Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IF184802

RANCANG BANGUN SISTEM PENGAMATAN BERBASIS WSN UNTUK DETEKSI DINI KEJADIAN TANAH LONGSOR DENGAN METODE MULTIPLE THRESHOLDING

MUHAMMAD RYANDA NUGRAHA M
NRP 05111640000180

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

Dosen Pembimbing II
Dr. Radityo Anggoro , S.Kom, M.Sc

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IF184802

RANCANG BANGUN SISTEM PENGAMATAN BERBASIS WSN UNTUK DETEKSI DINI KEJADIAN TANAH LONGSOR DENGAN METODE MULTIPLE THRESHOLDING

MUHAMMAD RYANDA NUGRAHA M
NRP 05111640000180

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

Dosen Pembimbing II
Dr. Radityo Anggoro , S.Kom, M.Sc

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

DESIGN AND DEVELOPMENT OF WSN-BASED OBSERVATION SYSTEM FOR EARLY DETECTION ON LANDSLIDE OCCURRENCE WITH MULTIPLE THRESHOLDING METHOD

MUHAMMAD RYANDA NUGRAHA M
NRP 05111640000180

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

Second Advisor
Dr. Radityo Anggoro , S.Kom, M.Sc

DEPARTMENT OF INFORMATICS ENGINEERING
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM PENGAMATAN BERBASIS WSN UNTUK DETEKSI DINI KEJADIAN TANAH LONGSOR DENGAN METODE MULTIPLE THRESHOLDING

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
MUHAMMAD RYANDA NUGRAHA M
NRP: 05111640000180

Disetujui oleh Pembimbing tugas akhir:

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

2. Dr. Radityo Anggoro, S.Kom, M.Sc. .....
   (NIP. 19841016 2008121002) (Pembimbing 2)

SURABAYA
JUNI, 2020

(Halaman ini sengaja dikosongkan)

DESIGN AND DEVELOPMENT OF WSN-BASED OBSERVATION SYSTEM FOR EARLY DETECTION ON LANDSLIDE OCCURRENCE WITH MULTIPLE THRESHOLDING METHOD

Nama Mahasiswa	**:**	**Muhammad Ryanda Nugraha M**
NRP	**:**	**05111640000180**
Departemen	**:**	**Teknik Informatika - FTEIC ITS**
Dosen Pembimbing 1	**:**	**Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**:**	**Dr. Radityo Anggoro , S.Kom, M.Sc.**

Abstrak

Wireless Sensor Network (WSN) adalah sekumpulan node sensor yang saling berkomunikasi secara nirkabel untuk mengumpulkan data pada lingkungan disekitarnya Modul nRF24l01 merupakan salah satu contoh alat komunikasi untuk WSN, modul ini menggunakan daya yang sangat rendah sehingga cocok digunakan pada *node-node* dalam *wireless sensor network* guna melakukan monitoring tanah longsor.

Metode yang digunakan dalam penelitian ini dalam pendeteksian tanah longsor adalah multiple thresholding. Pertama dilakukan simulasi tanah longsor sambal mengumpulkan data nilai akselerasi tanah dengan modul ADXL345 pada saat terjadi longsor menggunakan node slave sebagai pengirim data dan node master sebagai penerima. Setelah itu, data yang diperoleh digunakan untuk mencari nilai threshold yang sesuai berdasarkan changing point pada dataset. Juga digunakan decision tree sehingga kondisi longsor akan ditetapkan setelah nilai yang didapatkan oleh sensor melewati seluruh nilai threshold yang telah ditetapkan pada setiap node slave.

Berdasarkan hasil uji coba dengan menggunakan beberapa scenario uji coba mulai simulasi longsor sungguhan hingga

menggerakkan sensor secara manual untuk menguji akurasi ditemukan bahwa akurasi mengalami penerunan saat sensor digerakkan secara manual dengan tempo yang cepat.Namun, pada saat uji coba simulasi longsor sungguhan akurasi yang didapatkan diatas 60% sehingga metode ini cocok digunakan untuk kondisi dimana gangguan external sangat minim.

Kata kunci: ***Wireless Sensor Network, Changing Point, Threshold, nRF24, ADXL345*****.**

RANCANG BANGUN SISTEM PENGAMATAN BERBASIS WSN UNTUK DETEKSI DINI KEJADIAN TANAH LONGSOR DENGAN METODE MULTIPLE THRESHOLDING

Student's Name : Muhammad Ryanda Nugraha M
Student's ID : 05111640000180
Department : Informatics Engineering- FTEIC-ITS
First Advisor : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
Second Advisor : Dr. Radityo Anggoro , S.Kom, M.Sc.

