TUGAS AKHIR – KI141502

Rancang Bangun Sistem Monitoring Struktur Bangunan Berbasis Jaringan Sensor Nirkabel dengan Analisis Nilai Modal Struktur (Studi Kasus Prototype Jembatan)

M ILHAM MIRZA A
NRP 5113100149

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

Dosen Pembimbing II
Dr. Dwa Desa Warnana

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

Rancang Bangun Sistem Monitoring Struktur Bangunan Berbasis Jaringan Sensor Nirkabel dengan Analisis Nilai Modal Struktur (Studi Kasus Prototype Jembatan)

M ILHAM MIRZA A
NRP 5113100149

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

Dosen Pembimbing II
Dr. Dwa Desa Warnana

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

UNDERGRADUATE THESES – KI141502

STRUCTURAL MONITORING SYSTEM DESIGN BASED ON WIRELESS SENSOR NETWORK WITH STRUCTURAL MODAL VALUE ANALYSIS (CASE STUDY BRIDGE PROTOTYPE)

M ILHAM MIRZA A
NRP 5113100149

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

Second Advisor
Dr. Dwa Desa Warnana

Department of Informatics
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM MONITORING STRUKTUR BANGUNAN BERBASIS JARINGAN SENSOR NIRKABEL DENGAN ANALISIS NILAI MODAL STRUKTUR (STUDI KASUS PROTOTYPE JEMBATAN)

TUGAS AKHIR

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

Oleh:
M ILHAM MIRZA A
NRP: 5113100149

Disetujui oleh Pembimbing Tugas Akhir:

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

2. Dr. Dwa Desa Warnana ..... (Pembimbing 2)
   (NIP. 197601232000031001)

SURABAYA
JUNI, 2017

RANCANG BANGUN SISTEM MONITORING STRUKTUR BANGUNAN BERBASIS JARINGAN SENSOR NIRKABEL DENGAN ANALISIS NILAI MODAL STRUKTUR (STUDI KASUS PROTOTYPE JEMBATAN)

Nama Mahasiswa	**: M ILHAM MIRZA A**
NRP	**: 5113100149**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono S.Kom, M.Eng., Ph.D.**
Dosen Pembimbing 2	**: Dr. Dwa Desa Warnana**

Abstrak

Sistem monitoring struktural adalah sistem yang digunakan untuk memantau suatu struktur bangunan untuk mendapatkan status kondisi dari struktur bangunan tersebut. Sistem monitoring ini akan mendapatkan nilai modal dari struktur yang diamati. Nilai modal tersebut adalah eigen frequency dari struktur yang dapat menunjukkan perubahan dari struktur bangunan.

Pada tugas akhir ini dikembangkan prototipe sistem monitoring struktural dengan daya rendah. Alat yang digunakan untuk proses pengambilan data adalah Arduino *Uno dengan accelerometer ADXL-345 3-axis model GY-291 dengan media pengiriman nRF2401.Metode yang digunakan untuk ekstraksi nilai eigen frequency dari data getaran adalah algoritma FFT (Fast Fourier Transform).*

Uji coba yang dilakukan dibagi menjadi dua bagian; pertama menggunakan prototipe jembatan dimana perubahan struktur diwakili oleh longgarnya baut pada prototype jembatan. Percobaan yang kedua menggunakan jembatan sesungguhnya dengan membandingkan struktur dari lokasi yang berbeda. Hasil dari uji coba tersebut diperoleh akurasi yang cukup baik. Pada uji coba pada prototipe jembatan akurasi yang didapatkan sebesar

93,33% sedangkan pada uji coba pada jembatan sebenarnya sebesar 67,5%.

Kata kunci: Structure monitoring, Fast Fourier Transform, ADXL345, nRF2401, Eigen frequency.

STRUCTURAL MONITORING SYSTEM DESIGN BASED ON WIRELESS SENSOR NETWORK WITH STRUCTURAL MODAL VALUE ANALYSIS(CASE STUDY BRIDGE PROTOTYPE)

Student's Name	**: M ILHAM MIRZA A**
Student's ID	**: 5113100149**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Waskitho Wibisono S.Kom, M.Eng., Ph.D.**
Second Advisor	**: Dr. Dwa Desa Warnana**

Abstract

Structural health monitoring system is a system built with purpose to monitor a structural object to discover the condition of the structure. This monitoring system will be discovering modal value of the structure. That modal value is eigen frequency,where if there is eigen frequency change means there is a change in structural object.

In this undergraduate thesis, a structural monitoring system with low power usage has been developed. The device used to collect data from prototype is an Arduino *Uno module equipped with an ADXL-345, 3-axis accelerometer model GY-291 and nRF24l01 as data transmission media. FFT(Fast Fourier Transform) has been used as the method to extract eigen frequency from vibration data.*

The experiment is divided into two parts; first part of the experiment is conducted using a bridge prototype where the bolt will be used as a mean to change the structure. The second part of the experiment is conducted on a real bridge where the eigen frequency of one location on the bridge is compared to eigen frequency of different location on the structure.

The result of experiments done in this thesis were adequate. Accuracy of the prototype experiment was 93,33% and the bridge experiment was 67,5%.

Keywords : Structural Monitoring System, Fast Fourier Transform, ADXL345, nRF2401, Eigen frequency.

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:

"**RANCANG BANGUN SISTEM MONITORING STRUKTUR BANGUNAN BERBASIS JARINGAN SENSOR NIRKABEL DENGAN ANALISIS NILAI MODAL STRUKTUR (STUDI KASUS PROTOTYPE JEMBATAN)**"

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. Kedua orang tua penulis yang selalu memberikan dukungan doa, moral, dan material yang tak terhingga kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Waskitho Wibisono S.Kom, M.Eng., Ph.D. selaku pembimbing I yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
3. Bapak Dr. Dwa Desa Warnana selaku pembimbing II yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini.
4. Ibu Diana Purwitasari, S.Kom., M.Sc**.** selaku dosen wali penulis dari sejak diterima di Teknik Informatika sampai dengan semester tujuh yang telah memberikan arahan, masukan dan motivasi kepada penulis.
5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala departemen Teknik Informatika ITS.

6. Bapak Dr. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator Tugas Akhir penulis.
7. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
8. Teman-teman Keluarga Muslim Informatika, yang sudah banyak meluruskan penulis.
9. Teman-teman angkatan 2013, yang sudah mendukung saya selama perkuliahan.
10. 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, Juni 2017

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Jembatan-jembatan di Indonesia maupun dunia mayoritas dibangun pada abad ke-20, dimana baja merupakan teknologi baru yang tidak diketahui bagaimana ketahanannya pada jangka waktu yang panjang. Terdapat banyak faktor yang memengaruhi kelayakan sebuah jembatan, baik dari sisi alami juga dari perbuatan manusia. Jembatan-jembatan di Indonesia kebanyakan memiliki sistem pemeliharaan yang sangat minim bahkan hampir tidak ada.[1] Karenanya dibutuhkan suatu sistem yang memantau ketahanan jembatan tersebut secara pasif dan memiliki harga terjangkau. Salah satu sistem monitoring kesehatan struktur adalah dengan mendapatkan nilai modal dari struktur yang diamati. Nilai modal adalah *eigen frequency* dari struktur yang menunjukkan perubahan struktur.

Pada penelitian sebelumnya, metode pencarian nilai modal struktur diterapkan pada perangkat yang tersambung dengan kabel dan memiliki harga yang cukup mahal. Perangkat tersebut memiliki biaya pemeliharaan yang cukup tinggi dikarenakan pemeliharaan kabel yang menghubungkan perangkat-perangkat tersebut. [2]

Teknologi jaringan sensor nirkabel merupakan teknologi yang relatif baru dan dapat diaplikasikan secara fleksibel dalam pemantauan kesehatan sebuah struktur. Tersedianya perangkat terjangkau seperti mikrokontroller Arduino UNO dapat digunakan sebagai pengganti perangkat yang mahal tersebut.[3]

Dalam penelitian ini, penulis menerapkan metode pencarian nilai modal struktur pada jembatan menggunakan *node* yang menjadikan Arduino UNO sebagai pemroses data. Penulis membuat sistem dimana *node* mengirim data *accelerometer* dari lokasi jembatan dan mengirim data tersebut melalui perangkat nrf24l01 sebagai *transreceiver* yang terpasang pada *node.* Data

yang dikirim akan diterima oleh sink node yang terpasang pada perangkat komputer.

1.2 Rumusan Masalah

Tugas akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana sistem dapat mencari nilai modal berdasarkan getaran yang tercatat akselerometer?
2. Bagaimana protokol pengiriman data yang digunakan untuk memenuhi kebutuhan diatas?
3. Bagaimana bentuk dan struktur data yang digunakan dalam pengiriman data getaran?
4. Bagaimana prosedur penerimaan data pada *sink node?*

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini memiliki batasan sebagai berikut:

1. Sensor yang digunakan untuk mendeteksi getaran adalah sensor jenis akselerometer ADXL345
2. Pengambilan data sensor meggunakan perangkat Arduino Uno tanpa perangkat penyimpanan memori tambahan
3. Jumlah sensor terbatas, sehingga jaringan sensor berukuran relatif kecil
4. Pengujian dilakukan di lingkungan sekitar ITS

1.4 Tujuan

Tujuan pembuatan Tugas Akhir ini adalah membuat suatu sistem pemantauan berbasis getaran yang dapat mendeteksi kerusakan struktur pada jembatan dan menjawab masalah yang terjadi saat pembuatan sistem, seperti metode yang digunakan untuk mencari nilai modal, sistem protokol pengiriman dan

penerimaan data dan bentuk struktur data yang digunakan untuk mengirimkan data pada sistem yang dibangun.

1.5 Manfaat

Manfaat yang didapatkan dari pengerjaan tugas akhir ini adalah adanya sistem terjangkau yang dapat memantau getaran yang terjadi pada jembatan, hasil pemantauan dapat digunakan untuk mencari perubahan nilai *eigen frequency* yang merupakan nilai modal struktur sehingga kerusakan dapat diperkirakan.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal

Tahap awal tugas akhir ini adalah menyusun proposal tugas akhir. Proposal ini berisi tentang perencanaan implementasi SHM (*Structural Health Monitoring*) menggunakan jaringan sensor nirkabel berbasis getaran, sebagai solusi pemantauan kesehatan bangunan.

1.6.2 Studi Literatur

Pada tahap ini dilakukan untuk mencari informasi dan studi literatur apa saja yang dapat dijadikan referensi untuk membantu pengerjaan Tugas Akhir ini. Studi literatur yang dilakukan dalam pengerjaan Tugas Akhir ini adalah mengenai implementasi metode FFT pada analisis data getaran untuk menentukan *eigen frequency* untuk memantau kesehatan struktur bangunan.

1.6.3 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal Tugas Akhir. Untuk membangun metode yang telah dirancang sebelumnya, maka dilakukan implementasi dengan menggunakan suatu perangkat lunak yaitu NetBeans dan Arduino IDE sebagai IDE dan basis data SQLite.

1.6.4 Pengujian dan Evaluasi

Pada tahap ini metode yang telah disusun diuji coba dengan menggunakan prototype jembatan yang telah dibuat juga jembatan sesungguhnya. Pada percobaan bagian pertama, prototipe jembatan akan diubah strukturnya dengan meloggarkan baut-baut yang menyusunnya dan dicari nilai perubahan *eigen frequency*-nya. Pada bagian kedua uji coba, uji coba dilakukan di atas jembatan sesungguhnya dengan membandingkan nilai *eigen frequency* pada suatu lokasi dengan lokasi lain di atas jembatan.

1.6.5 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:

Bab I. Pendahuluan
Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.

Bab II. Tinjauan Pustaka
Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini. Secara garis besar, bab ini berisi tentang FFT, nilai modal, *structure monitoring.*

Bab III. Perancangan Perangkat Lunak
Bab ini berisi pembahasan mengenai perancangan dari metode FFT yang digunakan untuk mendeteksi nilai modal struktur.

Bab IV. Implementasi
Bab ini menjelaskan implementasi yang berbentuk *pseudocode* yang berupa *pseudocode* dari metode FFT dan ekstraksi nilai modal.

Bab V. Hasil Uji Coba dan Evaluasi
Bab ini berisikan hasil uji coba dari algoritma FFT yang digunakan untuk mendeteksi nilai modal yang sudah diimplementasikan pada *pseudocode*. Uji coba dilakukan dengan menggunakan prototype jembatan yang memiliki baut sebagai variabel yang diubah. Hasil evaluasi mencakup nilai *eigen frequency* dari struktur awal dan nilai *eigen frequency* pada struktur yang telah diubah.

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.

Daftar Pustaka
Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.

Biodata Penulis
Bab ini berisi biodata pennulis tugas akhir ini

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar dan alat-alat yang digunakan dalam pengembangan sistem pada tugas akhir. Teori-teori tersebut diantaranya adalah *modal value*, *Fast Fourier Transform* dan beberapa teori lain yang mendukung pembuatan tugas akhir.

2.1 Mikrokontroller Arduino UNO

Arduino Uno adalah papan mikrokontroller yang didesain berdasrkan ATmega328P (lihat gambar 2.1). Mikrokontroller Arduino Uno memiliki 14 pin input dan output, 6 input analog, unit pemroses *quartz crystal* yang beroperasi pada 16 MHz , *port* koneksi untuk kabel USB, *port* daya dan tombol *reset*.[4]

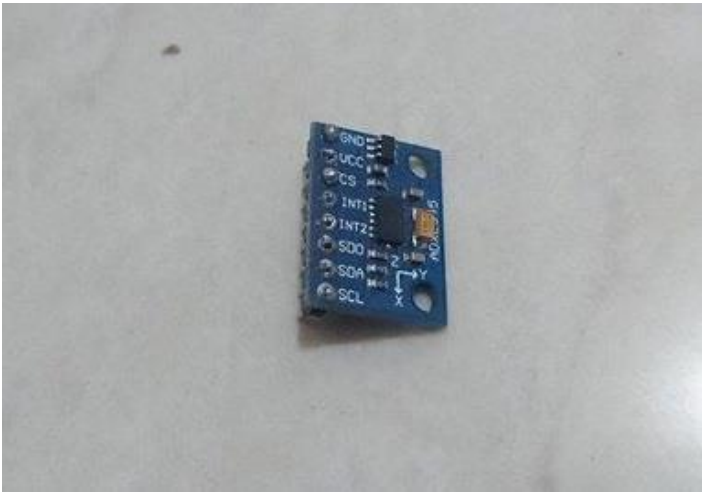
Gambar 2.1 Mikrokontroller Arduino Uno

Mikrokontroller akan berfungsi sebagai pemroses data yang akan dikirim oleh *node* pengirim data yang diletakkan pada lokasi uji coba dan sebagai pemroses data yang diterima oleh *sink node*.

2.2 Sensor Accelerometer ADXL-345

Accelerometer adalah sebuah jenis sensor yang mendeteksi besarnya akselerasi yang terjadi pada sensor. Penghitungan jumlah getaran dilakukan dengan menghitung banyaknya jumlah perubahan akselerasi pada sensor.

Accelerometer yang digunakan pada pembuatan tugas akhir ini adalah accelerometer versi ADXL-345 dengan nomor sei GY-291 (lihat gambar 2.2). Accelerometer ini merupakan perangkat jenis *Analog Device*. Perangkat ini mampu melakukan pengukuran dengan range ±2g, ±4g, ±8g, ±16g dimana g mewakili nilai gravitasi bumi sebesar 9,81 m/s^2 . Pengukuran tersebut dapat dilakukan dengan resolusi pengukuran cukup tinggi yaitu sebesar 13-bit.



Gambar 2.2 *Accelerometer* ADXL-345 Seri GY-291

Accelerometer ini menggunakan daya yang sangat rendah sebesar 23 μA pada saat pengukuran dan sebesar 0,1 μA saat keaadan *standby* pada tegangan 2,5V.[5]

2.3 Modul nRF24L01

Modul Wireless nRF24L01 (lihat gambar 2.3) adalah modul komunikasi jarak jauh yang memanfaatkan pita gelombang RF (Radio Frequency) 2.4GHz ISM (*Industrial, Scientific and Medical*) yang didesain untuk jaringan nirkabel yang membutuhkan penggunaan daya sangat rendah. Modul ini berupa sebuah chip *transreceiver* tunggal yang memiliki *baseband logic Enhanced Shockburst.*[6]

Gambar 2.3 Modul NRF24l01

Berikut adalah fitur-fitur yang dimiliki oleh *transreceiver* NRF24l01 [6]:

- Data rate dapat mencapai 2Mbps
- Penggunaan daya sangat rendah dengan rincian:
  - 11.3mA TX pada kekuatan *output* 0dBm
  - 12.3mA RX pada data rate 2Mbps
  - 22μA pada kondisi standby

  - 900nA pada kondisi kekurangan daya
- Regulator voltase tertanam pada chip
- Dapat melakukan *handling* paket secara otomatis

Modul ini cocok digunakan dengan Mikrokontroller Arduino. Modul ini akan menjadi pengirim data yang telah didapatkan sensor kepada server untuk langkah analisis getaran

2.4 FFT (Fast Fourier Transform)

FFT adalah algoritma yang menghitung transformasi diskrit Fourier dari sebuah sekuens atau kebalikannya.[7] Algoritma ini akan digunakan untuk mengubah data akselerasi yang didapatkan dari akselerometer dari bentuk *time domain* ke bentuk *frequency domain* dimana nilai modal (*eigen frequency*) dapat ditemukan.

