)	98.8 %	43.1 %	45.5 %	72.7 %	64.5 %	88.2 %	68.0 %	48.0 %	37.3 %	39.1 %	60.5%

Tabel 8.3 Rincian Hasil Uji Coba UJ-P01 pada Panjang Data 1280 karakter

Efektifitas Kompresi Pada Panjang Data 1280		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	87.3 %	3.8 %	-5.1 %	1.3 %	-3.6 %	-6.6 %	13.8 %	-5.1 %	-5.5 %	20.1 %	10.0%

	HS (5,4)	92.0 %	5.9 %	-0.3 %	19.3 %	3.7 %	0.2 %	20.5 %	0.4 %	-1.5 %	28.1 %	16.8%
	HS (6,5)	95.1 %	7.0 %	18.8 %	27.3 %	13.0 %	24.5 %	24.1 %	65.3 %	67.6 %	31.4 %	37.4%
	HS (7,6)	97.0 %	10.4 %	43.2 %	40.1 %	21.0 %	24.7 %	24.6 %	81.9 %	82.1 %	34.3 %	45.9%
	HS (8,7)	98.2 %	53.9 %	45.9 %	40.5 %	22.5 %	26.8 %	21.3 %	81.1 %	81.2 %	30.9 %	50.2%
	HS (9,8)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal

Tabel 8.4 Rincian Hasil Uji Coba UJ-P01 pada Panjang Data 1345 karakter

Efektifitas Kompresi Pada Panjang Data 1345		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	87.3 %	4.3 %	-5.0 %	1.1 %	-3.3 %	-6.1 %	13.5 %	-5.0 %	-5.6 %	20.7 %	10.2%
	HS (5,4)	92.0 %	6.1 %	-0.4 %	19.7 %	3.6 %	1.2 %	19.9 %	0.4 %	-1.6 %	29.0 %	17.0%
	HS (6,5)	95.1 %	7.1 %	20.4 %	28.3 %	12.9 %	24.2 %	23.9 %	65.9 %	67.4 %	32.0 %	37.7%
	HS (7,6)	97.1 %	10.2 %	43.6 %	40.3 %	21.0 %	24.5 %	24.5 %	81.9 %	81.4 %	34.8 %	45.9%

	HS (8,7)	98.1 %	95.6 %	46.8 %	40.8 %	22.5 %	26.5 %	21.0 %	81.2 %	81.5 %	31.2 %	54.5%
	HS (9,8)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal

Tabel 8.5 Rincian Hasil Uji Coba UJ-P01 pada Panjang Data 1870 karakter

Efektifitas Kompresi Pada Panjang Data 1870		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	87.3 %	3.0 %	-4.2 %	2.6 %	-3.4 %	-5.0 %	-8.7 %	-4.2 %	-8.8 %	-8.1 %	5.1%
	HS (5,4)	92.0 %	4.9 %	2.2 %	9.4 %	6.0 %	5.9 %	-6.8 %	-0.6 %	-4.2 %	2.9 %	11.2%
	HS (6,5)	95.1 %	5.8 %	15.0 %	20.5 %	22.1 %	32.6 %	-5.8 %	7.7 %	-2.8 %	13.3 %	20.4%
	HS (7,6)	97.1 %	24.5 %	22.1 %	29.1 %	28.9 %	36.8 %	-0.3 %	18.6 %	1.6 %	19.6 %	27.8%
	HS (8,7)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal
	HS (9,8)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal

Tabel 8.6 Rincian Hasil Uji Coba UJ-P02 pada Jarak ±10 Meter

Pengiriman Data Via ZigBee (±10 meter)		Percobaan ke-										Akurasi
		1	2	3	4	5	6	7	8	9	10	
Jumlah Data Masuk	Sesuai	41	41	41	41	41	41	41	41	41	41	100.0%
	Tidak	0	0	0	0	0	0	0	0	0	0	

Tabel 8.7 Rincian Hasil Uji Coba UJ-P02 pada Jarak ±20 Meter

Pengiriman Data Via ZigBee (±20 meter)		Percobaan ke-										Akurasi
		1	2	3	4	5	6	7	8	9	10	
Jumlah Data Masuk	Sesuai	40	36	39	38	36	35	40	38	27	27	86.8%
	Tidak	1	5	2	3	5	6	1	3	14	14	