Abstract

Wireless Sensor Network (WSN) is a collection of sensor nodes that communicate wirelessly to each other to collect data in the surrounding environment. The nRF2410 module is an example of a communication tool for WSN, this module uses very low power making it suitabel for use in nodes in the wireless sensor network to monitor landslides.

The method used in this study for landslide detection is multiple thresholding. The first is to do a landslide simulation while collecting acceleration value data with the ADXL345 module in the event of a landslide using the slave node as the data sender and the master node as the receiver. After that, the data obtained is used to find the appropriate threshold value based on changing points in the dataset. A decision tree is also used so that the landslide condition will be determined after the value obtained by the sensor passes all the threshold values set at each slave node.

Based on the results of trials using several trial scenarios starting from a real landslide simulation to manually move the sensor to test the accuracy it was found that the accuracy experienced lighting when the sensor was moved manually with a fast tempo. However, when testing the real landslide simulation the accuracy obtained above 60% so that this method is suitabel for conditions where external interference is minimal.

Kata kunci: ***Wireless Sensor Network, Changing Point, Threshold, nRF24, ADXL345*****.**

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul

"RANCANG BANGUN SISTEM PENGAMATAN BERBASIS WSN UNTUK DETEKSI DINI KEJADIAN TANAH LONGSOR DENGAN METODE MULTIPLE THRESHOLDING".

Harapan dari penulis, semoga apa yang tertulis di dalam buku tugas akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini dan ke depannya, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT. Dan Nabi Muhammad SAW. yang telah membimbing penulis selama hidup.
2. Keluarga penulis (Ayah, Ibu, Adik dan keluarga penulis yang lain) yang selalu memberikan dukungan baik berupa doa, moral, dan material yang tak terhingga kepada penulis, sehingga penulis dapat menyelesaikan tugas akhir ini.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng.,Ph.D. dan Bapak Dr. Radityo Anggoro , S.Kom, M.Sc . selaku Dosen Pembimbing penulis yang telah membimbing, memberikan nasihat, dan memotivasi penulis sehingga penulis dapat menyelesaikan tugas akhir ini.
4. Bapak Dr. Eng. Chastine Fatichah, S.Kom,, M.Kom. selaku kepala Departemen Teknik Informatika ITS.
5. Bapak dan Ibu Dosen yang telah memberikan ilmunya selama penulis berkuliah di Teknik Informatika ITS.

6. Teman-teman Kanda ITS yang telah menjadi obat rindu akan kampung halaman selama berada di ITS.
7. Laboratorium NCC juga adminnya yang telah menemani penulis selama pengerjaan tugas akhir ini.
8. Sahabat IGS yang telah menjadi teman bermain saat jenuh dalam mengerjakan tugas
9. UKM Robotika ITS yang telah menjadi tempat penulis dalam berkarya selama kurang lebih 2 tahun.
10. Teman Teman Internship di NTUST selama penulis internship di Taipei, Taiwan (Andrew, Mbak Daus, Rasyid, Aufa, Hazdik, Faris, Sivra, dan Angelo). Juga Zahrah yang telah memberikan informasi terkait internship di NTUST
11. Teman satu tim pada saat Gemastik (Ubut,Agung,Mala,Aang) yang menjadi awal mula munculnya ide Tugas Akhir penulis
12. Teman-teman satu bimbingan (Ubut, Agung, Affan, Chendra, Yasinta dan Iqbal) yang selalu memberikan dukungan dan informasi selama mengerjakan tugas akhir ini.
13. Teman-teman angkatan 2016 (TC16) yang sudah menjadi saksi hidup perjalanan karir penulis selama berkuliah di Teknik Informatika ITS.
14. Ibu-Ibu CS yang sangat baik dan memastikan TC selalu menjadi tempat yang nyaman untuk penulis.
15. Para staff dan pengurus Asrama ITS selalu berusaha memastikan keamanan dan kenyamanan penghuninya.
16. Untuk orang-orang yang tidak dapat disebutkan satu persatu oleh penulis dan pembaca buku tugas akhir ini.

Namun, penulis memohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya. Tetap semangat dalam berjuang untuk menjalani kehidupan, Jangan pernah menyerah, karena Allah masih ingin melihat kita berjuang semampu kita.

Semoga kita semua selalu diberi kebahagiaan lahir dan batin dan kesuksesan dunia akhirat. Aamiin.