Penerapan algoritma ini akan dilakukan dalam bahasa pemrograman Java menggunakan library JTransform. Library ini adalah library mutithreaded FFT open-source pertama yang ditulis dengan bahasa pemrograman Java.[8]

Berikut adalah fitur-fitur dari library JTransform [8]:

- Implementasi DFT (Discrete Fourier Transform) , DCT (Discrete Cosine Transform), DST (Discrete Sine Transform) dan DHT (Discrete Hartley Transform) dalam bahasa pemrograman Java
- Medukung 1, 2 dan 3 dimensi transformasi data
- Mendukung pengolahan data dalam bentuk *array* 1-dimensi dengan ukuran besar
- Peningkatan performa eksekusi algoritma FFT

2.5 Nilai Modal Struktur

Nilai modal struktur adalah nilai-nilai *natural* yang dimiliki sebuah struktur. Nilai-nilai tersebut diantaranya adalah:

1. Eigen frequency

Eigen frequency adalah frekuensi getaran yang terjadi pada struktur bangunan secara *natural* tanpa adanya gaya yang

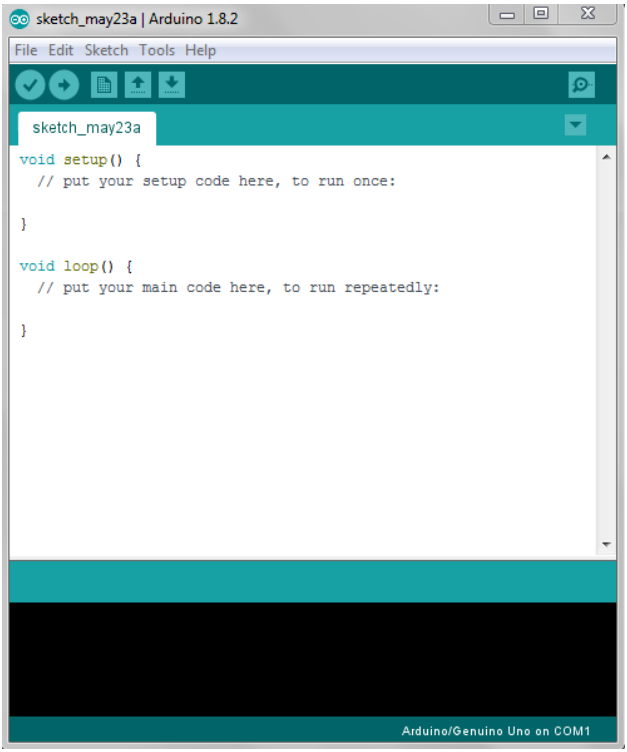
mempengaruhi struktur bangunan. Bisa juga disebut *natural frequency*. Perubahan *Eigen frequency* dapat menunjukkan perubahan dari struktur bangunan, sehingga kerusakan bisa dideteksi dengan mencari perubahan *eigen frequency* seiring waktu. Nilai ini yang akan penulis gunakan untuk mendeteksi perubahan struktur pada sistem yang dibuat.[9], [10]

2. *Mode Shapes*

Mode shape adalah pola getaran yang dihasilkan oleh sebuah sistem pada frekuensi tertentu. Pada setiap frekuensi mode shape yang dihasilkan sebuah sistem pasti berbeda.[11]

2.6 Arduino IDE

Arduino menyediakan IDE (*Integrated Development Environment*) yang merupakan aplikasi *cross-platforrm* ditulis dengan bahasa Java. Arduino IDE mencakup code-editor yang memiliki banyak fitur seperti *text cutting, text pasting*, fitur pencarian dan pencocokan teks, indentasi otomatis, *code highlighting* dan juga fitur untuk meng-*compile* kode dan mengunggah kode ke dalam papan Arduino dan banyak fitur lainnya.



Gambar 2.4 Tampilan awal Arduino IDE

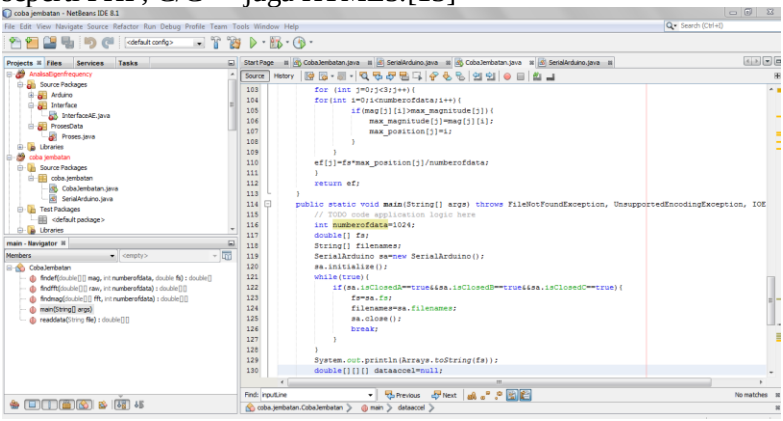
Program yang ditulis dengan IDE ini disebut *sketch. Sketch* disimpan di dalam komputer dengan format ekstensi *.ino.* Pada IDE versi sebelum 1.0, *sketch* disimpan dengan format ekstensi *.pde.*

Bahasa yang diggunakan untuk menulis *sketch* adalah bahasa campuran Java dengan bahasa C. *Sketch* memiliki dua fungsi dasar yaitu fungsi *setup()* dan *loop().* [12]

Pada pengerjaan tugas akhir ini penulis menggunakan Arduino IDE untuk menulis program *node* pengirim dan sink node. IDE yang digunakan penulis adalah Arduino IDE versi 1.6.11.

2.7 Netbeans IDE

Netbeans IDE ditujukan untuk pengembangan aplikasi dalam bahasa Java, tetapi IDE ini juga mendukung bahasa lain seperti PHP, C/C++ juga HTML5.[13]



Gambar 2.5 Tampilan Netbeans IDE

Netbeans IDE juga merupakan aplikasi *cross-platform*, dapat digunakan pada sistem operasi Windows, Linux, Solaris dan Mac OS X.[14]

Netbeans IDE memiliki modul-modul yang mendukung pengembangan aplikasi seperti modul Swing dan AWT untuk pengembangan antarmuka dalam bahasa Java.

Netbeans IDE ini digunakan penulis untuk pengembangan aplikasi pemantauan nilai modal struktur yang ditulis dalam bahasa pemrograman Java.

2.8 MySQL

MySQL adalah perangkat lunak sistem manajemen basis data SQL atau DBMS yang multi-thread, multi-user, dengan sekitar 6 juta pengguna di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi

GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus di mana penggunaannya tidak cocok dengan penggunaan GPL. [15]

Beberapa keistimewaan MySQL antara lain [16]:

1. **Portabilitas**. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.
2. **Perangkat lunak sumber terbuka**. MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis.
3. **Multi-user**. MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. '**Performance tuning**', MySQL memiliki kecepatan yang menakjubkan dalam menangani query sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5. **Ragam tipe data**. MySQL memiliki ragam tipe data yang sangat kaya, seperti signed / unsigned integer, float, double, char, text, date, timestamp, dan lain-lain.
6. **Perintah dan Fungsi**. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah Select dan Where dalam perintah (*query*).
7. **Keamanan**. MySQL memiliki beberapa lapisan keamanan seperti level subnetmask, nama host, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. **Skalabilitas dan Pembatasan**. MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (records) lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

9. **Konektivitas**. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, Unix soket (UNIX), atau Named Pipes (NT).
10. **Lokalisasi**. MySQL dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meski pun demikian, bahasa Indonesia belum termasuk di dalamnya.
11. **Antar Muka**. MySQL memiliki antar muka (interface) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (Application Programming Interface).
12. **Klien dan Peralatan**. MySQL dilengkapi dengan berbagai peralatan (tool) yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk online.
13. **Struktur tabel**. MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan basis data lainnya semacam PostgreSQL ataupun Oracle.

Dalam penelitian ini MySQL digunakan untuk menyimpan data detail pemantauan kesehatan struktur sesuai waktu pengambilan data. Data yang disimpan dalam basis data MySQL yaitu; waktu pengambilan data, *node* pengambil data, *sampling rate* dan nama berkas berekstensi *.txt* di mana aplikasi menulis data akselerasi yang diambil oleh akselerometer.

2.9 Arduino to Java

Data yang didapat oleh sink node dapat diambil dengan menghubungkan Netbeans dengan *serialport* dimana sink node terhubung. Pengambilan data ini dilakukan dengan menggunakan library *rxtx* yang mendukung pembacaan *serialport* dalam bahasa pemrograman Java.

Data selanjutnya ditulis dalam file berekstensi *.txt* dan data detail pengambilan disimpan ke dalam basis data MySQL. Semua proses ini dilakukan oleh komputer bukan oleh *sink node*.

BAB III
ANALISIS DAN PERANCANGAN

Perancangan merupakan bagian penting dari pembuatan perangkat lunak yang terdiri dari perencanaan-perencanaan. Bab ini membahas mengenai perancangan jaringan sensor yang akan digunakan untuk mengambil data dan implementasi metode pengolahan data dari sistem monitoring struktur bangunan yang dibangun penulis. Pembahasan akan meliputi deskripsi umum, perancangan dan implementasi.

3.1 Deskripsi Umum

Sistem monitoring struktur yang dibangun pada penelitian ini merupakan sistem *monitoring* struktur berjenis *output-only*, yaitu sistem yang memantau keadaan struktur berdasarkan data yang diambil sensor. Data yang diambil sensor merupakan data akelserasi yang terjadi pada struktur bangunan dan dari data tersebut akan ditemukan nilai modal struktur untuk mengetahui keadaan struktur yang dipantau.

Sistem monitoring ini menggunakan data akselerasi yang diambil oleh *node* pengirim, dimana *node* pengirim terpasang akselerometer dan *node* pngirim terpasang pada struktur bangunan yang dipantau. Kemudian data dikirim ke *sink node* menggunakan *transreceiver* nRF24l01 yang terpasang pada *node* pengirim dan sink node. Data yang diterima oleh *sink node* akan diolah oleh komputer untuk dicari nilai modal strukturnya.

3.2 Arsitektur Sistem

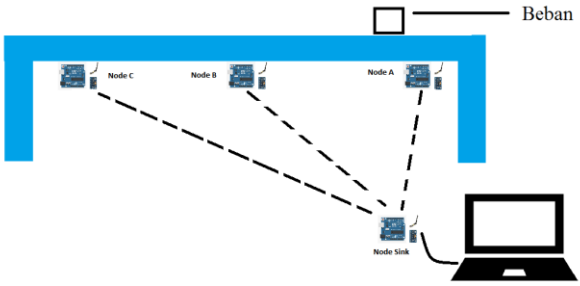
Arsitektur sistem monitoring struktur ini akan terdiri dari *node-node* sensor dan komputer sebagai pengolah data getaran. Seperti yang sudah disebutkan di bab sebelumnya, Sistem yang dibangun pada penelitian ini menggunakan mikrokontroller Arduino sebagai pengolah data awal, sensor getar (akselerometer)

ADXL-345 dengan nomor seri GY-291 sebagai sensor yang mengambil nilai akselerasi yang terjadi pada struktur bangunan dan modul radio NRF24l01 sebagai *transreceiver* data dari *node* pengirim kepada *sink node* yang tersambung dengan komputer untuk pengolahan data lebih lanjut. Setiap *node* akan menggunakan baterai sebagi sumber energi.

Setelah pengiriman data dilakukan, data pada *sink node* akan diolah dengan komputer yang tersambung pada *sink node*. *Sink node* akan terhubung dengan komputer melalui kabel USB secara fisik dan menggunakan *library rxtx* untuk mengambil data dari *serial port* yang terhubung pada *sink node*. Data yang sampai ke komputer akan diolah menggunakan algoritma FFT untuk mencari nilai *magnitude* tiap getaran. Kemudian data diubah dari yang asalnya berformat *time-domain* menjadi *frequency-domain* dengan menggunakan rumus pencarian nilai frekuensi. Puncak pertama dari data yang telah diubah tersebut adalah *eigen frequency*-nya.

Pengambilan data dilakukan secara *live* dimana data yang masuk pada *sink node* akan langsung diproses untuk dicari nilai *eigen frequency*-nya.

Arsitektur sistem dapat dilihat pada gambar 3.1.



Gambar 3.1 Arsitektur Sistem

3.3 Perancangan Perangkat Keras

Perangkat keras dari penelitian ini terdiri dari tiga *node* pemantau struktur (pengambil data), satu *sink node* dan satu buah komputer.

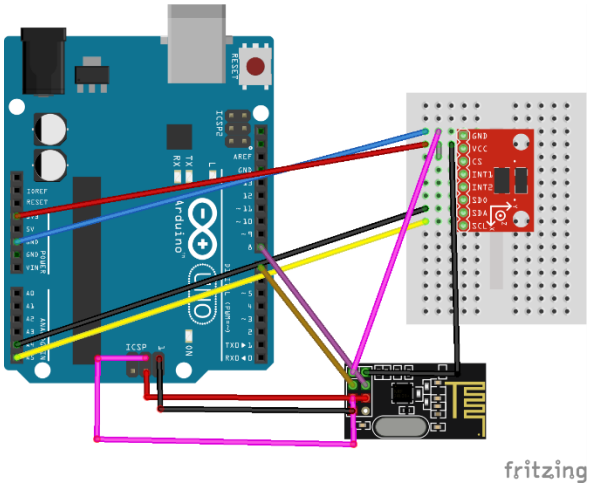
Penjelasan komponen adalah sebagai berikut:

- *Node* pengambil data

  Node ini terdiri atas :

  1. Mikrokontroller Arduino, sebagai pengatur pengambilan data dan pengiriman data dari *node*
  2. Modul radio nRF24l01, sebagai media pengirim dan penerima data dari *node* lain
  3. Modul sensor akselerometer, sebagai pengambil data s*ample* getaran pada titik *node* berada
  4. Baterai kotak 9v sebagai sumber energi

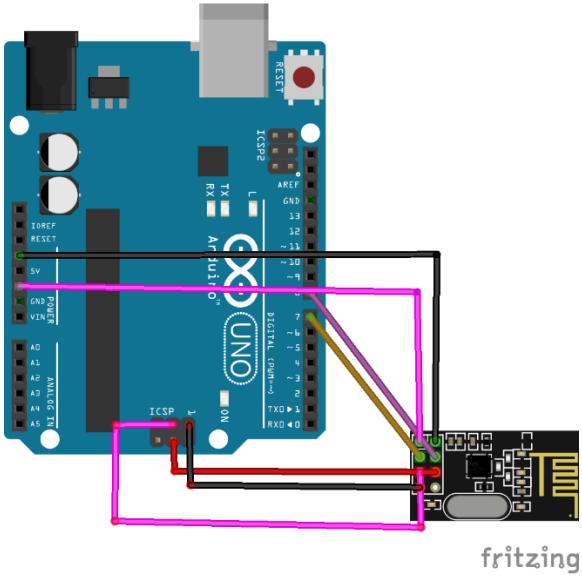
  Gambar rangkaian *node* pengambil data dapat dilihat pada gambar 3.2.



Gambar 3.2 Rangkaian *Node* Pengambil Data

- *Node Sink*
  Node ini terdiri atas :
  1. Mikrokontroller Arduino, sebagai pengatur penerima data dari *node* pengambil data
  2. Modul radio nRF24l01, sebagai media penerima data dari *node* lain

  Gambar rangkaian *sink node* dapat dilihat pada gambar 3.3

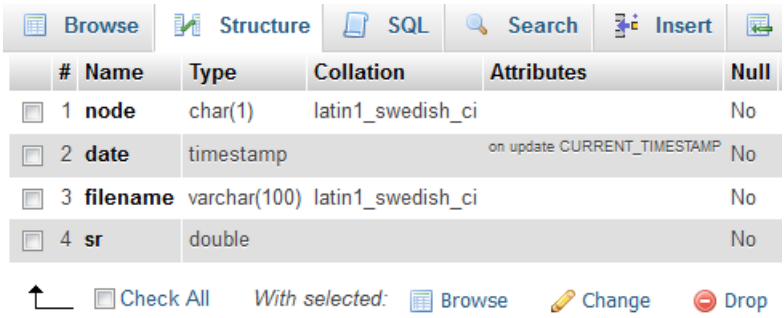


Gambar 3.3 Rangkaian *Sink Node*

- Komputer
  Komputer akan menjalankan tugas sebagai pengolah data getaran untuk mencari nilai modal struktur.