Tabel 8.8 Rincian Hasil Uji Coba UJ-P02 pada Jarak ±30 Meter

Pengiriman Data Via ZigBee (±30 meter)		Percobaan ke-										Akurasi
		1	2	3	4	5	6	7	8	9	10	
Jumlah Data Masuk	Sesuai	27	32	37	38	30	34	26	20	12	29	69.5%
	Tidak	14	9	4	3	11	7	15	11	29	12	

Tabel 8.9 Rincian Hasil Uji Coba UJ-P03 pada Jarak ±10 Meter

Pengiriman Data Via ZigBee (±10 meter)		Percobaan ke-										Akurasi	Total Akurasi
		1	2	3	4	5	6	7	8	9	10		
End device ke *Router*	Sesuai	41	41	41	41	41	41	41	41	41	41	100.0%	100.0%
	Tidak	0	0	0	0	0	0	0	0	0	0		
Router ke Cordinator	Sesuai	41	41	41	41	41	41	41	41	41	41	100.0%	
	Tidak	0	0	0	0	0	0	0	0	0	0		

Tabel 8.10 Rincian Hasil Uji Coba UJ-P03 pada Jarak ±20 Meter

Pengiriman Data Via ZigBee (±10 meter)		Percobaan ke-										Akurasi	Total Akurasi
		1	2	3	4	5	6	7	8	9	10		
End device ke *Router*	Sesuai	38	40	29	38	40	40	38	31	40	38	90.7%	90.6%
	Tidak	3	1	12	3	1	1	3	10	1	3		
Router ke Cordinator	Sesuai	38	40	29	38	40	40	38	31	40	37	90.5%	
	Tidak	3	1	12	3	1	1	3	10	1	4		

Tabel 8.11 Rincian Hasil Uji Coba UJ-P03 pada Jarak ±30 Meter

Pengiriman Data Via ZigBee (±10 meter)		Percobaan ke-										Akurasi	Total Akurasi
		1	2	3	4	5	6	7	8	9	10		
End device ke *Router*	Sesuai	31	40	40	23	40	40	40	40	40	9	83.7%	82.6%
	Tidak	10	1	1	18	1	1	1	1	1	32		
Router ke Cordinator	Sesuai	29	37	40	23	40	37	40	40	39	9	81.5%	
	Tidak	12	4	1	18	1	4	1	1	2	32		

Tabel 8.12 Rincian Hasil Uji Coba UJ-P03 pada Panjang Data 584 Karakter

Waktu Kompresi Pada Panjang Data 584		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.021668	0.061192	0.063682	0.066196	0.062816	0.062168	0.063084	0.0624	0.061324	0.054956	0.058035
	HS (5,4)	0.014168	0.054824	0.052968	0.058036	0.056048	0.047524	0.052024	0.03274	0.05576	0.045132	0.046922
	HS (6,5)	0.010436	0.051796	0.044422	0.05368	0.054028	0.039334	0.050396	0.02044	0.051008	0.038812	0.041416
	HS (7,6)	0.009128	0.051924	0.041576	0.052336	0.051124	0.01888	0.023332	0.017176	0.020936	0.036704	0.032312
	HS (8,7)	0.008444	0.04648	0.044192	0.035056	0.039484	0.019624	0.023364	0.018768	0.020644	0.038712	0.029477
	HS (9,8)	0.009796	0.04566	0.044852	0.036096	0.038076	0.019476	0.025196	0.019356	0.022348	0.0396	0.030046

Tabel 8.13 Rincian Hasil Uji Coba UJ-P03 pada Panjang Data 980 Karakter

Waktu Kompresi Pada Panjang Data 980		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.036	0.1030122	0.0991088	0.1043288	0.105516	0.1052766	0.0980722	0.1078044	0.1044366	0.105556	0.096876
	HS (5,4)	0.0233166	0.009198	0.0881322	0.0795844	0.008662	0.0823244	0.008842	0.0975552	0.0886366	0.0882766	0.081484
	HS (6,5)	0.0165044	0.0877244	0.0781688	0.0649644	0.0787772	0.0568844	0.0841044	0.0741722	0.0735766	0.0730922	0.068796
	HS (7,6)	0.0138122	0.0885566	0.006996	0.003318	0.007762	0.0037476	0.0354566	0.00568	0.0679722	0.0665166	0.054723
	HS (8,7)	0.012844	0.0636244	0.0656444	0.0361612	0.0728366	0.0033568	0.0377444	0.0610488	0.0728966	0.0721722	0.052898
	HS (9,8)	0.0137322	0.00636	0.0625556	0.003756	0.0040268	0.0020992	0.0390644	0.0648766	0.0785044	0.0778166	0.049897