Surabaya, Juni 2020

Muhammad Ryanda Nugraha M

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Tanah longsor merupakan bencana yang sangat merugikan, tanah longsor tidak terjadi secara tiba-tiba, tanah longsor dimulai dari hujan dengan intensitas tinggi dan dapat menjadi pemicu tanah longsor jika tidak adanya daya serap air yang cukup. Tanah longsor bisa merusak berbagai macam property ,bangunan ataupun mengancam suatu lingkungan ekosistem. Tanah longsor sering terjadi di lingkungan yang sensitif seperti hutan, perbukitan atau dataran tinggi. Apabila tanah longsor sudah terjadi dalam skala besar seperti di hutan atau kawasan dataran tinggi maka akan sangat sulit untuk mengatasinya. Karena itu maka perlu adanya sistem untuk memantau suatu lingkungan terhadap kemungkinan terjadinya tanah longsor.

Perkembangan teknologi komunikasi saat ini teleh berkembang dengan sangat pesat, salah satunya dibidang *Wireless Sensor Network* (WSN). Wireless Sensor Network (WSN) adalah arsitektur jaringan nirkabel yang terdistribusi biasanya digunakan dalam jumlah besar untuk memanta suatu kondisi lingkungan atau sistem oleh pengukuran parameter fisik seperti suhu, tekanan, atau kelembapan.

Dalam tugas akhir ini, akan dibuat sistem deteksi dini tanah longsor yang akan mengimplementasikan mekanisme Wireless Sensor Network (WSN) untuk mendapatkan data sensor-sensor, data sensor tersebut akan dikumpulkan untuk menjadi dataset dalam proses pencarian nilai threshold. Kemudian akan dibuat sistem deteksi dini tanah longsor jika nilai-nilai data dari sensor yang dikumpulkan melebihi parameter threshold yang telah ditentukan.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana metode pengambilan dan pemilihan data-data sensor untuk melakukan pengamatan tanah longsor?
2. Bagaimana cara mendapatkan nilai thresholding untuk parameter sensor?
3. Bagaimana mendeteksi tanah longsor dari parameter-parameter sensor yang didapatkan?

1.3 Batasan Permasalahan

Berdasarkan masalah yang diuraikan oleh penulis, maka batasan masalah pada tugas akhir ini adalah:

1. Menggunakan platform Arduino sebagai microcontroller.
2. Menggunakan modul nRF24L01 sebagai media transmisi data ke node master.
3. Menggunakan Sensor ADXL345 sebagai sensor untuk mengukur akselerasi pergerakan tanah.
4. Menggunakan Arduino Uno sebagai base station dan media transmisi untuk mengirimkan data.
5. Menggunakan Laptop MSI Leopard sebagai sarana komputasi.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Membangun sebuah sistem deteksi dini tanah longsor yang dapat memantau kondisi suatu lingkungan agar dapat memberikan peringatan dini akan potensi tanah longsor apabila nilai-nilai threshold sensor yang didapatkan melebihi batas nilai suatu kondisi lingkungan normal.

1.5 Manfaat

Hasil dari pengerjaan Tugas Akhir ini memiliki manfaat untuk menghasilkan sebuah sistem pengamatan tanah longsor yang dapat digunakan dalam mendeteksi terjadinya tanah longsor pada suatu lingkungan.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi gambaran tentang tugas akhir yang akan dibuat. Pendahuluan proposal tugas akhir meliputi hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah yang menjadi konstrain dari tugas akhir, tujuan pembuatan tugas akhir, dan manfaat dari hasil tugas akhir. Di dalam proposal tugas akhir juga dijabarkan mengenai tinjauan pustaka yang menjadi referensi pendukung dalam pembuatan tugas akhir ini. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Studi literatur yang dilakukan dalam pengerjaan Tugas Akhir ini adalah mengenai Rancang Bangun Sistem Pengamatan Berbasis Wsn Untuk Deteksi Dini Kejadian Tanah Longsor Dengan Metode Multiple Thresholding. Sehingga, studi literatur ini dapat diterapkan pada perancangan jaringan sensor nirkabel yang dapat melakukan pendeteksian tanah longsor.

1.6.3 Implementasi Sistem

Implementasi merupakan tahap untuk mengimplementasikan metode-metode yang sudah diajukan pada proposal tugas akhir. Untuk membangun algoritma yang telah dirancang sebelumnya, implementasi bahasa C/C++ dan python sebagai bahasa pemrograman untuk uji coba mengimplementasikan metode yang sudah diajukan.