3.4 Perancangan *Database*

Basis data MySQL digunakan untuk menyimpan detail pengambilan data yang telah dilakukan. Tabel penyimpana n data ini dinamakan tabel *filet*. Data yang disimpan meliputi *node*, *sampling rate*, waktu pengambilan dan nama berkas tempat penulisan data dilakukan. Rancangan tabel basis data dapat dilihat pada gambar 3.4.



Gambar 3.4 Atribut Table *file*

Penjelasan atribut tabel dapat dilihat pada tabel 3.1

Tabel 3.1 Detail Atribut Tabel *file*

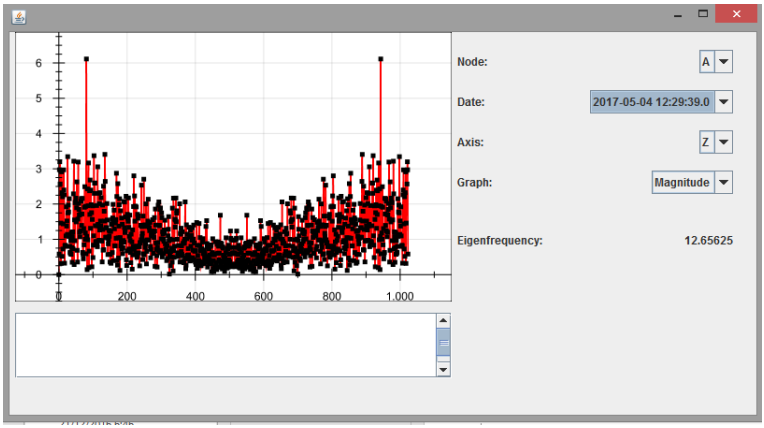
No.	Nama Atribut	Penjelasan	Tipe Data	Nilai Data
1	*Node*	Atribut *node* digunakan untuk membedakan sumber data. Dikarenakan *node* yang digunakan hanya tiga, maka nilai *node* terbatas	Char(1)	‘A’,’B’,’C’

		sampai huruf ‘C’.		
2	Date	Atribut *date* digunakan untuk menyimpan waktu pengambilan data	Timestamp	
3	Filename	Atribut *filename* digunakan unruk menyimpan nama berkas tempat penulisan data dilakukan	varchar	
4	sr	Atribut *sr* digunakan untuk menyimpan nilai *sampling rate* yang akan digunakan untuk pebgolahan data selanjutnya.	Double	

3.5 Perancangan Antarmuka *Monitoring* Struktur

Sistem yang dibangun pada tugas akhir ini dibangun menggunakan bahasa pemrograman Java dan bebentuk aplikasi *desktop* dimana aplikasi hanya dapat diakses pada komputer yang memiliki aplikasi ini saja. Rancangan antarmuka juga ditulis dengan bahasa pemrograman Java menggunakan *library Swing* yang telah disediakan oleh NetBeans IDE.

Rancangan antarmuka aplikasi dapat dilihat pada gambar 3.5.



Gambar 3.5 Rancangan Antarmuka

Algoritma ini digunakan untuk mengubah data getaran ke bentuk data *Fourier*. Dari data tersebut dapat diubah menjadi data magnitude dimana *eigen frequency* dapat diekstrak. Pengimplementasi algoritma ini menggunakan library jTransform. Output dari proses ini adalah data *fourier* yang berukuran dua kali jumlah data getaran yang diproses.

3.6 Diagram Alir Sistem

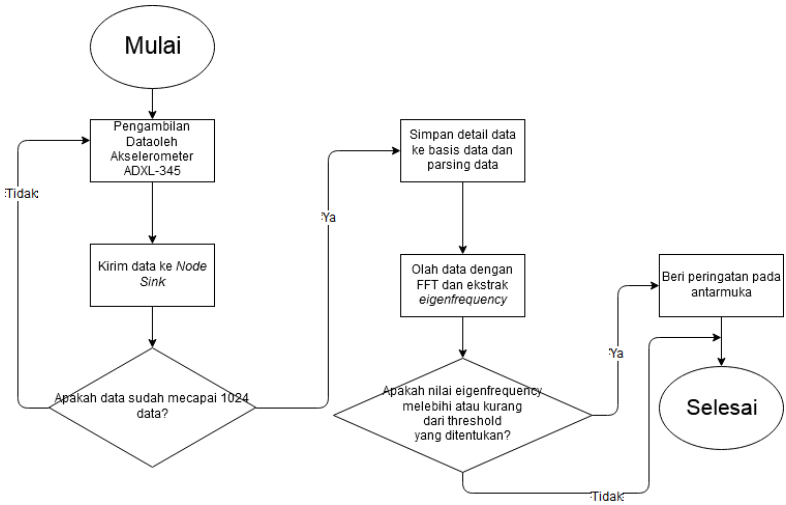
Diagram alir sistem dibuat untuk memudahkan dalam merancang dan memahami seluruh proses yang dilakukan di dalam sistem. Diagram alir terdiri dari diagram alir keseluruhan sistem, diagram alir pengamblian dan pengiriman data dari *node* pemantau ke *sink node*, diagram alir pencatatan dan penyimpanan data dan diagram alir ekstraksi *eigen frequency*.

3.6.1 Diagram Alir Keseluruhan Sistem

Rancangan sistem *monitoring* struktur bangunan berbasis jaringan sensor dengan analisis nilai modal struktur, dimulai

dengan mengambil nilai getaran dari prototype jembatan. Tahap selanjutnya adalah penyusunan data untuk dikirim ke *sink node*.

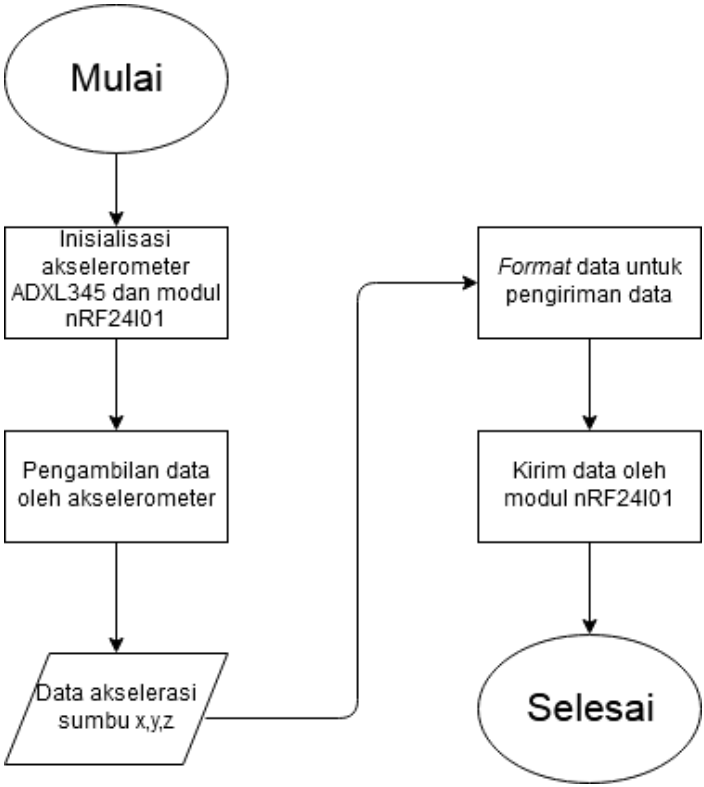
Proses pengiriman dilakukan secara konstan sampai jumlah data yang terkirim sebanyak 1024 data pada *sink node*. Data selanjutnya diolah menjadi bentuk magnitude menggunakan algoritma FFT. Dari data magnitude tersebut dapat diambil nilai *eigen frequency* dari struktur prototype. Diagram alir sistem dilihat pada gambar 3.6.



Gambar 3.6 Diagram Alir Keseluruhan

3.6.2 Diagram Alir Pengambilan Data

Pengambilan data dilakukan pada *node* pemantau dan data dikirimkan ke *sink node* untuk pengolahan lebih lanjut. Proses yang terjadi pada tahap ini adalah inisialisasi akselerometer ADXL345, *formatting data* oleh modul nRF24l01. Diagram alir pengambilan dan pengiriman data dapat dilihat pada gambar 3.7.



Gambar 3.7 Diagram Alir Pengambilan Data

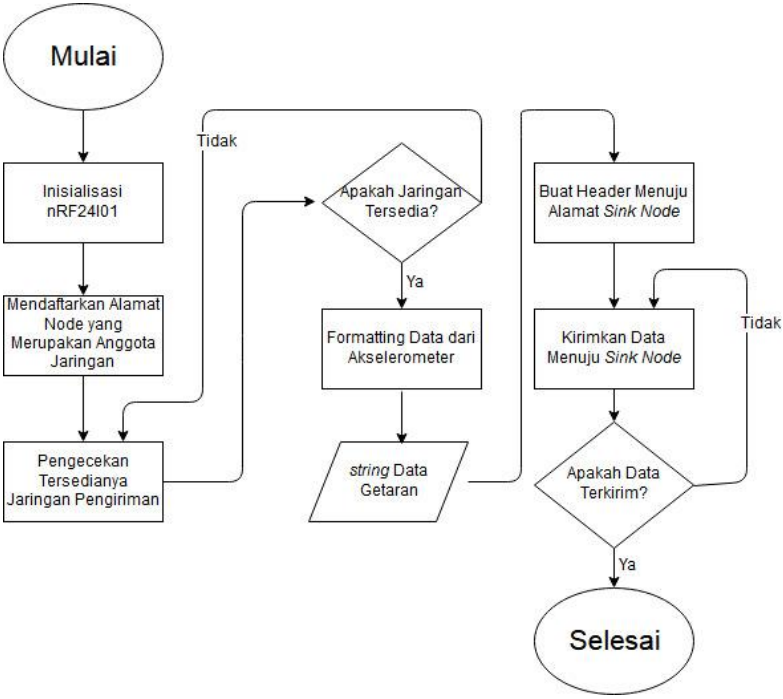
3.6.3 Diagram Alir Protokol Pengiriman Data

Protokol pengiriman data oleh *node* pemantau menuju *sink node* akan dijelaskan pada subbab ini. Data yang telah diambil oleh sensor akselerometer akan dikirim dengan protokol yang tidak memiliki *delay* antara pengiriman datanya.

Hal ini dilakukan untuk mendapatkan nilai *sampling rate* maksimal dikarenakan pengamatan nilai *eigen frequency*

membutuhkan *sampling rate* setidaknya dua kali dari nilai maksimal *eigen frequency* struktur.[17]

Diagram alir protokol pengiriman data dapat dilihat pada gambar 3.8.



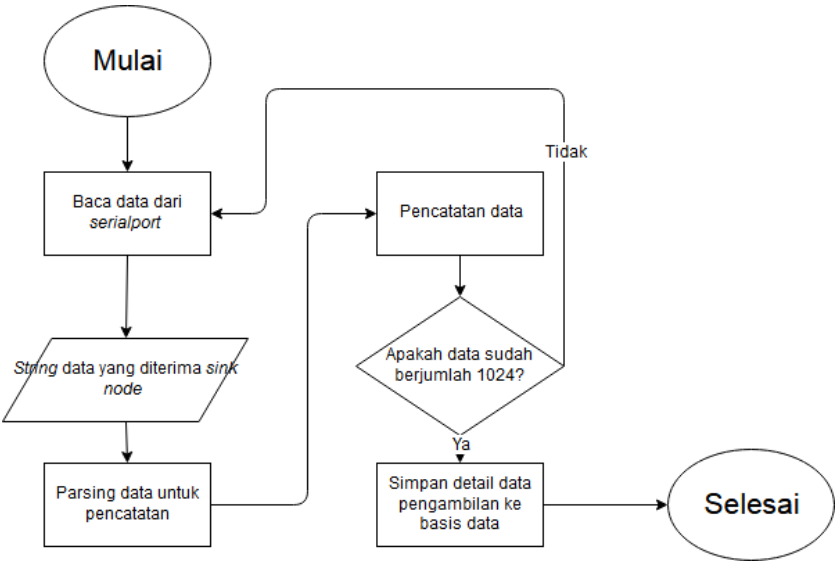
Gambar 3.8 Diagram Alir Protokol Pengiriman pada *Node* Pemantau

3.6.4 Diagram Alir Pencatatan dan Penyimpanan Data

Pencatatan data dan penyimpanan data adalah tahap lanjutan setelah *sink node* menerima data dari *node* pemantau. Setelah data diterima *sink node* sistem akan mencatat data yang masuk ke *sink node* ke dalam berkas berformat *.txt*.

Data yang disimpan dalam satu berkas berjumlah 1024 data yang merupakan *window size* dari penelitian ini. *Window* adalah interval data yang akan diolah dalam satu proses pengolahan. Data yang disimpan dalam berkas inilah yang akan diolah dengan algoritma FFT untuk proses ekstraksi nilai *eigen frequency* struktur.

Setelah data yang dicatat berjumlah 1024, detail pengambilan data dimasukkan ke dalam basis data untuk kemudahan akses. Diagram alir pencatatan dan penyimpanan data dapat dilihat pada gambar 3.9.



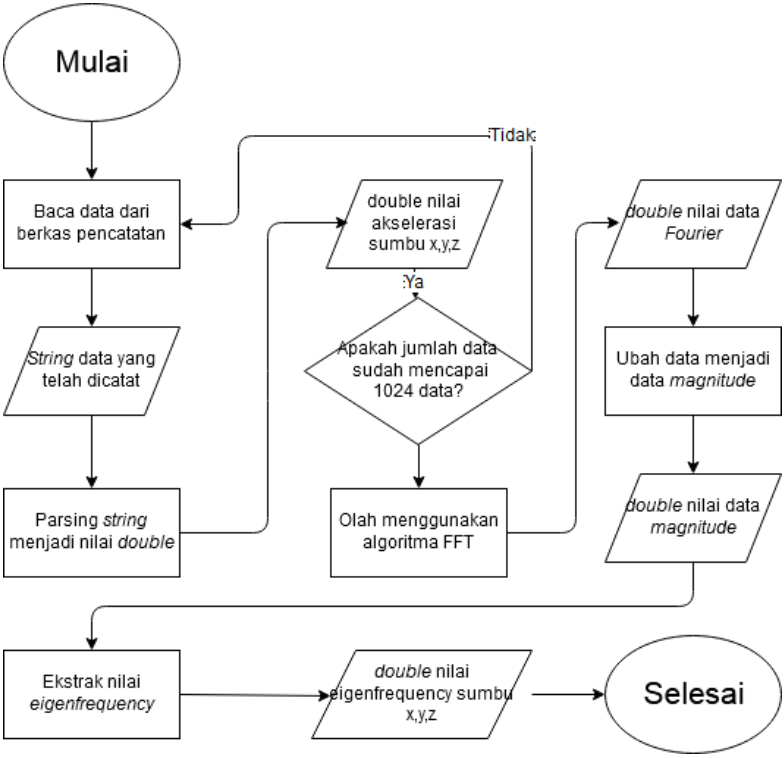
Gambar 3.9 Diagram Alir Pencatatan dan Penyimpanan Data

3.6.5 Diagram Alir Ekstraksi *Eigen frequency*

Setelah data yang dibutuhkan telah siap untuk diolah, tahap pengolahan data dilakukan. Seperti yang dijelaskan di atas, tahap pengolahan data memiliki *window size* dimana jumlah data yang

diolah dalam satu proses pengolahan ditentukan sebelumnya yaitu sebanyak 1024 data. 1024 data ini diperlukan untuk pengolahan menggunakan algoritma FFT, bila ada data yang rusak, hasil pengolahan dengan algoritma FFT akan gagal.

Tahap pengolahan data memiliki beberapa proses, yaitu; pengolahan dengan FFT, mengubah data FFT menjadi bentuk data *magnitude*, dan ekstraksi *eigen frequency*. Diagram alir ekstraksi *eigen frequency* dapat dilihat pada gambar 3.10.



Gambar 3.10 Gambar Diagram Alir Ekstraksi Nilai *Eigen frequency*

3.7 Struktur Data yang Digunakan pada Pengiriman Data Getaran

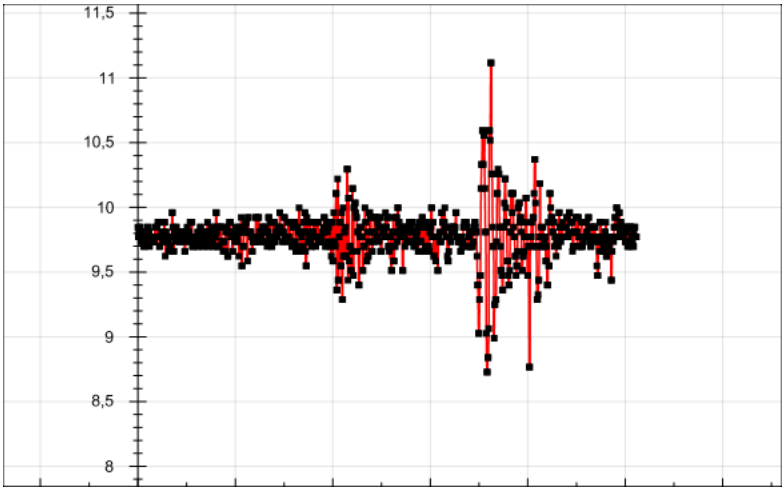
Tipe data yang digunakan pada pengiriman data oleh node pemantau adalah tipe data *string* (*array* dari banyak *char*). Tipe data ini digunakan karena tipe data ini memiliki efisiensi dalam ukuran dibandingkan tipe data lain yang dapat digunakan untuk pengiriman data dengan jumlah data jamak.