Tabel 8.14 Rincian Hasil Uji Coba UJ-P03 pada Panjang Data 980 Karakter

Waktu Kompresi Pada Panjang Data 1280		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.046784	0.134308	0.14115	0.13454	0.14152	0.14224	0.129448	0.140028	0.140336	0.122632	0.127334
	HS (5,4)	0.029956	0.120004	0.128064	0.10656	0.125156	0.12678	0.114456	0.126348	0.126988	0.104836	0.110915
	HS (6,5)	0.021332	0.115304	0.104852	0.00956	0.113136	0.009804	0.109652	0.061012	0.059364	0.009682	0.087705
	HS (7,6)	0.017192	0.113122	0.007988	0.008186	0.106356	0.009696	0.112556	0.003802	0.003806	0.009118	0.077806
	HS (8,7)	0.01574	0.006753	0.008215	0.008442	0.116148	0.010511	0.136576	0.003652	0.003832	0.010132	0.0784
	HS (9,8)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal

Tabel 8.15 Rincian Hasil Uji Coba UJ-P03 pada Panjang Data 1345 Karakter

Waktu Kompresi Pada Panjang Data 1345		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.049096	0.140392	0.148608	0.141748	0.148216	0.148996	0.136256	0.147052	0.147492	0.128608	0.133646
	HS (5,4)	0.031416	0.126404	0.134668	0.111352	0.131168	0.132336	0.12044	0.132528	0.133572	0.109812	0.116421
	HS (6,5)	0.022332	0.120116	0.108376	0.099092	0.119226	0.105184	0.1155	0.063644	0.062128	0.101736	0.091741
	HS (7,6)	0.017664	0.118092	0.082736	0.08562	0.111744	0.105064	0.118672	0.040044	0.04092	0.096272	0.081683
	HS (8,7)	0.016248	0.068128	0.085728	0.088632	0.123212	0.111364	0.144996	0.039204	0.039576	0.106688	0.082378
	HS (9,8)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal

Tabel 8.16 Rincian Hasil Uji Coba UJ-P03 pada Panjang Data 1870 Karakter

Waktu Kompresi Pada Panjang Data 1870		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.067972	0.19612	0.2062668	0.2000064	0.2037288	0.2055966	0.2089444	0.2057444	0.2095488	0.2083688	0.191235
	HS (5,4)	0.0431688	0.1763332	0.1830884	0.1770008	0.1769	0.1767796	0.1902288	0.1845088	0.1884856	0.1794644	0.167594
	HS (6,5)	0.0305552	0.1697756	0.1606048	0.15772	0.1497244	0.1332044	0.1841488	0.1686444	0.1819244	0.16003	0.149662
	HS (7,6)	0.0246044	0.1421322	0.1507764	0.1468088	0.1416122	0.12818	0.1810336	0.1556	0.1790844	0.1513996	0.140122
	HS (8,7)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal
	HS (9,8)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal

Tabel 8.17 Rincian Hasil Uji Coba UJ-P04 pada Panjang Data 584 Karakter

Waktu Dekompresi Pada Panjang Data 584		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.007508	0.0218	0.022624	0.023324	0.022284	0.021784	0.022352	0.021624	0.021844	0.019212	0.020436
	HS (5,4)	0.004494	0.021284	0.020588	0.0226	0.021548	0.018476	0.020104	0.01232	0.021484	0.01728	0.018062
	HS (6,5)	0.004028	0.020812	0.018168	0.021636	0.0217	0.015712	0.02034	0.00842	0.020184	0.015476	0.016648
	HS (7,6)	0.003556	0.021136	0.017324	0.021244	0.02062	0.007972	0.009912	0.007392	0.008728	0.014976	0.013286
	HS (8,7)	0.003316	0.017804	0.017348	0.013976	0.015072	0.008064	0.00986	0.007456	0.00888	0.014912	0.011669
	HS (9,8)	0.003252	0.017548	0.017664	0.014424	0.014684	0.008332	0.010196	0.007768	0.009224	0.015464	0.011856