1.6.4 Pengujian dan Evaluasi

Pengujian dan evaluasi dari hasil Tugas Akhir ini akan diujicobakan dengan membangun jaringan sensor nirkabel menggunakan modul wireless nRF2L01 yang berlokasi di halaman depan rumah penulis.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan permasalahan, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan dari metode, algoritma, *library*, dan *tools* yang digunakan dalam penyusunan tugas akhir ini. Kajian teori yang dimaksud berisi tentang penjelasan singkat mengenai *wireless communication*, *changing point* dan *decision tree*.
3. Bab III. Perancangan Perangkat Lunak dan Perangkat Keras
   Bab ini berisi pembahasan mengenai desain dari jaringan sensor nirkable yang akan dibuat, meliputi arsitektur dann proses perangkat lunak dan perangkat keras.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi dari desain dari jaringan yang akan dilakukan pada tahan desain, meliputi potongan *pseudocode* yang terdapat dalam perangkat ludan dan perangkat keras yang digunakan.
5. Bab V. Pengujian dan Evaluasi
   Bab ini berisi hasil uji coba dan evaluasi dari implementasi jaringan sensor nirkabel yang dibuat dengan melihat keluaran yang dihasilkan, analisa dan eveluasi untuk mengatahui akurasi dari sistem yang dibuat.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami

pada proses pengerjaan tugas akhir, dan saran untuk pengembangan tugas akhir ke depannya.

7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.
8. Lampiran
   Dalam lampiran terdapat kode sumber program secara keseluruhan.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode dan alat yang digunakan dalam tugas akhir. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan riset yang berkaitan.

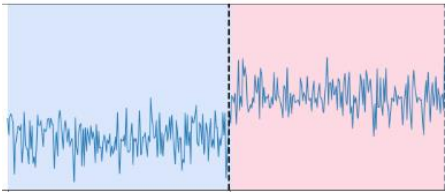
2.1 *Wireless Sensor Network*

Wireless Sensor Network (WSN) adalah arsitektur jaringan nirkabel yang terdistribusi biasanya digunakan dalam jumlah besar untuk memonitor suatu kondisi lingkungan atau sistem oleh pengukuran parameter fisik seperti suhu, tekanan atau kelembapan. Jaringan sensor nirkabel ini dioperasikan dengan daya baterai dan energinya tidak selalu diperbaharui karena masalah biaya, lingkungan dan bentuknya.

2.2 Changing point

Changing point adalah titik dimana data yang diberikan mengalami perubahan yang signifikan, istilah ini kadang ditemukan pada signal processing untuk melakukan segmentasi pada data time series. Pada gambar 2.2.1 dapat dilihat changing point dari grafik yang ditampikan berada di garis putus-putus.

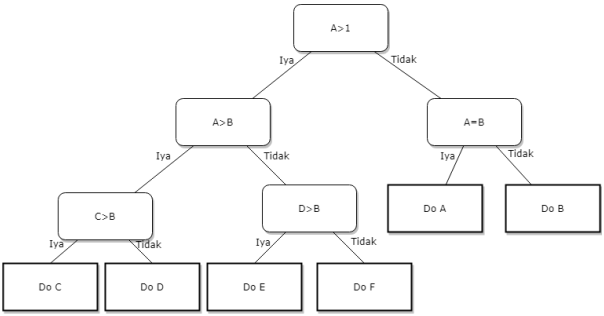
Changing point dapat ditemukan dengan melihat persentase perubahan data untuk setiap perubahan yang terjadi kemudian mengambil persentase yang paling besar.



Gambar 2.2.1 Changing Point

2.3 Decision Tree

Decision tree adalah alat bantu dalam pengambilan keputusan yang menggunakan model tree untuk setiap keputusan yang diambil beserta konsekuensi dari keputusan tersebut. Decision tree seringkali digunakan untuk melakukan pengambilan keputusan berdasarkan data yang telah diolah sebelumnya.



Gambar 2.3.1 *Contoh Decision Tree*

2.4 Modul ADXL345

ModulADXL345 adalah sensor accelerometer dan gyrometer 3-axis yang dapat digunakan untuk mendeteksi adanya pergerakan atau getaran pada suatu lokasi. Sensor ini berukuran kecil dan menggunakan daya sangat kecil sehingga cocok digunakan untuk pendeteksian dengan jangka waktu yang lama. Sensor ini memiliki nilai offset 0.15 sampai 0.2

Gambar 2.4.1 Modul ADXL345

=

2.5 Modul nRF24L01

Modul Wireless nRF24L01 adalah modul komunikasi jarak jauh yang dilengkapi dengan sirkuit *Low Noise Amplifier* (LNA) dan *Power Amplifier* (PA). nRF24l01 dapat mentransmisikan data hingga jarak 1,1 km dengan kecepatan transmisi data dari modul NRF24L01 ini dapat mencapai 2 Mbps pada frekuensi 2.4 GHz. Modul ini didesain untuk jaringan nirkabel yang membutuhkan penggunaan daya sangat rendah dan cocok digunakan dengan *Microcontroller* Arduino.