Tipe data lain yang mendukung pengiriman data jamak adalah tipe data *struct* yang dapat mengirimkan jumlah data jamak sesuai keinginan pengguna, berbeda dengan tipe data primitif seperti *char*. Akan tetapi ukuran data yang digunakan melebih tipe data primitif sehingga tidak cocok untuk digunakan dalam protokol yang membutuhkan nilai *sampling rate* tinggi.[18]

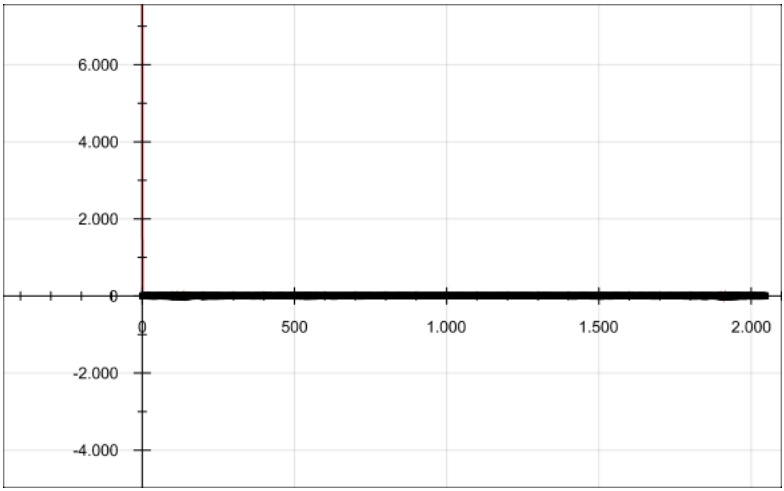
3.8 Pengolahan Data dengan Fast Fourier Transform

Data akselerasi yang telah dicatat diolah dengan algoritma *Fast Fourier Transform* untuk didapatkan nilai *Discrete Fourier Transform* (DFT)-nya. Data yang telah diolah menjadi DFT menjadi representasi bentuk *frequency-domain* dari data asalnya.

Contoh perubahan data dari data akselerasi menjadi data Discrete Fourier Transform dapat dilihat pada gambar 3.11 dan gambar 3.12. Dapat dilihat bila data yang telah diubah memiliki jumlah indeks dua kali lipat dari data asal. Data inilah yang akan diubah menjadi data *magnitude* untuk dicari *eigen frequency*-nya.[19]



Gambar 3.11 Contoh Data Akselerasi yang Dicatat



Gambar 3.12 Contoh Data Akselerasi yang Telah Diubah Menjadi *Discrete Fourier Transform*

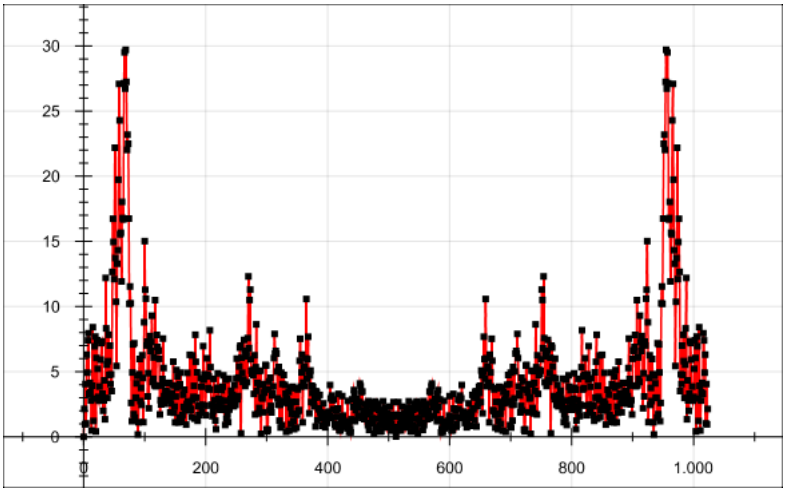
3.9 Konversi ke *Magnitude*

Proses konversi data dari bentuk DFT (*Discrete Fourier Transform*) ke bentuk *magnitude* dilakukan menggunakan persamaan yang dapat dilihat pada persamaan 3.1.[20]

$$M(i) = \sqrt{fft(2 * i)^2 + fft(2 * i + 1)^2} \quad \text{..... (3.1)}$$

Di mana:
M(i) = *Magnitude* pada indeks i
Fft(i) = Nilai *Fourier* pada indeks i

Magnitude atau dapat disebut amplitudo terbesar akan dicari frekuensinya untuk mendapatkan *eigen frequency* struktur. Gambar contoh data DFT yang telah diubah menjadi data *magnitude* dapat dilihat pada gambar 3.13.



Gambar 3.13 Contoh Data *Magnitude*

3.10 Pencarian Frekuensi

Frekuensi dari suatu gelombang dapat dicari dengan persamaan yang dapat dilihat pada persamaan 3.2[20]:

$$F = i * Sr/n \quad \text{..... (3.2)}$$

Di mana:

F = Nilai frekuensi
i = Indeks data pada data *magnitude*
Sr = Rata-rata jumlah pengambilan *sample* per detik
n = Jumlah data *sample*

Frekuensi dari nilai *magnitude* terbesar akan diambil sebagai *eigen frequency* struktur.

3.11 Penentuan *Threshold*

Penentuan *threshold* untuk menentukan peringatan perubahan nilai *eigen frequency* dilakukan dengan membandingkan hasil uji coba pada struktur yang dianggap normal dengan struktur yang dianggap tidak normal. Nilai yang terdekat antara data dari struktur yang dianggap normal dan tidak normal diambil nilai tengahnya dan nilai tersebut menjadi nilai *threshold node* lokasi uji coba. Nilai yang telah ditentukan akan dimasukkan secara *manual* pada antarmuka sistem *monitoring* struktur.

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Cakupan implementasi dari perancangan ini meliputi perangkat *node* dalam mengambil data dan mengirimkan data, pencatatan dan penyimpanan data, pengolahan data untuk mnemukan nilai *eigen frequency* dan implementasi antarmuka.

4.1 Lingkungan Implementasi

Implementasi sistem monitoring struktur menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada tabel 4.1.

Tabel 4.1 Tabel Perangkat Pengembangan

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz (8 CPUs), ~2.3GHz
	Memori	8192MB RAM
	Arduino Uno	2kB memori SRAM
	nRF24l01	2Mbps RF transceiver IC for the 2.4GHz ISM (Industrial, Scientific and Medical)
	ADXL-345	Sensitifitas ±2G
Perangkat Lunak	Sistem Operasi	Windows 8.1
	Perangkat Pengembang	NetBeans 8.0.2 dan Arduino IDE 1.6.11

4.2 Implementasi

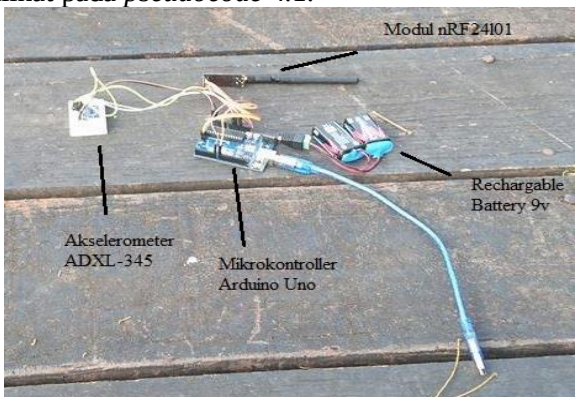
Pada sub bab implementasi ini menjelaskan mengenai pembangunan perangkat lunak secara detail dan menampilkan *pseudocode* yang digunakan mulai dari pengambilan data getaran dengan sensor ADXL-345 hingga proses ekstraksi nilai *eigen frequency*.

4.2.1 Mikrokontroller Arduino

Pada sub bab ini akan dijelaskan pengimplemntasian kode pada perangkat Mikrokontroller Arduino. Mikrokontroller Arduino menjadi unit pemroses data pada *node* pemantauan atau *node* pengambil data juga pada *sink node.*

4.2.1.1 *Node* Pengambil Data

Pada *node* pengambil data (lihat gambar 4.1), Arduino akan membaca data getaran dari modul akselerometer dan mengolah data ke format pengiriman data. Selanjutnnya data yang telah diambil dikirim oleh mikrokontroller Arduino melalui modul *transreceiver* nRF24l01. Kode sumber dari *node* pengambil data dapat dilihat pada *pseudocode* 4.1.



Gambar 4.1 *Node* Pengambilan Data

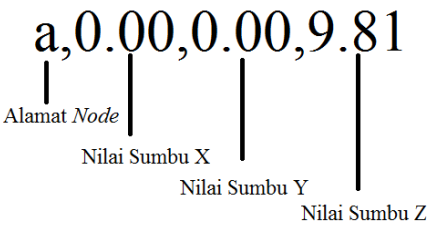
```
1   Function Setup:
2       Initialize ADXL_345(2G)
3       Initialize Radio
4       Set Radio Address
5   End Function
6
7   Function loop:
8       Initialize double x,y,z
9       Set x as x-axis value from accelerometer
10      Set y as y-axis value from accelerometer
11      Set z as z-axis value from accelerometer
12      Initialize double ax,ay,az
13      Set ax as acceleration value of x
14      Set ay as acceleration value of y
15      Set az as acceleration value of z
16      Initialize String send
17          Set send as node address, ax,ay,az
18      While kirim is not sent
19         Send kirim
20  End Function
```

Keluaran: Data dikirim sesuai format yang ditentukan

***Pseudocode* 4.1 Program pada *Node* Pengambil Data**

Proses pengiriman data dilakukan tanpa menyimpan data yang akan dikirim dikarenakan keterbatasan memori pada modul Arduino. Nilai *eigen frequency* akan dipengaruhi kecepatan pengiriman dari *node* pengambil data karena *sample rate* merupakan faktor pada nilai frekuensi yang dihasilkan.

Data yang dikirim adalah nilai akselerasi setiap sumbu dan alamat *node*. Format data yang dikirim dibuat untuk memudahkan proses *parsing* juga untuk meningkatkan efisiensi pengiriman data oleh *node* pengambil data. Contoh format pengiriman data adalah dapat dilihat pada gambar 4.2.



Gambar 4.2 Format Pengiriman Data

4.2.1.2 Sink Node

Sink node adalah perangkat penerima data dari *node* pengambil data. *Node* ini menerima data getaran (akselerasi) secara konstan dari tiga *node* pengambil data. Hal ini dikarenakan perangkat *node* pengambilan data tidak memiliki media penyimpanan data, sehingga data akselerasi yang telah diambil harus segera dikirim. *Pseudocode* implementasi program *sink node* dapat dilihat pada *pseudocode* 4.2.

```
1   Function Setup:
2       Initialize radio
3       Set radio address
4       Set all radio address of sampling node
5   End Function
6
7   Function loop:
8        Initialize array of char x with specified
9        size
10       Read data from radio and write to array x
11       Print x
12  End Function
```

Keluaran: Data di-*print* sesuai yang diterima *sink node*

Pseudocode* 4.2 Program Pada *Sink Node

Data yang di-*print* akan dibaca oleh program melalui *serial port*. Data pada *serial port* akan terbaca sebagai data berjenis string. Data inilah yang akan diolah untuk pencarian nilai *eigen frequency*.

4.2.2 Komputer

Pada bagian komputer, tahapan proses yang dilakukan adalah sebagai berikut:

1. Pembacaan data dari *sink node*
2. Penyimpanan detail data pada basis data MySQL
3. *Parsing* data untuk pengolahan data
4. Proses pengolahan data menggunakan algoritma FFT
5. Pengubahan nilai DFT(*Discrete Fourier Transform*) dari data akselerasi menjadi nilai *magnitude*
6. Pencarian nilai *eigen frequency*
7. Penentuan *threshold* nilai *eigen frequency* untuk sistem peringatan
8. Implementasi pada antarmuka

4.2.2.1 Pembacaan Data dari *Node Sink*

Proses pembacaan data dari *sink node* diawali dengan inisialisasi untuk pembacaan *Serial Port* di mana *sink node* tersambung dengan komputer melalui koneksi USB. *Pseudocode* program mendeteksi *sink node* dapat dilihat pada *pseudocode* 4.3.

```
1   Function initialize(InterfaceTA ta)
2          Initialize PortId:=NIL
3          Initialize  PORT_NAMES  as  array  of  port
4          acknowledged
5          Initialize    enumeration    portEnum    as
6          CommportIdentifier in system
7          while (portEnum.hasMoreElements())
8           initialize currPortId:=(CommPortIdentifier)
9          portEnum.nextElement()
10
11         for each String portName in PORT_NAMES
12         if (currPortId.getName().equals(portName))
13  currPortId
14
15 if (portId == NIL)
16 InterfaceTA write "Could not find COM port"
17 End for
18 End while
19 Listen to PortId
20
21 ENDFUNCTION
```

Pseudocode* 4.3 Fungsi Inisialisasi Pembacaan *Serial Port

Langkah pertama pada *pseudocode* 4.3 adalah pencarian *sink node* yang tersambung pada komputer. Apabila program tidak dapat menemukan *sink node* yang tersambung pada komputer program akan menulis "Could not find COM port." pada konsol netbeans.

Langkah berikutnya adalah pembacaan data dari serial port untuk dicatat ke dalam berkas .txt. Tahapan ini mencakup pembuatan berkas .txt untuk pencatatan, *parsing* untuk pencatatan data sesuai asal *node* data. *Pseudocode* implementasi program pencatatan data dapat dilihat pada *pseudocode* 4.4.

```
1   Function SerialEvent(read Port)
2    While(data available)
3          If first char in data is a
4             If data in FileA is less than 1024
5                Write data in FileA
6             Else
7                Update database
8                Make new FileA
9                Write data in FileA
10            Endif
11         Elsif first char in data is b
12            If data in FileB is less than 1024
13               Write data in FileB
14            Else
15               Update database
16               Make new FileB
17               Write data in FileB
18            Endif
19         Else
20            If data in FileC is less than 1024
21               Write data in FileC
22            Else
23               Update database
24               Make new FileC
25               Write data in FileC
26            Endif
27         Endif
28  EndWhile
    ENDFUNCTION
```

***Pseudocode* 4.4 Fungsi Menulis Data Getaran ke Dalam Berkas**

Pencatatan data diawali dengan pengecekan adanya data pada *port* yang dibaca, ketika data tersedia, data dibaca sebagai variabel *inputLine*. *InputLine* kemudian dites kualitasnya dengan fungsi *checkline()* untuk memastikan data tidak rusak.

Pengecekan kualitas data ditulis pada fungsi *checkline*() yang *pseudocode*-nya dapat dilihat pada *pseudocode* 4.5. Fungsi ini memastikan hasil parsing dapat diubah menjadi data jenis *double.*

Data yang lolos kemudian dicek alamat asalnya yaitu karakter paling awal dari data tersebut. Data akan dicatat sesuai *node* asalnya.

Jumlah data dihitung menggunakan variabel *counterA*, *counterB* dan *counterC*. Ketika proses dimulai semua variabel tersebut bernilai nol. Ketika data pertama masuk, berkas untuk pencatatan akan dibuat. Data yang dicatat akan dihitung setiap pencatatan agar tidak melebihi *window* yang telah ditentukan.

Ketika data sudah mencukupi *window* variabel penghitungan data di-*reset* menjadi nol kembali, sehingga ketika ada data masuk akan dibuat berkas baru untuk pencatatan. Karena pengambilan data juga berlangsung bersamaan dengan pencatatan data, penghitungan nilai *sampling rate* juga dilakukan pada tahap ini.