Tabel 8.18 Rincian Hasil Uji Coba UJ-P04 pada Panjang Data 980 Karakter

Waktu Dekompresi Pada Panjang Data 980		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.012284	0.035876	0.035028	0.036416	0.036992	0.036704	0.033568	0.03786	0.036616	0.036988	0.033833
	HS (5,4)	0.009132	0.03538	0.033396	0.03058	0.033016	0.031452	0.033584	0.037476	0.0337	0.03354	0.031182
	HS (6,5)	0.006336	0.035096	0.031668	0.025748	0.031816	0.02226	0.03402	0.029592	0.02906	0.029048	0.027464
	HS (7,6)	0.005524	0.0355	0.028088	0.013848	0.03194	0.014696	0.015896	0.02222	0.026916	0.026316	0.022094
	HS (8,7)	0.005112	0.024004	0.024576	0.014084	0.028384	0.012812	0.015588	0.022248	0.025992	0.0255	0.01983
	HS (9,8)	0.00498	0.023648	0.023448	0.014484	0.017084	0.009504	0.015828	0.022508	0.025996	0.025396	0.018288

Tabel 8.19 Rincian Hasil Uji Coba UJ-P04 pada Panjang Data 1280 Karakter

Waktu Dekompresi Pada Panjang Data 1280		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.015876	0.046616	0.049724	0.047216	0.049572	0.049604	0.044404	0.049548	0.049648	0.041994	0.044415
	HS (5,4)	0.011808	0.046156	0.049404	0.040054	0.048236	0.048076	0.042816	0.048066	0.0493	0.039524	0.042452
	HS (6,5)	0.009544	0.04602	0.042028	0.037892	0.045032	0.039488	0.042	0.02362	0.022652	0.038012	0.034629
	HS (7,6)	0.007024	0.044916	0.031672	0.032804	0.04158	0.03962	0.041588	0.015384	0.01536	0.03604	0.030599
	HS (8,7)	0.006488	0.025948	0.029416	0.03186	0.03924	0.037412	0.040564	0.015192	0.015192	0.035996	0.027731
	HS (9,8)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal

Tabel 8.20 Rincian Hasil Uji Coba UJ-P04 pada Panjang Data 1345 Karakter

Waktu Dekompresi Pada Panjang Data 1345		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.016644	0.048788	0.052208	0.049628	0.051996	0.051872	0.046696	0.052012	0.052156	0.043912	0.046591
	HS (5,4)	0.012376	0.048838	0.051904	0.042404	0.050688	0.050104	0.045144	0.051128	0.051796	0.041426	0.044518
	HS (6,5)	0.010012	0.048288	0.043344	0.039416	0.047288	0.041604	0.044268	0.024572	0.023884	0.039804	0.036244
	HS (7,6)	0.007336	0.047224	0.033302	0.03432	0.043648	0.041716	0.043864	0.016112	0.016448	0.037724	0.032141
	HS (8,7)	0.006816	0.026388	0.030504	0.033304	0.041252	0.039376	0.042684	0.015912	0.015808	0.037732	0.028978
	HS (9,8)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal

Tabel 8.21 Rincian Hasil Uji Coba UJ-P04 pada Panjang Data 1870 Karakter

Waktu Dekompresi Pada Panjang Data 1870		Percobaan ke-										Rata - rata
		1	2	3	4	5	6	7	8	9	10	
Konfigurasi	HS (4,3)	0.022916	0.068004	0.071952	0.069312	0.070876	0.071644	0.073064	0.071908	0.073212	0.072968	0.066589
	HS (5,4)	0.016948	0.067448	0.069956	0.067048	0.067268	0.067192	0.073028	0.071136	0.072344	0.06898	0.064135
	HS (6,5)	0.013716	0.067488	0.063748	0.06202	0.05898	0.053052	0.073368	0.067628	0.072576	0.06418	0.059676
	HS (7,6)	0.010972	0.056604	0.059596	0.056564	0.055104	0.05058	0.071548	0.062212	0.071072	0.06056	0.055481
	HS (8,7)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal
	HS (9,8)	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal	gagal

BIODATA PENULIS

Muhamad Hendri Febriansyah merupakan anak kembar dari pasangan Bapak Jamaludin dan Ibu Suryati. Lahir di Mataram pada tanggal 28 Februari 1996. Penulis menempuh pendidikan formal dimulai SDN 29 Mataram (2002-2008), MTsN 1 Mataram (2008-2011), SMAN 1 Mataram (2011-2014) dan S1 Departemen Informatika ITS (2014-2018). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi seperti Himpunan Mahasiswa Teknik Computer-Informatika (2015-2016) dan KMI (2015-2017). Penulis juga aktif dalam berbagai kegiatan kepanitiaan yaitu SCHEMATICS 2015-2016 divisi Keamanan dan Perizinan. Penulis juga menyukai kegiatan pecinta alam. Penulis memiliki hobi traveling dan bermain game. Penulis dapat dihubungi melalui email : hendrifebriansyah28@gmail.com.