Modul nRF24l01 dapat berfungsi sebagai *transmitter* dan juga sebagai *receiver*. Namun nRF tidak dapat menjadi *transmitter* dan *receiver* dalam satu waktu sehingga diperlukan pergantian mode.

Selain itu, Modul ini memiliki kelebihan dalam kehandalan. Portabilitas, dan pemakaian daya. Pemakaian daya yang rendah dan dimensi yang kecil membuat modul ini banyak dipakai dalam bidang yang memerlukan komunikasi nirkabel tidak terkecuali jaringan sensor nirkabel.

Gambar 2.5.1 *Modul nRF24l01*

2.6 Arduino

Arduino adalah micro-controller single-board yang bersifat sumber terbuka, diturunkan dari Wiring platform dan dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. Produk Arduino bersifat *open source* dan dilisensikan dibawah GNU Lesser General Public License (LGPL) atau GNU General Public License (GPL), membuat produk Arduino dapat didistribusikan dan digunakan oleh siapapun tanpa harus membayar lisensi maupun royalti.

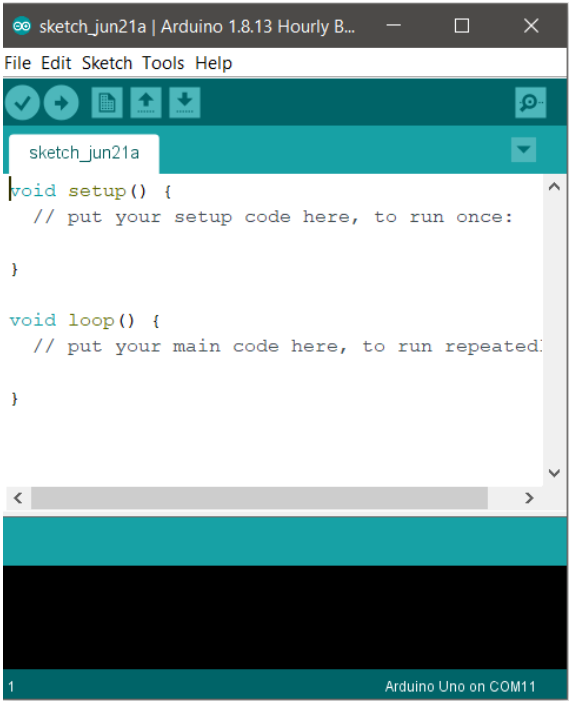
Board Arduino mempunyai antarmuka komunikasi serial, termasuk *Universal Serial Bus* (USB) yang digunakan untuk memprogram mikrokontroler Arduino melalui komputer. Board arduino menggunakan beragam jenis mikroprosesor dan mikrokontroler. Board Arduino dilengkapi dengan set pin *input/output* (I/O) digital dan analog yang bisa digunakan untuk berkomunikasi dengan board lainnya atau modul expansi yang terdapat di pasaran. Mikrokontroler Arduino dapat diprogram menggunakan bahasa C dan C++.

Gambar 2.6.1: Arduino Uno R3

2.7 Arduino IDE

Arduino IDE merupakan sebuah perangkat lunak yang digunakan sebagai tempat untuk menulis code program dari suatu skema rangkaian yang terhubung dengan board Arduino yang dibangun dengan bahasa pemrograman Java dan bersifat cross-platform. Barisan kode dalam Arduino IDE ditulis mengikuti aturan dari C/C++ dan baris kode ini disebut dengan istilah sketch.

. Arduino IDE mempunyai fitur deteksi otomatis bila ada Arduino yang dihubungkan ke komputer. Arduino IDE dipakai karena memiliki kompatibilitas baik dengan semua perangkat Arduino.Dalam melakukan debugging Arduino IDE memiliki fitur serial monitor menampilkan data yang diproses didalam arduino.



Gambar 2.7.1 Arduino IDE

(***Halaman ini sengaja di kosongkan***)

BAB III
PERANCANGAN

Perancangan merupakan bagian penting dalam pembuatan perangkat lunak dan perangkat keras yang berupa perencanaan secara teknis dari sistem jaringan yang dibuat. Pada Bab ini akan dibahas mengenai perancangan dan implementasi rancang bangun sistem pengamatan berbasis wsn untuk deteksi dini kejadian tanah longsor dengan metode multiple thresholding, Menggunakan modul Wireless nRF24L01 untuk berkomunikasi antar node dan menggunakan sensor ADXL345 untuk melakukan pengambilan data dan pengujian.

3.1 Deskripsi Umum

Pada tugas akhir ini akan dibuat sebuah implementasi *wireless sensor network* menggunakan mikrokontroler Arduino.