Variabel penyimpanan nilai *sampling rate* adalah variabel *FA*, *FB*, *FC*. Detail pengambilan data (*node* asal, waktu memasukkan detail data, nama berkas data pencatatan data dan *sampling rate* pencatatan data) yang telah dilakukan, dimasukkan ke dalam basis data yang telah dibuat sebelumnya.

```
1   Function checkline(String line)
2   Initialize count := 0, countz := 0, jarak := 0
3   String datarawx := "", datarawy := "", datarawz
4   := ""
5   data[] := [0..2]
6   try
7   for i := 0 loop till i < line.length() by i++
8   each step
9         if (line.charAt(i) == ',')
10        count++
11        else if (count == 1)
12        datarawx := datarawx + line.charAt(i)
13        else if (count == 2)
14        datarawy := datarawy + line.charAt(i)
15        else if (count == 3)
16        if (line.charAt(i) == '.')
17        countz := i
18        jarak := countz + 3
19        if (countz > 0 AND countz <= jarak)
20        datarawz := datarawz + line.charAt(i)
21        countz++
22        else if (countz == 0)
23        datarawz := datarawz + line.charAt(i)
24  data[0] := Double.valueOf(datarawx)
25  data[1] := Double.valueOf(datarawy)
26  data[2] := Double.valueOf(datarawz)
27  catch (Exception e)
28  return false
29  return true

31  ENDFUNCTION
```

***Pseudocode* 4.5 Fungsi Pengecekan Kualitas Data**

Semua fungsi tersebut merupakan fungsi dalam kelas SerialArduino.java. Kelas ini dibuat khusus fungsi pengambilan data dari *port* tempat *sink node* terhubung pada komputer.

4.2.2.2 Penyimpanan Detail Data Pengambilan pada Basis Data MySQL

Pada subbab sebelumnya telah disebutkan jika penyimpanan detail pengambilan data dilakukan setiap terkumpulnya data sejumlah window yang telah ditentukan. Pada subbab ini akan dijelaskan detail fungsi yang dipanggil untuk menyimpan data pada basis data MySQL pada sistem monitoring yang dibangun pada tugas akhir ini.

Pada awal aplikasi dibuka, aplikasi akan membuat sambungan pada server basis data MySQL. Fungsi membuat sambungan tersebut dinamakan *ConInit*(). *Pseudocode ConInit*() dapat dilhat pada *pseudocode* 4.6.

```
1   FUNCTION ConInit()
2   Initialize con as connection to database
3   Initialize stmt as statement to database
4   Initialize status as connection status aof
    database
5           return status
    ENDFUNCTION
```

***Pseudocode* 4.6 Fungsi Pembuatan Sambungan pada Basis Data**

Pada *Pseudocode* 4.6 aplikasi akan membuat sambungan dengan basis data MySQL (baris 5) dan akan memastikan sambungan tersebut telah dibuat dengan variabel status (baris 6). Fungsi ini akan mengembalikan nilai status sambungan bertipe data *boolean* (baris 7).

Setelah sambungan dibuat, data detail pencatatan data getaran dapat dimasukkan ke dalam basis data. Fungsi memasukkan data tersebut dinamakan InsertToDB(). Pseudoceode InsertToDB() dapat dilihat pada *pseudocode* 4.7.

```
1   FUNCTION InsertToDB(String node, String filename,
2   Timestamp date, fs)
3   Execute sql query ("INSERT INTO file
4   (node,date,filename,sr) VALUES ('" + node + "','"
    + date.toString() + "','" + filename + "','" +
    String.valueOf(fs) + "')")
    ENDFUNCTION
```

***Pseudocode* 4.7 Fungsi Memasukkan Data pada Basis Data**

Perintah SQL akan dieksekusi dengan memanggil fungsi *execute*. Fungsi *execute* akan mengeksekusi perintah SQL dalam bentuk data *string* yang dijadikan parameter pada fungsi tersebut.

Data yang telah masuk pada basis data dapat diambil untuk menampilkan data pada antarmuka aplikasi juga untuk melakukan pengolahan data lebih lanjut. Fungsi memanggil data ini akan dibagi menjadi tiga fungsi.

Fungsi pertama adalah fungsi mengambil data waktu pencatatan data getaran dilakukan. Data waktu berguna untuk menunjukkan perubahan pada nilai *eigen frequency* seiring waktu. Fungsi mengambil data waktu disebut *GetTimes*(). *Pseudocode* fungsi *GetTimes*() dapat dilihat pada *pseudocode* 4.8.

```
1   FUNCTION GetTimes(String node)
2           Execute query("SELECT * FROM file WHERE
3   node='" + node + "'")
4           Initialize rowcount := number of row in
5   query result
6           Initialize times := String[0..rowcount-
7   1]
8           rowcount := 0
9           while (Iterate query result)
10              times[rowcount] := "date" in each
11  result row
12              rowcount++
13          return times
    ENDFUNCTION
```

***Pseudocode* 4.8 Fungsi Pengambilan Data Waktu Pencatatan Data Getaran**

Fungsi ini akan mengembalikan semua data waktu pengambilan data dari *node* tertentu. *Node* yang ingin diamati dapat ditentukan pada antarmuka aplikasi. Fungsi ini akan digunakan ketika pengguna mengganti *node* yang ingin diamati. Karena pengambilan data dilakukan secara *live*, setiap *node* memiliki waktu pencatatan berbeda-beda. Pencatatan waktu juga memudahkan pembedaan data yang diambil sebelumnya dengan data yang baru saja dicatat.

Untuk menampilkan data getaran pada antarmuka, dibutuhkan detail data yng ingin ditampilkan, detail data tersebut adalah *node* yang ingin diamati dan waktu pengambilan data. Setelah data tersebut ditentukan di antarmuka, nama berkas pencatatan data getaran dapat dipanggil pada basis data. Fungsi pengambilan nama berkas tersebut dinamakan *GetFiles*(). *Pseudocode GetFiles*() dapat dilihat pada *pseudocode* 4.9.

```
1   FUNCTION GetFiles(String node, Timestamp ts)
2           Initialize Filename := ""
3           Execute query("SELECT * FROM file where
4   node='" + node + "' AND date='" + ts.toString()
5   + "'")
6           while (Iterating query result)
7               Filename := "filename" attribute in
8   query result
9           return Filename
    ENDFUNCTION
```

***Pseudocode* 4.9 Fungsi Pengambilan Nama Berkas Data Getaran pada Basis Data**

Fungsi ini akan mendapatkan alamat berkas data getaran sesuai *node* dan waktu pengambilan yang ingin dilihat pengguna. Setelah berkas data getaran ditentukan, dimulailah pengolahan data untuk pencarian nilai *eigen frequency*.

Seperti yang telah dijelaskan pada bab 3, pencarian nilai *eigen frequency* ini memerlukan nilai *sampling rate* dalam rumus penentuannya. Nilai *sampling rate* ditentukan setiap jumlah data

getaran mencapai *window* yang telah ditentukan. Nilai sampling rate ini disimpan pada basis data untuk proses perhitungan tersebut.

Fungsi untuk mengambil nilai *sampling rate* pada basis data dinamakan fungsi GetFS(). Fungsi ini akan mengembalikan nilai *double* yang merupakan nilai *sampling rate* dari pencatatan data getaran. *Pseudocode* fungsi GetFS() dapat dilihat pada *pseudocode* 4.10.

```
1   FUNCTION GetFs(String Filename)
2           Initialize fstemp := ""
3           Execute query("SELECT * FROM file where
4   filename='" + Filename + "'")
5           while (iterating query result)
6               fstemp := "sr" attribute of query
7   result
8           Initialize fs := Double.valueOf(fstemp)
9           return fs
    ENDFUNCTION
```

***Pseudocode* 4.10 Fungsi Pengambilan Nilai *Sampling Rate* pada Basis Data**

Semua fungsi yang disebutkan pada subbab ini merupakan fungsi dalam kelas Database.java.

4.2.2.3 Parsing Data Untuk Pengolahan Data

Setelah data yang ingin diamati ditentukan pada antarmuka, berkas data getaran akan dibaca untuk di-*parsing* menjadi nilai yang dapat diolah dan ditentukan nilai *eigen frequency*-nya.

Fungsi ini dinamakan fungsi *readdata*(). Fungsi ini akan mengembalikan nilai getaran pada sumbu x, sumbu y dan sumbu z sejumlah *window* data yang ditentukan. Fungsi ini merupakan salah satu fungsi pada kelas Proses.java. *Pseudocode* fungsi readdata() dapat dilihat pada *pseudocode* 4.11

```
1   FUNCTION readdata(String file) throws
2   FileNotFoundException, IOException
    Initialize data := [0..2][0..1023]
3   Initialize line
4   Initialize datarawz := "", datarawy := "",
    datarawx := ""
5   Initialize count
    Initialize counter := 0
6   Initialize countz, jarak
7   while ((read line from file AND counter < 1024)
8   count := 0
9   countz := 0
10  jarak := 0
11  for i := 0 loop till i < line.length() by i++ each
12  step
13  if (line.charAt(i) == ',')
14  count++
15  else if (count == 1)
16  datarawx := datarawx + line.charAt(i)
17  else if (count == 2)
18  datarawy := datarawy + line.charAt(i)
19  else if (count == 3)
20  if (line.charAt(i) == '.')
21  countz := i
22  jarak := countz + 3
23  if (countz > 0 AND countz <= jarak)
24  datarawz := datarawz + line.charAt(i)
25  countz++
26  else if (countz == 0)
27  datarawz := datarawz + line.charAt(i)
28  data[0][counter] := Double.valueOf(datarawx)
29  data[1][counter] := Double.valueOf(datarawy)
30  data[2][counter] := Double.valueOf(datarawz)
31  datarawz := datarawy := datarawx := ""
32  counter++
33  return data
34  ENDFUNCTION
```

***Pseudocode* 4.11 Fungsi Parsing Data Getaran pada Berkas Pencatatan Data**

4.2.2.4 Proses Pengolahan Data Menggunakan Algoritma FFT

Data yang telah di-*parsing* selanjutnya diolah menggunakan algoritma FFT untuk dicari transformasi DFT-nya. Implementasi penggunaan algoritma FFT pada tugas akhir ini menggunakan *library* JTransform yang detailnya dapat dilihat pada bab 2.4.

Fungsi transformasi data getaran menjadi data DFT dinamakan fungsi *FindFFT*(). Fungsi ini termasuk fungsi dalam kelas Proses.java. *Pseudocode FindFFT*() dapat dilhat pada *pseudocode* 4.12

```
1   FUNCTION FindFFT([][] raw)
2           Initialize datatotal := raw[0].length
3           Initialize [][] fft :=
4   [0..2][0..datatotal * 2-1]
5           for i := 0 loop till i < 3 by i++ each
6   step
                Copy array raw [i] to fft[i]
7               Transform data to in array fft[i] to
8   DFT  transformation
9           return fft
10  ENDFUNCTION
```

***Pseudocode* 4.12 Implementasi Algoritma FFT Menggunakan *library* JTransform**

Data getaran yang ingin ditransformasi sebelumnya dipindah kedalam *array* yang berukuran dua kali lipat dari jumlah getaran yang ingin diolah. Hal ini dikarenakan transformasi ke dalam bentuk DFT akan men-generate nilai imaginary dari nilai getaran awal dan mengubah data dari yang berbentuk *time-domain* menjadi *frequency-domain.*

4.2.2.5 Pengubahan Nilai DFT(*Discrete Fourier Transform*) Dari Data Akselerasi Menjadi Nilai *Magnitude*

Nilai data yang telah diubah menjadi DFT akan diubah lagi menjadi nilai *magnitude* di mana nilai *eigen frequency* dapat ditemukan. Pengubahan nilai DFT menjadi nilai *magnitude* dapat dilakukan menggunakan rumus yang dapat dilihat pada bab 3.8.

Fungsi mengubah nilai *magnitude* dinamakan fungsi *FindMagnitude*(). *Pseudocode* dari fungsi ini dapat dilihat pada *pseudocode* 4.13.

```
1   FUNCTION  FindMagnitude(fft[][])
2           Initialize re, im
3           Initialize numberofdata := fft[0].length
4   / 2
5           initialize          magnitude          :=
6   [0..2][0..numberofdata-1]
            for x := 0 loop till x < 3 by x++ each
7   step
8           for i := 1 loop till i < numberofdata by
9   i++ each step
10                  re := fft[x][2 * i]
11                  im := fft[x][2 * i + 1]
                    magnitude[x][i] := sqrt(re * re +
12  im * im)
            return magnitude
    ENDFUNCTION
```

Pseudocode* 4.13 Fungsi Mengubah Nilai DFT Menjadi Nilai *Magnitude

4.2.2.6 Pencarian Nilai *Eigen frequency*

Pencarian nilai *eigen frequency* dilakukan pada data yang telah diubah menjadi bentuk *magnitude*-nya. *Eigen frequency* adalah nilai frekuensi dari puncak pertama pada data *magnitude*. Nilai frekuensi ditentukan dengan rumus yang dapat dilihat pada bab 3.9.

Fungsi pencarian nilai *eigen frequency* dinamakan fungsi *FindEF*(). *Pseudocode FindEF*() dapat dilihat pada *pseudocode* 4.14.

```
1   FUNCTION  FindEF([][] magnitude, sr)
2           Initialize  EF := [0..2]
3           Initialize numberofdata :=
4   magnitude[0].length
5           Initialize Sr := sr, max_magnitude := -1
6           Initialize max_position := [] { 0, 0, 0
    }
7           for x := 0 loop till x < 3 by x++ each
8   step
9               for i := 0 loop till i <
10  numberofdata by i++ each step
11                  if (magnitude[x][i] >
    max_magnitude)
12                      max_magnitude :=
13  magnitude[x][i]
14                      max_position[x] := i
                max_magnitude := -1
15              EF[x] := Sr * max_position[x] /
    numberofdata
16          return EF
    ENDFUNCTION
```

***Pseudocode* 4.14 *Pseudocode* Mencari Nilai *Eigen frequency* dari Nilai *Magnitude* Data**

Setelah puncak pertama dari data ditentukan, pengaplikasian rumus mencari frekuensi dilakukan tehadap data puncak dan program mengembalikan nilai frekuensi dari data puncak tersebut.

4.2.2.7 *Threshold* Nilai *Eigen frequency* untuk Sistem Peringatan

Penentuan nilai *threshold* pada aplikasi dilakukan dengan cara melihat hasil uji coba nilai *eigen frequency* struktur. Struktur yang dianggap baik dijadikan patokan sebagai nilai *eigen frequency* yang normal sedangkan struktur yang tidak normal dijadikan patokan sebagai *eigen frequency* struktur yang rusak.

Sistem menyediakan fungsi *input* batas atas dan batas bawah pada antarmukanya. Fungsi ini akan dibahas pada subbab implementasi selanjutnya.

4.2.2.8 Implementasi Antarmuka

Implementasi antarmuka akan dibagi menjadi beberapa bagian. Bagian-bagian tersebut adalah antarmuka menampilkan grafik, antarmuka pemilihan data yang ditampilkan, antarmuka penentuan *threshold.*

1. Menampilkan Grafik
   Sistem dapat menampilkan grafik data, baik itu data getaran, data konversi menggunakan FFT dan data *magnitude.* Sistem menggunakan *library* GRAL untuk menampilkan grafik pada antarmukanya.
2. Pemilihan Data yang Ditampilkan
   Data yang ingin ditampilkan dapat dipilih pada sistem antarmuka. Detail yang ditentukan akan langsung diproses setiap pemilihan dilakukan tanpa membutuhkan perintah tambahan dari pengguna.
3. Penentuan *Threshold*
   Penentuan *Threshold* dilakukan dengan memasukkan nilai batas atas dan batas bawah secara manual. Antarmuka menyediakan dialog box yang akan muncul ketika tombol set limit ditekan.

Tampilan antarmuka dapat dilihat pada uji coba fungsional yang didokumentasikan pada bab berikutnya.

BAB V
HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai pengujian dan evaluasi dari sistem monitoring struktur yang telah dikembangkan. Sistem akan diuji coba baik secara fungsionalitas maupun performa menggunakan skenario yang telah ditentukan.

Pengujian fungsionalitas meliputi setiap bagian sistem yang telah dikembangkan baik pada bagian *node*, maupun unit pengolah data. Pengujian performa meliputi hasil olahan data pada sistem *monitoring* struktur yang telah dibangun.

5.1 Uji Coba Fungsionalitas

Uji coba fungsionalitas dilakukan untuk memeriksa setiap bagian sistem baik perangkat keras maupun kode yang telah ditulis untuk mengetahui apakah setiap komponen sistem berjalan dengan semestinya

5.1.1 Lingkungan Uji Coba

Lingkungan pengujian pada uji coba fungsionalitas menggunakan spesifikasi keras dan perangkat lunak seperti yang ditunjukkan pada tabel 5.1.

Tabel 5.1 Spesifikasi Lingkungan Pengujian Fungsionalitas

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz (8 CPUs), ~2.3GHz
	Memori	8192MB RAM
	Arduino Uno	2kB memori SRAM

	nRF24l01	2Mbps RF transceiver IC for the 2.4GHz ISM (Industrial, Scientific and Medical)
	ADXL-345	Sensitifitas ±2G
Perangkat Lunak	Sistem Operasi	Windows 8.1
	Perangkat Pengembang	NetBeans 8.0.2 dan Arduino IDE 1.6.11

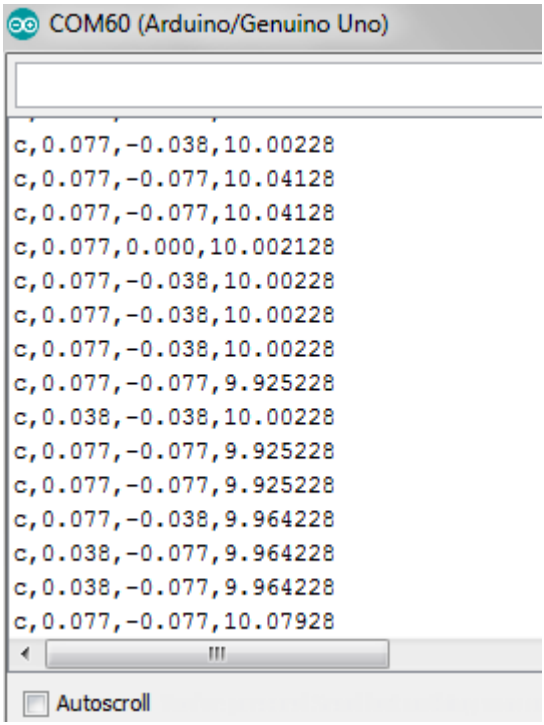
5.1.2 Uji Coba Pengiriman Data dari *Node* Pemantau ke *Node Sink*

Seperti yang dijelaskan pada bab 3 sistem ini diawali dari pengiriman data oleh *node* pemantau ke *node sink* untuk pengolahan lebih lanjut. Skenario uji coba pengiriman data dapat dilihat pada tabel 5.2.

Tabel 5.2 Skenario Uji Coba Fungsionalitas 1

Nama	Uji coba pengiriman data dari *node* pemantau ke *node sink*
Tujuan	Mengirimkan data
Skenario	*Node* pemantau mengirim data getaran ke *node sink*
Hasil Uji Coba	*Node sink* berhasil menerima data dari *node* pemantau

Pengiriman data pada *node* dapat diamati melalui salah satu fitur Arduino IDE. Pengiriman data yang diamati melalui Arduino IDE dapat dilihat pada gambar 5.1.



Gambar 5.1 Pengamatan Pengiriman Data pada *Node Sink* yang Tersambung pada Komputer Menggunakan Fitur Arduino IDE

5.1.3 Uji Coba Pencatatan Data Getaran pada Berkas .txt

Setelah data getaran diterima pada *node sink,* data kemudian dicatat pada berkas berformat .txt. Proses pencatatan dan pengiriman berlangsung secara bersamaan sehingga nilai sampling rate dipengaruhi oleh kecepatan pencatatan. Karena itulah sistem penyimpanan data pada tugas akhir ini tidak langsung ke dalam basis data tetapi pada berkas. Sistem juga akan secara otomatis mengganti berkas pencatatan ketika jumlah data yang dicatat mencapai nilai *window* data yang ditentukan (dalam tugas akhir ini

window pengamatan bernilai 1024 data). Skenario uji coba ini dapat dilihat pada tabel 5.3.

Tabel 5.3 Skenario Uji Coba Fungsionalitas 2

Nama	Uji coba pencatatan data getaran pada berkas berformat .txt
Tujuan	Mencatat data getaran yang terjadi pada struktur yang diamati
Skenario	Data yang dibaca dari *serial port* ditulis ke dalam berkas berformat .txt
Hasil Uji Coba	Data getaran berhasil dicatat ke dalam berkas berforrmat .txt

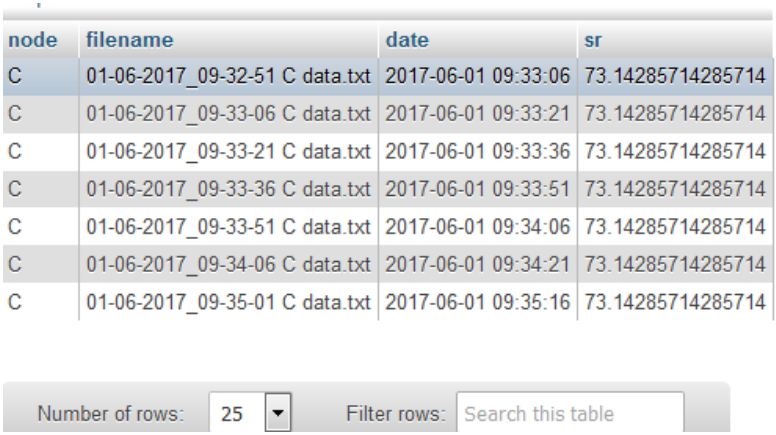
5.1.4 Uji Coba Penyimpanan Detail Pengambilan Data pada Sistem Basis Data MySQL

Setelah data dicatat ke dalam berkas berformat .txt, sistem kemudian menyimpan detail pengambilan data ke dalam sistem basis data MySQL. Proses ini dilakukan secara langsung ketika data yang tercatat ke dalam sebuah berkas mencapai nilai *window* yang telah ditentukan (1024 data). Skenario uji coba pada uji coba ini dapat dilihat pada tabel 5.4.

Tabel 5.4 Skenario Uji Coba Fungsional 3

Nama	Uji coba penyimpanan detail pengambilan data pada sistem basis data MySQL
Tujuan	Mencatat detail pengambilan data yang telah dilakukan sistem
Skenario	Detail data dimasukkan ke dalam sistem basis data MySQL
Hasil Uji Coba	Detail data berhasil masuk pada sistem basis data MySQL

Data yang telah masuk ke dalam sistem basis data MySQL dapat dilihat menggunakan fitur *phpmyadmin.* Hasil uji coba ini dapat dilihat pada gambar 5.2.



node	filename	date	sr
C	01-06-2017_09-32-51 C data.txt	2017-06-01 09:33:06	73.14285714285714
C	01-06-2017_09-33-06 C data.txt	2017-06-01 09:33:21	73.14285714285714
C	01-06-2017_09-33-21 C data.txt	2017-06-01 09:33:36	73.14285714285714
C	01-06-2017_09-33-36 C data.txt	2017-06-01 09:33:51	73.14285714285714
C	01-06-2017_09-33-51 C data.txt	2017-06-01 09:34:06	73.14285714285714
C	01-06-2017_09-34-06 C data.txt	2017-06-01 09:34:21	73.14285714285714
C	01-06-2017_09-35-01 C data.txt	2017-06-01 09:35:16	73.14285714285714

Gambar 5.2 Data yang Berhasil Masuk ke Dalam Sistem Basis Data MySQL Diamati Menggunakan *phpmyadmin*

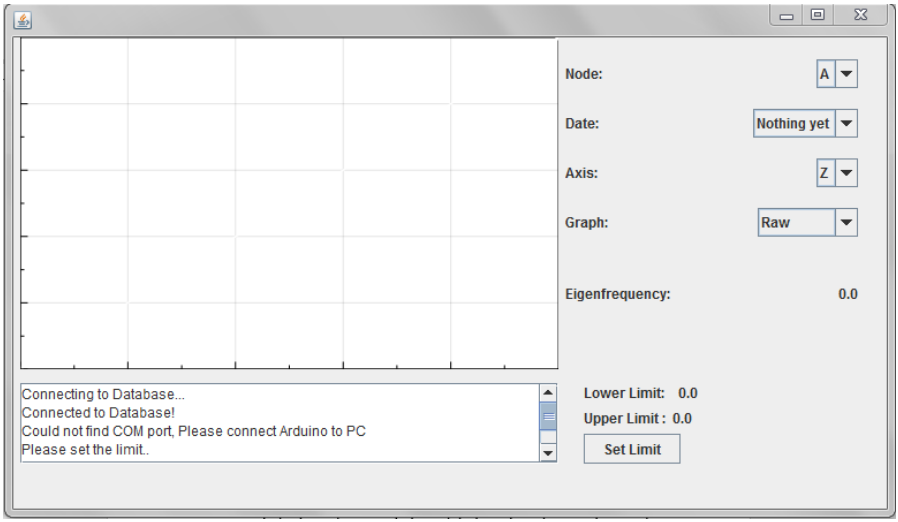
5.1.5 Uji Coba Menampilkan Data yang Telah Diolah

Data yang telah dicatat kemudian akan diolah menggunakan algoritma FFT untuk diubah menjadi bentuk DFT. Data DFT kemudian diolah lebih lanjut menjadi data *magnitude* dari getaran yang telah dicatat. Data-data tersebut kemudian ditampilkan pada antarmuka sistem. Uji coba ini akan menguji pengolahan data, penampilan data yang belum diolah dan sudah diolah dan hasil penghitungan *eigen frequency*. Skenario uji coba ini dapat dilihat pada tabel 5.5.

Tabel 5.5 Skenario Uji Coba Fungsional 4

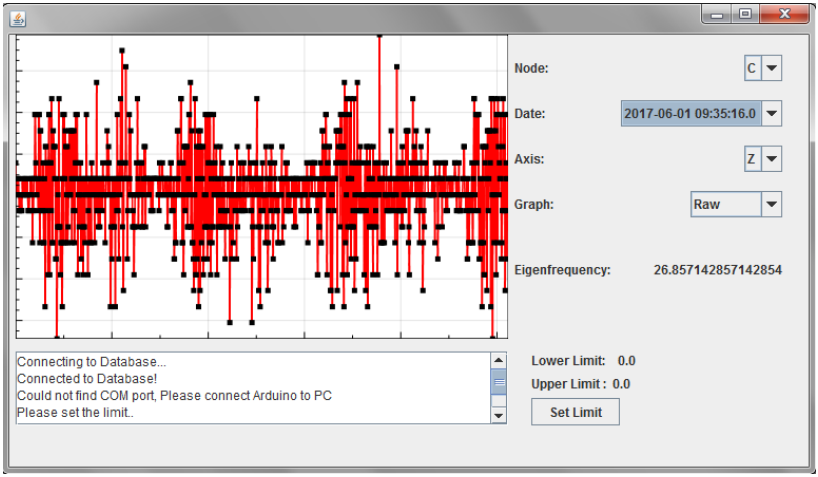
Nama	Uji coba menampilkan data yang telah diolah
Tujuan	Menampilkan data hasil olahan dan nilai *eigen frequency* yang berhasil didapatkan
Skenario	Membuka aplikasi awal sebelum ada data kemudian melakukan proses mengambil data
Hasil Uji Coba	Data berhasil ditampilkan dengan baik.

Sebelum adanya data yang tersedia, aplikasi akan memiliki penampilan yang sedikit berbeda dengan ketika data sudah tersedia. Penampilan aplikasi sebelum adanya data dapat dilihat pada gambar 5.3.



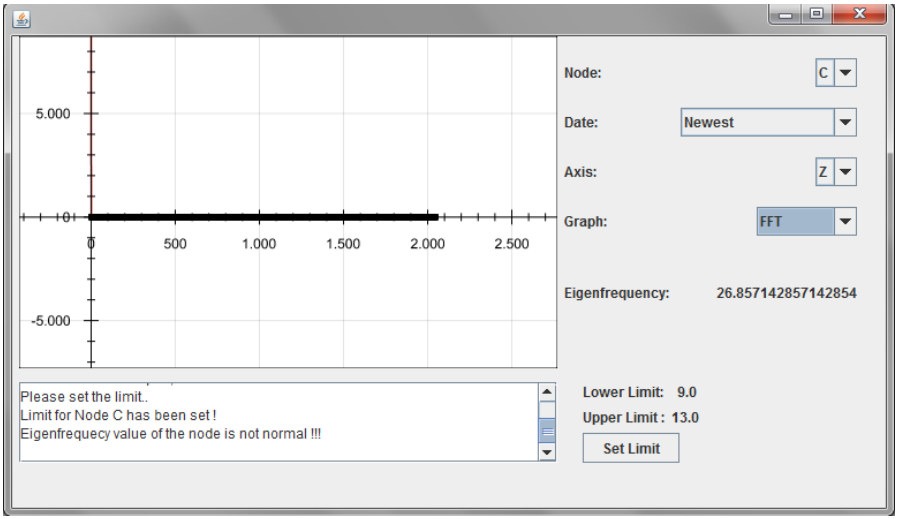
Gambar 5.3 Aplikasi Saat Belum Ada Data yang Masuk

Penampilan aplikasi setelah ada data dapat dilihat pada gambar 5.4.

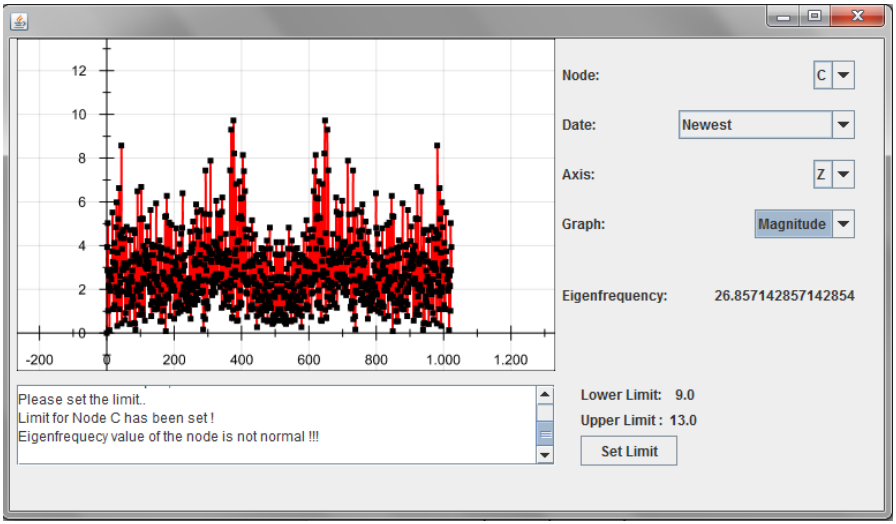


Gambar 5.4 Penampilan Aplikasi Setelah Ada Data yang Dapat Diolah (Data *Raw*)

Aplikasi dapat menampilkan data sesuai *axis* dan bentuk data yang diamati. Bentuk data yang dapat diamati adalah bentuk data mentah (data getaran yang telah dicatat), data transformasi DFT menggunakan algoritma FFT dan data berbentuk *magnitude*. Pilihan bentuk data dapat dilihat pada gambar 5.5 dan gambar 5.6.



Gambar 5.5 Penampilan Aplikasi Setelah Ada Data yang Dapat Diolah (Data DFT)



Gambar 5.6 Penampilan Aplikasi Setelah Ada Data yang Dapat Diolah (Data *Magnitude*)

5.1.6 Uji Coba Memasukkan *Threshold*

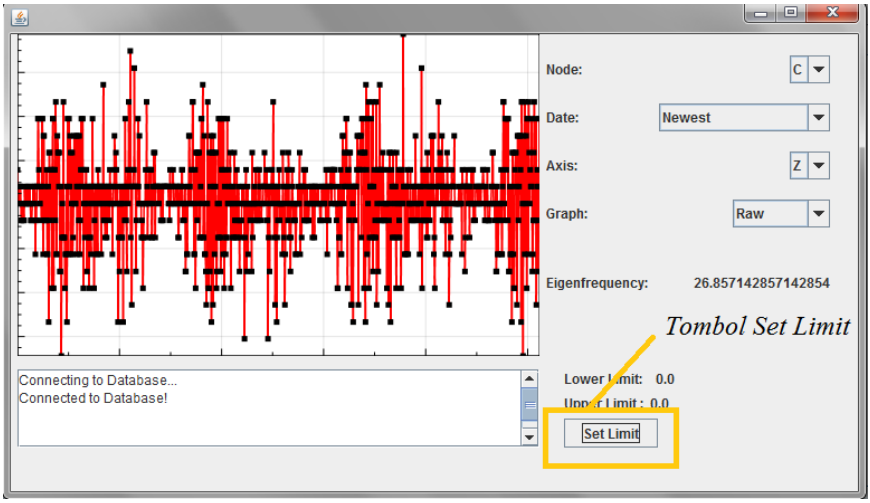
Proses memasukkan *threshold* dilakukan secara *manual* dengan menekan tombol *set limit* dan memasukkan batas atas dan batas bawah baru. Penentuan nilai *threshold* dijelaskan pada bab 3.10. Setiap *node* memiliki nilai batas yang berbeda, sehingga nilai batas yang dimasukkan akan sesuai dengan *node* yang dipilih pada antarmuka.

Karena *eigen frequency* yang diuji adalah *eigen frequency* dari sumbu z, peringatan hanya akan ditampilkan ketika nilai *eigen frequency* sumbu z tidak dalam batasan normal. Skenario uji coba ini dapat dilihat pada tabel 5.6.