Pada *wireless sensor network* yang akan dibuat terdiri dari beberapa node berdasarkan scenario uji coba yang dilakukan. Satu node akan berperan manjadi *master node* sementara node-node lainnya akan menjadi *slave node*. Setiap node masing-masing berisi modul nRF24L01 sebagai media *transmisi* dan modul ADXL345 untuk melakukan pengambilan data dan pengujian.

Node master akan digunakan pada saat testing sebagai media penerima data akselerasi dari node slave melalui modul nRF24L01 dan data-data yang diterima akan disimpan untuk nantinya ditentukan apakah kondisi tersebut longsor atau tidak

Node slave memiliki dua fungsi dimana kedua fungsi tersebut memiliki rangkaian yang berbeda. Fungsi pertama adalah sebagai data collector untuk memperoleh dataset dan sebagai alat testing untuk hasil threshold yang telah didapatkan. Node master pada saat bertindak sebagai penerima data dari data collector terhubung dengan laptop tanpa dengan menggunakan kabel. Setelah dataset dikumpulkan akan dilakukan proses komputasi untuk memperoleh nilai threshold untuk setiap dataset

Node master akan terhubung dengan computer dan mendapat daya tetap dari computer, sedangkan *slave node* menggunakan powerbank sebagai penyuplai daya.

Data yang didapatkan dari *slave* node akan diolah menggunakan changing point detection untuk mencari nilai threshold untuk setiap node slave. Setelah ditemukan nilai threshold nilai tersebut akan dimasukkan ke sensor untuk selanjutnya dilakukan testing menggunakan nilai tersebut.

3.2 Daftar Istilah

Daftar istilah yang sering digunakan pada buku tugas akhir ini dapat dilihat pada Tabel 3.2.1.

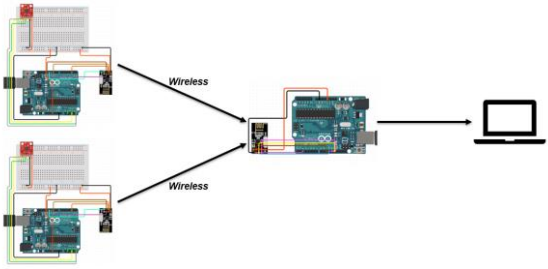
Tabel 3.2.1.6.5.1 Daftar Istilah

No.	Istilah	Penjelasan
1	*Node*	Sebuah titik redistribusi atau titik akhir dari suatu komunikasi.
2	*Slave Node*	*Slave node* merupakan kumpulan node-node yang terdapat dalam jaringan sensor nirkabel yang berfungsi sebagai mengumpulkan dan mengirim data juga untuk menguji nilai threshold
3	*Master node*	*Master node* merupakan node yang menjadi penerima data dari *slave node*.
4	*Changing point*	*Changing point* adalah suatu titik pada dataset dimana terjadi perubahan signifikan pada nilai yang menyebabkan perubahan pola pada data

3.3 Arsitektur Sistem

Pada sub bab ini akan di jelaskan mengenai arsitektur umum jaringan sensor nirkabel yang akan dibangun. Sistem ini terdiri dari beberapa node yang terdiri dari satu node sebagai *master* dan dua

node lain sebagai *slave* node, setiap node menggunakan *mikrocontroler* Arduino, modul nRF24L01 yang menggunakan SPI sebagai interface komunikasi dengan Arduino Komunikasi yang akan dibangun merupakan komunikasi secara nirkabel dengan memanfaatkan gelombang RF 2,4GHz.



Gambar 3.3.1 *Arsitektur Sistem*

Node master merupakan node yang berfungsi sebagai media perantara untuk menerima data dari beberapa node slave, sehingga data yang dikumpulkan dapat diterima dan dikirimkan ke suatu penyimpanan sehingga data tersebut dapat diolah untuk mengetahui kondisi lingkungan yang ada.

Sementara itu node slave memiliki dua fungsi. Fungsi pertama adalah sebagai pengumpul data, dimana node slave mencatat perubahan akselerasi tanah menggunakan modul ADXL345 kemudian data tersebut dimasukkan ke dataset untuk nantinya diolah dalam proses penentuan threshold. Fungsi kedua adalah sebagai alat testing, setelah threshold ditemukan nilai tesebut akan dimasukkan ke node slave untuk dilakukan uji coba apakah sensor dapat menentukan apakah situasi yang terjadi pada saat simulasi adalah tanah longsor atau bukan.