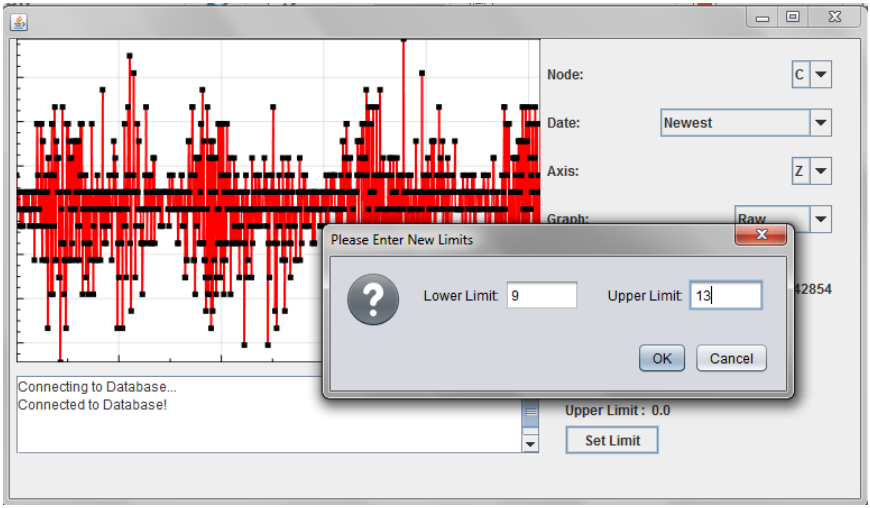
Tabel 5.6 Skenario Uji Coba Fungsional 5

Nama	Uji coba memasukkan *threshold*
Tujuan	Memasukkan batas bawah dan atas nilai *eigen frequency* yang dianggap normal dan sistem memberi peringatan ketika nilai *eigen frequency* tidak normal
Skenario	Menekan tombol set limit dan memasukkan nilai batas atas dan bawah nilai *eigen frequency* yang dianggap normal
Hasil Uji Coba	Proses memasukkan *threshold* dan sistem peringatan berjalan dengan baik.

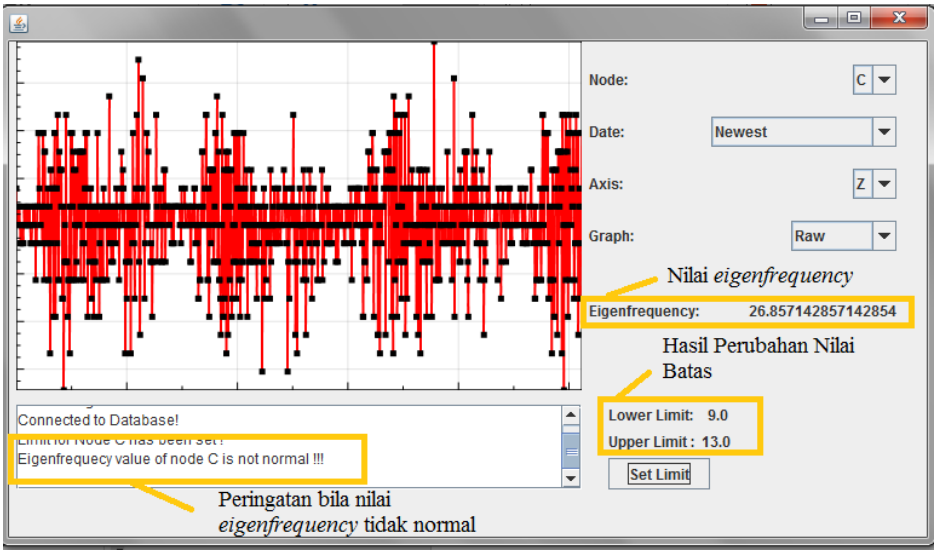
Tombol *set limit* dapar dilihat pada gambar 5.7. tombol tersebut akan menampilkan kotak dialog permintaan masukan nilai batas atas dan batas bawah pada pengguna. Kotak dialog permintaan masukan dapat dilihat pada gambar 5.8. Hasil penentuan batas dan peringatan bila nilai *eigen frequency* tidak normal dapat dilihat pada gambar 5.9.



Gambar 5.7 Letak Tombol Set Limit



Gambar 5.8 Tampilan Kotak Dialog Penentuan Batas Normal



Gambar 5.9 Hasil Perubahan Nilai Batas dan Peringatan Bila Nilai *Eigen frequency* Data yang Diamati Tidak Normal

5.2 Uji Coba Performa

Pada bagian subbab ini uji coba dibagi menjadi dua bagian. Bagian pertama adalah bagian di mana uji coba dilakukan pada prototipe jembatan yang dibuat oleh penulis. Uji coba ini bertujuan untuk membuktikan bila perubahan pada struktur mengakibatkan perubahan pada nilai *eigen frequency*. Pada bagian kedua uji coba dilakukan pada struktur jembatan sebenarnya dengan tujuan membuktikan apabila perubahan pada struktur yang mengakibatkan perubahan nilai *eigen frequency* juga berlaku pada obyek sebenarnya.

Pada tugas akhir ini, alat pemantau getaran yang digunakan mencatat getaran yang terjadi pada sumbu x, y, dan z struktur. Akan tetapi pengamatan nilai *eigen frequency* di fokuskan terhadap nilai *eigen frequency* pada sumbu z dikarenakan

percobaan dilakukan dengan memberi eksitasi pada sumbu z struktur.

Hal ini dikarenakan eksitasi pada sumbu x dan y tidak dapat dilakukan pada objek sebenarnya yaitu jembatan statistika ITS, sehingga percobaan dilakukan hanya pada nilai *eigen frequency* pada sumbu z saja.

5.2.1 Uji Coba Sistem pada Prototipe Jembatan

Uji coba prototipe ini dilakukan pada prototipe yang dibangun menggunakan besi yang dapat didapatkan dengan mudah pada toko bangunan. Variabel uji coba yang diubah pada uji coba ini adalah baut pada titik A, B dan C. Bentuk prototipe dan letak baut dapat dilihat pada gambar 5.10. Pengamatan juga dilakukan pada prototipe dalam bentuk rusak. Bentuk prototipe rusak dapat dilihat pada gambar 5.11.

Gambar 5.10 Prototipe Jembatan yang Digunakan Untuk Uji Coba pada Prototipe

Gambar 5.11 Prototipe Jembatan Dalam Kondisi Rusak

Ada beberapa skenario uji coba yang dilakukan pada bagian uji coba ini. Skenario-skenario tersebut adalah sebagai berikut:

1. Pengukuran tingkat longgarnya baut sebagai acuan kondisi tidak normal pada prototipe
2. Pengambilan nilai *eigen frequency* saat prototipe dianggap normal sebanyak 10 kali untuk menghitung akurasi ketepatan sistem dalam mendeteksi *eigen frequency* dan mengetahui *sampling rate node*.
3. Pengambilan nilai *eigen frequency* saat prototipe dianggap tidak normal sebanyak 10 kali untuk menghitung akurasi ketepatan sistem dalam mendeteksi *eigen frequency* dan mengetahui *sampling rate node*.

Akurasi uji coba akan disimpulkan dengan hasil *true positive, true negative, false positive dan false negative*. Berikut adalah keterangan dari istilah tersebut:

- True Positive
  Keadaan di mana struktur yang diuji coba adalah normal, dan sistem menganggap nilai *eigen frequency*-nya normal

- True Negative
  Keadaan di mana struktur yang diuji coba tidak normal, dan sistem menganggap nilai *eigen frequency*-nya normal
- False Positive
  Keadaan di mana struktur yang diuji coba normal, dan sistem menganggap nilai *eigen frequency*-nya tidak normal
- False Negative
  Keadaan di mana struktur yang diuji coba tidak normal, dan sistem menganggap nilai *eigen frequency*-nya tidak normal

5.2.1.1 Skenario Uji Coba Pengukuran Tingkat Longgarnya Baut

Skenario uji coba ini dilakukan untuk menentukan nilai longgar baut di mana struktur dianggap tidak normal. Skenario uji coba ini dilakukan pada setiap *node* yang terapasan pada prototipe. Berikut hasil uji coba yang telah dilakukan.

Tabel 5.7 Hasil Percobaan Tingkat Kelonggaran Baut

Longgar Baut	Node A	Node B	Node C
Normal	15,33	16,42	12,125
1 Putaran	19,9	15,4	16,909
2 Putaran	23,45	14,3	15,583
3 Putaran	32,36	35,9	19,909

Hasil uji coba tersebut menunjukkan bahwa tingkat kelonggaran pada putaran pertama tidak selalu menunjukkan perbedaan signifikan dengan nilai *eigen frequency* pada keadaan baut normal. Pada putaran kedua nilai *eigen frequency* memiliki perbedaan cukup signifikan sehingga pada uji coba prototipe ini

keadaan tidak normal diwakili oleh baut longgar sebanyak dua putaran.

5.2.1.2 Skenario Uji Coba Akurasi pada Protipe dalam Keadaan Normal

Sebelum uji coba akurasi dilakukan, pengambilan nilai *eigen frequency* terlebih dahulu untuk menentukan batas atas dan bawah yang dianggap normal. Berikut hasil pengambilan nilai batas atas dan batas bawah.

Tabel 5.8 Hasil Pengambilan Nilai *eigen frequency* pada Prototipe Normal untuk Penentuan Batas Atas dan Bawah Sistem

Pengambilan	***Node* A**	***Node* B**	***Node* C**
1	15,33	15,3	14,85
2	14,33	18,92	12,125
3	16,060	16,42	13,6
Nilai Maksimum	**16,060**	**18,92**	**14,85**
Nilai Minimum	**14,33**	**15,3**	**12,125**

Pengambilan data untuk menentukan *threshold* normal juga dilakukan pada protipe dengan kondisi tidak normal. Berikut hasil pengambilan data pada prototipe dengan kondisi tidak normal.

Tabel 5.9 Hasil Pengambilan Nilai *eigen frequency* pada Prototipe dengan Kondisi Tidak Normal

Kondisi Prototipe	***Node* A**	***Node* B**	***Node* C**
1 baut kendur	25,25	14,0	16,92
2 baut kendur	26,466	20,35	19,4
3 baut kendur	30,733	20,92	20,46
Rusak	9,266	4,35	6,866

Rusak dengan Penahan	42,9	24,3	8,6

Data tersebut dijadikan patokan batas atas dan bawah pada skenario percobaan ini. Batas bawah ditentukan dengan menemukan nilai rata-rata dari nilai minimum *eigen frequency* normal dengan nilai *eigen frequency* tidak normal dibawah nilai minimum yang paling mendekati. Sedangkan batas atas ditentukan dengan menemukan nilai rata-rata dari nilai maksimum *eigen frequency* normal dengan nilai *eigen frequency* tidak normal diatas nilai maksimum yang paling mendekati. Berikut adalah batas atas dan bawah yang ditentukan pada system:

Tabel 5.10 Penentuan Batas Atas dan Bawah Sistem

Batas	***Node* A**	***Node* B**	***Node* C**
Bawah	11,798	14,65	10,3625
Atas	20,655	19,635	15,885

Hasil uji coba dapat adalah sebagai berikut:

Tabel 5.11 Hasil Uji Coba Prototipe dalam Keadaan Normal pada *Node* A

	Nilai *Eigen frequency*	**Sampling Rate**	**Kondisi yang Terdeteksi**	**Hasil**
1	17,2	68,26	Normal	True Positive
2	19,133	68,26	Normal	True Positive
3	23,33	68,26	Tidak Normal	False Positive
4	16,133	68,26	Normal	True Positive
5	11,866	68,26	Normal	True Positive

6	13,466	68,26	Normal	True Positive
7	12	68,26	Normal	True Positive
8	12,375	64	Normal	True Positive
9	11,8	68,26	Normal	True Positive
10	12,8	68,26	Normal	True Positive

Tabel 5.12 Hasil Uji Coba Skenario Prototipe dalam Keadaan Normal pada *Node* B

	Nilai *Eigen frequency*	**Sampling Rate**	**Kondisi yang Terdeteksi**	**Hasil**
1	16,35	73,14	Normal	True Positive
2	35,92	73,14	Tidak Normal	False Positive
3	17,35	73,14	Normal	True Positive
4	18,21	73,14	Normal	True Positive
5	19,21	73,14	Normal	True Positive
6	16,92	73,14	Normal	True Positive
7	17,285	73,14	Normal	True Positive
8	17	73,14	Normal	True Positive
9	16,85	73,14	Normal	True Positive

10	16,71	73,14	Normal	True Positive

Tabel 5.13 Hasil Uji Coba Skenario Prototipe dalam Keadaan Normal pada *Node* C

	Nilai *Eigen frequency*	**Sampling Rate**	**Kondisi yang Terdeteksi**	**Hasil**
1	14,6	68,26	Normal	True Positive
2	11,9	73,14	Normal	True Positive
3	12,466	68,26	Normal	True Positive
4	13,07	73,14	Normal	True Positive
5	12,5	73,14	Normal	True Positive
6	13,14	73,14	Normal	True Positive
7	13,21	73,14	Normal	True Positive
8	13,92	73,14	Normal	True Positive
9	12,73	68,26	Normal	True Positive
10	12,21	73,14	Normal	True Positive

Persentase True Positive pada masing masing *node* adalah sebagai berikut:

Tabel 5.14 Persentase Hasil Uji Coba Prototipe Skenario dalam Keadaan Normal

Node	**Jumlah *True Positive***	**Jumlah *False Positive***	**Persentase *True Positive***	**Persentase *False Positive***
A	9	1	90%	10%
B	9	1	90%	10%
C	10	0	100%	0%
Rata-rata			**93,33%**	**6,66%**

5.2.1.3 Skenario Uji Coba Akurasi pada Protipe dalam Keadaan Tidak Normal

Pada skenario ini akan diambil nilai *eigen frequency* prototipe dalam keadaan yang dianggap tidak normal. Keadaan tidak normal dibuat dengan mengendurkan baut pada titik-titik lokasi *node* pada prototipe. Batas atas dan bawah sistem tetap menggunakan hasil uji coba pada skenario sebelumnya. Berikut hasil uji coba pada prototipe dalam keadaan tidak normal:

Tabel 5.15 Hasil Uji Coba pada Prototipe dalam Keadaan Tidak Normal pada *Node* A

	Nilai *Eigen frequency*	**Sampling Rate**	**Kondisi yng Terdeteksi**	**Hasil**
1	24,6	68,26	Tidak Normal	False Negative
2	20	68,26	Tidak Normal	False Negative
3	10,733	68,26	Normal	True Negative

4	23,9	73,14	Tidak Normal	False Negative
5	20,92	73,14	Tidak Normal	False Negative
6	11,315	53,844	Tidak Normal	False Negative
7	26,466	68,26	Tidak Normal	False Negative
8	29,28	73,14	Tidak Normal	False Negative
9	4,42	73,14	Tidak Normal	False Negative
10	25,8571	73,14	Tidak Normal	False Negative

Tabel 5.16 Hasil Uji Coba pada Prototipe dalam Keadaan Tidak Normal pada *Node* B

	Nilai *Eigen frequency*	**Sampling Rate**	**Kondisi yang Terdeteksi**	**Hasil**
1	20,142	73,14	Tidak Normal	False Negative
2	21,85	73,14	Tidak Normal	False Negative
3	19,93	68,26	Tidak Normal	False Negative
4	21,85	73,14	Tidak Normal	False Negative
5	22,42	73,14	Tidak Normal	False Negative
6	10,066	68,26	Tidak Normal	False Negative
7	19,66	68,26	Tidak Normal	False Negative
8	22,35	73,14	Tidak Normal	False Negative

9	20,5	73,14	Tidak Normal	False Negative
10	19	64	Normal	True Negative

Tabel 5.17 Hasil Uji Coba pada Prototipe dalam Keadaan Tidak Normal pada *Node* C

	Nilai *Eigen frequency*	Sampling Rate	Kondisi yang Terdeteksi	Hasil
1	19,71	73,14	Tidak Normal	False Negative
2	19,92	73,14	Tidak Normal	False Negative
3	21,85	73,14	Tidak Normal	False Negative
4	25,2	73,14	Tidak Normal	False Negative
5	21,07	78,76	Tidak Normal	False Negative
6	18,9125	64	Tidak Normal	False Negative
7	19,92	73,14	Tidak Normal	False Negative
8	29,14	73,14	Tidak Normal	False Negative
9	16,866	68,26	Tidak Normal	False Negative
10	29,93	68,26	Tidak Normal	False Negative

False negative adalah keadaan di mana nilai yang didapat sesuai dengan kondisi uji coba yaitu keadaan tidak normal, *true negative* adalah keadaan di mana nilai yang didapat tidak sesuai dengan kondisi uji coba.

Persentase *false negative* pada masing masing *node* adalah sebagai berikut:

Tabel 5.18 Persentase Hasil Uji Coba Prototipe Prototipe dalam Keadaan Tidak Normal

Node	**Jumlah *False Negative***	**Jumlah *True Negative***	**Persentase *False Negative***	**Persentase *True Negative***
A	9	1	90%	10%
B	9	1	90%	10%
C	10	0	100%	0%
Rata-rata			**93,33%**	**6,66%**

Performa pengambilan data pada uji coba prototipe dari setiap *node* adalah sebagai berikut:

Tabel 5.19 Performa Pengiriman Data pada Uji Coba Prototipe

Node	**Rata-rata Pengambilan Data per Detik**
A	68,5974
B	71,951
C	71,73

5.2.2 Uji Coba pada Jembatan Sebenarnya

Uji coba ini dilakukan pada jembatan Statistika di lingkungan kampus ITS. Uji coba ini memiliki tujuan untuk membuktikan bila perbedaan eksitasi pada percobaan ini hampir tidak memengaruhi nilai *eigen frequency* struktur. Tujuan lainnya adalah untuk mencoba apakah sistem ini dapat diterapkan pada obyek sebenarnya.

Variabel uji coba ini adalah lokasi pengamatan, di mana pengamatan dilakukan pada bagian jembatan yang berbeda. Bagian-bagian tersebut ada yang dianggap sebagai bagian yang

sudah tidak layak, hampir tidak layak dan layak. Lokasi penempatan *node* dapat dilihat pada gambar 5.12 dan 5.13.

Gambar 5.12 Lokasi Pengamatan pada Jembatan Statistika

Gambar 5.13 Peletakan *Node* pada Jembatan Statistika

Batu pada gambar 5.12 dijadikan penanda lokasi penempatan *node*. Dapat dilihat pada lokasi A, kayu pada permukaan jembatan lepas dari kerangka jembatan. Lokasi A ini dianggap sebagai bagian yang tidak layak. Pada lokasi B, ketika dilewati orang berjalan, kayu permukaan jembatan bergetar bersamaan dengan tiga kayu disebelahnya. Hal ini dikarenakan adanya baut yang lepas dari bagian jembatan ini. Maka dari itu lokasi B juga dianggap sebagai bagian yang tidak layak. Pada lokasi C, ketika dilewati orang berjalan getarannya tidak begitu terasa dan kayu permukaan jembatan baru saja diperbarui. Lokasi C ini dianggap sebagai bagian jembatan yang masih layak.

Skenario uji coba pada jembatan sebenarnya hampir sama dengan skenario uji coba prototipe, di mana kondisi normal diwakili *node* C dan tidak normal diwakili *node* A dan B.

5.2.2.1 Skenario Uji Coba pada Jembatan Sebenarnya pada Lokasi yang Dianggap Normal

Pada skenario ini pengambilan nilai *eigen frequency* struktur dilakukan pada *node* C di mana lokasi *node* C dianggap normal. Pengambilan data getaran untuk menentukan batas atas dan bawah adalah sebagai berikut:

Tabel 5.20 Pengambilan *Eigen frequency* Normal Untuk Nilai Batas

Pengambilan	Hasil *Eigen frequency*
1	5,4375
2	3,375

Pengambilan data juga dilakukan pada *node* A dan B di mana kondisi lokasi *node* A dan B dianggap tidak normal. Berikut hasil pengambilan data pada *node* A dan B:

Tabel 5.21 Pengambilan *Eigen frequency* Tidak Normal Untuk Nilai Batas

Node	Nilai *Eigen frequency*
A	9,625
B	8,25

Dari pengambilan data tersebut, hanya didapatkan nilai yang mendekati nilai maksimum data normal, sehingga batas bawah tidak dapat ditentukan dengan cara pada uji coba prototipe. Nilai batas bawah ditentukan dengan asumsi. Penentuan batas atas dan bawah untuk uji coba ini adalah sebagai berikut:

Tabel 5.22 Penentuan Batas Atas dan Bawah Uji Coba Jembatan Sebenarnya

Batas	Nilai
Bawah	3
Atas	6,84375

Hasil uji coba pada *node* C yang dianggap normal adalah sebagai berikut:

Tabel 5.23 Hasil Uji Coba Jembatan Sebenarnya pada *Node* C

	Nilai *Eigen frequency*	Sampling Rate	Kondisi yang Terdeteksi	Hasil
1	0,53	12,1	Tidak Normal	False Positive
2	14,86	68,26	Tidak Normal	False Positive
3	5,437	64	Normal	True Positive
4	9,933	68,26	Tidak Normal	False Positive

5	3,375	64	Normal	True Positive
6	5,9375	64	Normal	True Positive
7	5,0625	64	Normal	True Positive
8	3,0625	64	Normal	True Positive
9	3,25	64	Normal	True Positive
10	2,875	64	Tidak Normal	False Positives

Persentase uji coba pada *node* C adalah sebagai berikut:

Tabel 5.24 Persentase Uji Coba Jembatan Sebenarnya Skenario 1

Jumlah *True Positive*	**Jumlah *False Positive***	**Persentase *True Positive***	**Persentase *False Positive***
6	4	60%	40%

5.2.2.2 Skenario Uji Coba pada Jembatan Sebenarnya pada Lokasi yang Dianggap Tidak Normal

Pada skenario ini akan diuji pengambilan nilai *eigen frequency* pada *node* A dan B. Lokasi *node* A dan B dianggap tidak normal, sehingga uji coba ini bertujuan untuk mengetahui kualitas deteksi struktur tidak normal oleh sistem. Batas atas dan bawah mengikuti nilai yang telah ditentukan pada skenario uji coba sebelumnya.

Berikut hasil dari uji coba pada jembatan sebenarnya pada lokasi yang dianggap tidak normal:

Tabel 5.25 Hasil Uji Coba *Node* A pada Uji Coba Jembatan Sebenarnya

	Nilai *Eigen frequency*	Sampling Rate	Kondisi yang Terdeteksi	Hasil
1	9,625	64	Tidak Normal	False Negative
2	11,75	64	Tidak Normal	False Negative
3	16,3125	64	Tidak Normal	False Negative
4	3,0	64	Normal	True Negative
5	22,46	68,26	Tidak Normal	False Negative
6	11,125	64	Tidak Normal	False Negative
7	11,1875	64	Tidak Normal	False Negative
8	3,0625	64	Normal	True Negative
9	2,75	64	Tidak Normal	False Negative
10	11,875	64	Tidak Normal	False Negative

Tabel 5.26 Hasil Uji Coba *Node* B pada Uji Coba Jembatan Sebenarnya

	Nilai *Eigen frequency*	Sampling Rate	Kondisi yang Terdeteksi	Hasil
1	3,6875	64	Normal	True Negative
2	0,5625	64	Tidak Normal	False Negative

3	1,813	23,81	Tidak Normal	False Negative
4	0,4375	64	Tidak Normal	False Negative
5	6,4375	64	Normal	True Negative
6	5,4375	64	Normal	True Negative
7	2,5	64	Tidak Normal	False Negative
8	2,125	64	Tidak Normal	False Negative
9	2,06	68,26	Tidak Normal	False Negative
10	8,25	64	Tidak Normal	False Negative

Persentase data uji coba adalah sebagai berikut:

Tabel 5.27 Persentase Data Uji Coba pada Jembatan Sebenarnya pada Lokasi yang Dianggap Tidak Normal

Node	**Jumlah *False Negative***	**Jumlah *True Negative***	**Persentase *False Negatives***	**Persentase *True Negatives***
A	8	2	80%	20%
B	7	3	70%	30%
Rata-rata			**75%**	**25%**

Performa pengambilan data pada skenario normal dan tidak normal uji coba jembatan sebenarnya dari setiap *node* adalah sebagai berikut:

Tabel 5.28 Rata-rata Jumlah Pengambilan Data per Detik

Node	Rata-rata Pengambilan Data per Detik
A	64,426
B	60,47
C	59,662

5.3 Evaluasi Umum Uji Coba

Subbab ini akan membahas hasil uji coba yang dilakukan baik fungsional maupun performa. Berikut adalah bahasan tersebut.

5.3.1 Evaluasi Uji Coba Fungsional

Pada uji coba fungsional, telah dibuat beberapa skenario untuk menguji jalannya aplikasi. Hasil uji coba fungsional adalah sebagai berikut:

Tabel 5.29 Evaluasi Uji Coba Fungsional

Uji Coba	Hasil Uji Coba
Pengiriman data dari *node* pemantau ke *node sink*	Berhasil
Pencatatan data getaran pada berkas berformat .txt	Berhasil
Penyimpanan detail pengambilan data pada sistem basis data MySQL	Berhasil
Menampilkan data yang telah diolah	Berhasil
Memasukkan *threshold*	Berhasil

Seluruh uji coba fungsional dapat dilakukan dengan baik oleh sistem yang dibangun pada tugas akhir ini.

5.3.2 Evaluasi Uji Coba Performa

Evaluasi akan dibagi menjadi dua karena uji coba performa juga dibagi dua. Uji coba sistem pada prototipe jembatan memiliki hasil sebagai berikut:

Tabel 5.30 Data Uji Coba Performa pada Prototipe

Hasil	Jumlah	Persentase
True Positive	28 dari 30 data	93,33%
False Positive	2 dari 30 data	6,66%
False Negative	28 dari 30 data	93,33%
True Negative	2 dari 30 data	6,66%

Dari data di atas dapat dicari akurasi sistem terhadap perubahan struktur. Akurasi deteksi perubahan struktur:

$$\text{Akurasi} = \frac{\text{TP+FN}}{TP+TN+FP+FN} \text{ x } 100\%$$

$$= \frac{93,33+93,33}{93,33+6,66+6,66+93,33} \text{ x } 100\%$$

$$= 93,33\%$$

Sedangkan data dari percobaan pada jembatan sebenarnya adalah sebagai berikut:

Tabel 5.31 Data Uji Coba Performa pada Jembatan Sebenarnya

Hasil	Jumlah	Persentase
True Positive	6 dari 10 data	60%
False Positive	4 dari 10 data	40%
False Negative	15 dari 20 data	75%
True Negative	5 dari 20 data	25%

Akurasi deteksi perubahan struktur pada uji coba pada jembatan sebenarnya adalah sebagai berikut:

$$\text{Akurasi} = \frac{\text{TP+FN}}{TP+TN+FP+FN} \text{ x } 100\%$$

$$= \frac{60+75}{60+40+25+75} \text{ x } 100\%$$

$$= 67{,}5\%$$

Performa pengiriman menggunakan modul nRF24l01 adalah sebagai berikut:

Tabel 5.32 Rata-rata Pengambilan Data pada Semua Uji Coba

Uji Coba	***Node* A**	***Node* B**	***Node* C**
Jembatan Sebenarnya	64,639	59,5915	65,065
Protipe	68,5974	71,951	71,73
Rata-rata	**66,6182**	**65,771**	**68,3975**

Semua *node* memiliki rata-rata pengiriman yang cukup baik yakni diatas 60 data per detik.

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan sistem lebih lanjut.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba *monitoring* kondisi struktur bangunan berdasarkan nilai modal struktur adalah sebagai berikut:

1. Implementasi pencarian nilai modal (*eigen frequency*) menggunkan algoritma FFT yang telah dibuat dapat dijadikan salah satu metode dalam sistem monitoring struktur bangunan.
2. Protokol pengiriman data yang digunakan untuk mengirim data getaran menggunakan protokol di mana data dikirim secara konstan tanpa *delay*. Performa pengiriman dari protokol tersebut pada setiap *node* terbukti cukup baik dengan rincian *node* A dengan rata-rata 66**,**6182 data per detik, *node* B dengan 65**,**771 data per detik dan *node* C dengan 68,3975 data per detik.
3. Bentuk data yang digunakan dalam pengiriman data dalah tipe data *string*, di mana data dibedakan oleh *char* pertama dari data yang dikirim. Tipe data ini dipilih setelah dibandingkan dengan tipe data *struct* yang memiliki ukuran lebih besar tidak cocok untuk digunakan pada protokol yang mengutamakan jumlah data yang dikirim per detiknya.
4. Prosedur penerimaan data dilakukan dengan membedakan sumber data (node pengirim) dan mencatat data tersebut ke dalam berkas sesuai sumber data getaran. Data yang telah dicatat tersebut kemudian disimpan detailnya ke dalam basis data untuk pengolahan data lebih lanjut.

5. Akurasi sistem dalam mendeteksi perubahan struktur didapatkan dengan terlebih dahulu menentukan *threshold* data yang dianggap normal dan tidak normal. Akurasi kemudian ditentukan dengan membandingkan hasil uji coba pengambilan data sebanyak sepuluh kali dengan *threshold* yang telah ditentukan.
6. Akurasi yang didapatkan pada uji coba menggunakan prototipe lebih tinggi dari uji coba pada jembatan sebenarnya. Akurasi dari uji coba sistem pada prototipe adalah 93,33% sedangkan pada uji coba sistem pada jembatan sebenarnya sebesar 67,5% akurasi.

6.2 Saran

Saran yang diberikan terkait pengembangan pada Tugas Akhir ini adalah sebagai berikut:

1. Sebaiknya gunakan akselerometer yang lebih sensitif untuk mendapatkan nilai yang lebih akurat (tugas akhir ini menggunakan akselerometer dengan sensitifitas 2G) karena ada beberapa data yang melenceng jauh pada percobaan pada jembatan sebenarnya dikarenakan kurangnya sensitifitas akselerometer terhadap getaran yang terjadi.
2. Untuk pengaplikasian dalam dunia sebenarnya membutuhkan sumber daya yang lebih tahan lama daripada baterai 9v.

DAFTAR PUSTAKA

[1] Yoyong Arfiadi, "Sistem Monitoring Kesehatan Struktur," *Fak. Tek. Univ. Atma Jaya Yogyak.*, Nov. 2007.

[2] S. C. Siriwardane, "Vibration measurement-based simple technique for damage detection of truss bridges: A case study," *Case Stud. Eng. Fail. Anal.*, vol. 4, pp. 50–58, Oct. 2015.

[3] F. Neitzel, B. Resnik, S. Weisbrich, and A. Friedrich, "Vibration monitoring of bridges," *Rep. Geod.*, 2011.

[4] "Arduino Uno & Genuino Uno," *Arduino-ArduinoBoardUno*. [Online]. Available: https://www.arduino.cc/en/main/arduinoBoardUno#techspecs. [Accessed: 22-May-2017].

[5] "ADXL345 Datasheet and Product Info | Analog Devices." [Online]. Available: http://www.analog.com/en/products/mems/accelerometers/adxl345.html#product-overview. [Accessed: 18-May-2017].

[6] Nordic Semiconductor, "nRF24L01_Product_Specification_v2_0.pdf." .

[7] C. V. Loan, *Computational Frameworks for the Fast Fourier Transform*. SIAM, 1992.

[8] "JTransforms - Piotr Wendykier." [Online]. Available: https://sites.google.com/site/piotrwendykier/software/jtransforms. [Accessed: 18-May-2017].

[9] P. Bhatt, *Maximum Marks Maximum Knowledge in Physics*. Allied Publishers.

[10] J.-T. Kim, Y.-S. Ryu, H.-M. Cho, and N. Stubbs, "Damage identification in beam-type structures: frequency-based method vs mode-shape-based method," *Eng. Struct.*, vol. 25, no. 1, pp. 57–67, Jan. 2003.

[11] R. D. Blevins, *Formulas for Natural Frequency and Mode Shape*. Krieger Publishing Company, 2001.

[12] "Arduino - Introduction." [Online]. Available: https://www.arduino.cc/en/guide/introduction. [Accessed: 28-May-2017].

[13] "HTML5 Web Development Support." [Online]. Available: https://netbeans.org/features/html5/index.html. [Accessed: 28-May-2017].
[14] "NetBeans MOVED." [Online]. Available: https://platform.netbeans.org/tutorials/. [Accessed: 28-May-2017].
[15] "MySQL." [Online]. Available: https://www.mysql.com/. [Accessed: 28-May-2017].
[16] "mysqldevelopment.com." [Online]. Available: http://www1.mysqldevelopment.com/?kw=web%20development. [Accessed: 28-May-2017].
[17] C. W. de Silva, *Vibration: Fundamentals and Practice, Second Edition*. CRC Press, 2006.
[18] "Working with Packing Structures." [Online]. Available: https://msdn.microsoft.com/en-us/library/ms253935.aspx. [Accessed: 16-Jul-2017].
[19] "Frequency-Domain Periodicity and the Discrete Fourier Transform - Eric Jacobsen." [Online]. Available: https://www.dsprelated.com/showarticle/175.php. [Accessed: 29-May-2017].
[20] Brett Ninness, "Spectral Analysis Using FFT." Department of Electrical and Computer Engineering The University of Newcastle, Australia.

BIODATA PENULIS

Muhammad Ilham Mirza Akbar lahir di Surabaya pada tanggal 11 Juli 1996. Penulis menempuh pendidikan formal dimulai dari TK Aisyiah (2000-2002), SD Muhammadiyah 6 (2002-2008), SMPN 6 Surabaya (2008-2011), SMAN 5 Surabaya (2011-2013) dan S1 Teknik Informatika ITS (2013-2017). Bidang studi yang diambil oleh penulis pada saat berkuliah di Teknik Informatika ITS adalah Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi seperti KMI (2016). Penulis juga aktif dalam berbagai kegiatan kepanitiaan yaitu SCHEMATICS 2014 divisi sie keamanan dan SCHEMATICS 2015 sie keamanan. Penulis dapat dihubungi melalui email: ilhammirza123@gmail.com.