3.4 Perancangan Node Master

Pada bagian ini akan dijelaskan mengenai node master, node master berfungsi sebagai node yang mengumpulkan data kondisi

lingkungan dari beberapa node slave yang akan diimplementasikan menggunakan mikrokontroller arduino uno. arduino akan terhubung dengan modul nRF24L01 untuk melakukan komunikasi nirkabel dengan node slave.

Pembahasan di sub bab ini juga menjelaskan mekanisme cara kerja Master node dalam mengumpulkan data dari slave node diagram alir.

3.4.1 Perancangan Rangkaian Utama

Rangkaian utama dari node master dan beberapa komponen-komponen penting untuk rancangannya akan dijelaskan dengan tabel 3.4.1.1 berikut ini.

Tabel 3.4.1.1 Tabel Komponen Node Master

No	Nama Komponen	Deskripsi	Jumlah
1	Arduino UNO	Merupakan sebuah mikrokontroller yang berfungsi sebagai komponen utama dan bisa disebut sebagai otak dari komponen itu sendiri, arduino yang digunakan adalah arduino uno rev 3.	1 buah
2	Modul nRF24L01	Merupakan modul komunikasi secara nirkabel yang memanfaatkan frekuensi gelombang radio 2,4GHz.	1 buah
3	*Breadboard*	Sebuah papan yang biasa disebut juga sebagai *project board* yang berfungsi untuk merangkai rangkaian elektronik.	1 buah
4	Kabel *Jumper*	Sebuah kabel elektrik yang menghubungkan	15-20 buah

		kompenen dan komponen lainnya.	
5	Kabel daya	Sebuah kabel elektrik yang menghubungkan Arduino dengan sumber daya.	1 buah

Setiap komponen-komponen akan dirangkai menjadi satu rangkaian utama sehingga menjadi satu rangkaian yaitu node master.

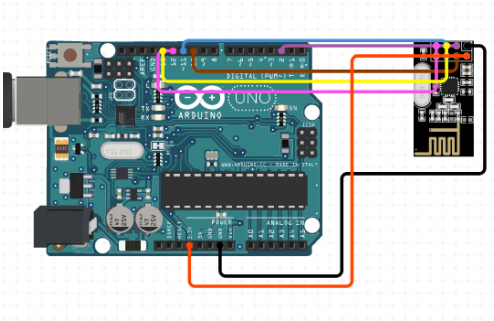
Berikut merupakan rancangan komponen untuk node master Pada komponen tersebut juga terdapat modul nRF24L01 dimana berfungsi sebagai modul komunikasi antara node master dan *node slave* dengan memanfaatkan frekuensi gelombang radio. Untuk terhubungnya koneksi pin antar komponen bisa dilihat berdasarkan tabel 3.4.1.3 berikut.

Tabel 3.4.1.1 Tabel Pin Antar Komponen Node Master

PIN	**Modul/Komponen**	**Deskripsi**
POWER		
5V	Breadboard (VCC)	Input Power
GND	Breadoard (GND)	Grounding
PIN		
9	nRF24L01	Terhubung dengan nRF24L01 pin CE
10	nRF24L01	Terhubung dengan nRF24L01 pin CSN
11	nRF24L01	Terhubung dengan nRF24L01 pin MOSI
12	nRF24L01	Terhubung dengan nRF24L01 pin MISO

13	nRF24L01	Terhubung dengan nRF24L01 pin SCK

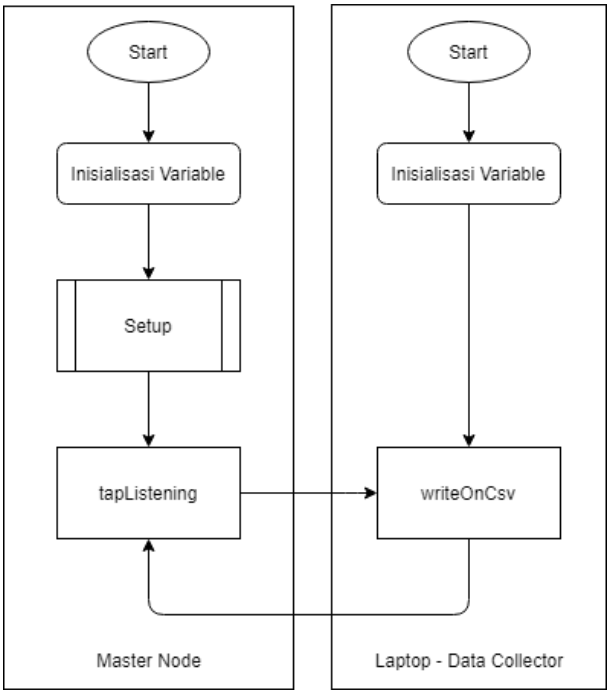
Setelah rangkaian dan komponen semuanya terhubung melalui rancangan seperti sebelumnya dijelaskan, maka rancangan rangkaian node master, dapat digambarkan seperti gambar 3.4.1 berikut.



Gambar 3.4.1 *Gambar Rangkaian Node Master*

3.4.2 Perancangan Diagram Alir Node Master- Data Collector

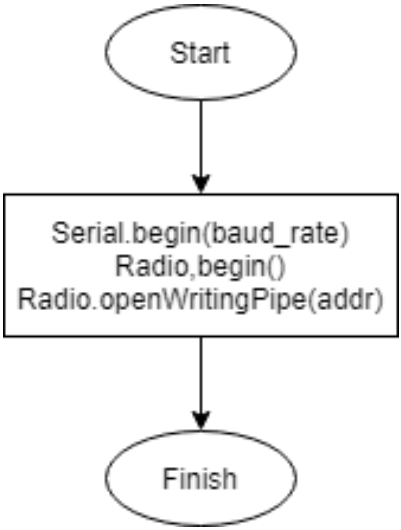
Diagram alir node master merupakan gambaran secara umum bagaimana node master bekerja sebagai data collector, yaitu cara kerja program yang dirancang sehingga node master dapat mengumpulkan data dari beberapa node slave untuk pengambilan keputusan.Diagram alir Node master sebagai data collector dapat dilihat melalui gambar 3.4.2.1.



Gambar 3.4.2 *Diagram Alir Node Master – Data collector*

Proses jalannya program pada Arduino terdapat subproses *setup* dan *tapListening*. Subproses setup berfungsi sebagai menginisiasi jalannya mikrokontroller yang berjalan hanya satu kali saat arduino dihidupkan, sedangkan *tapListening* yang berjalan di fungsi *loop* arduino berfungsi untuk mengeksekusi bagian-bagian program yang dijalankan secara berulang-ulang. Data sensor yang didapatkan dari node slave melalui *tapListening* kemudian disimpan dan dijadikan dataset menggunakan *writeOnCsv* yang dijalankan di laptop.

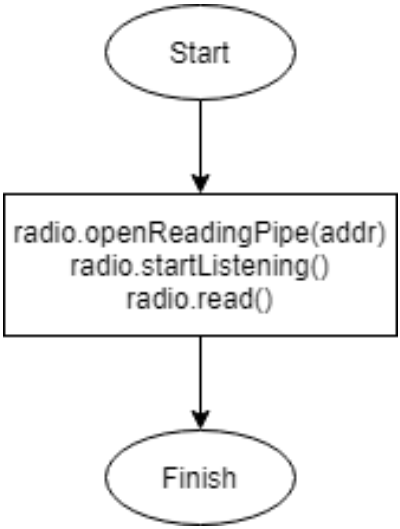
3.4.2.1 Diagram Alir Subproses Setup



Gambar 3.4.3 *Diagram Alir Subproses Setup*

Subproses setup berperan untuk menginisialisasi pertama kali ketika arduino dihidupkan atau tersambung dengan sumber daya listrik.Pertama akan program menjalankan *Serial.begin(baud_rate),* nilai *baud_rate* adalah nilai yang digunakan untuk menjalankan komponen, nilai yang digunakan adalah 9600, sedangkan Radio.begin untuk menginisialisasi nRF24L01 dan *Radio.openWritingPipe(address)* untuk menginisialisasi alamat alamat yang digunakan sebagai jalur komunkasi antara modul nrf. 3.4.3.2 merupakan diagram alir subproses setup.

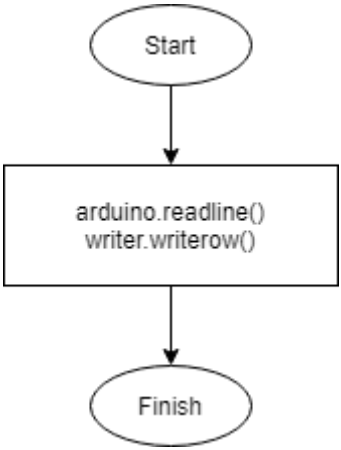
3.4.2.2 Diagram Alir Subproses tapListening



Gambar 3.4.4 *Diagram Alir subproses tapListening*

Subproses tapListening berperan menerima komunikasi dari modul nrf yang lain. radioOpenReadingPipe digunakan untuk membuka jalur komunikasi sesuai dengan address yang ditetapkan sebelumnya sementara radio. startListening untuk mulai menerima pesan yang dikirim. Radio.read berfungsi untuk membaca pesan yang diterima. gambar 3.4.2.4 merupakan diagram alir subproses tapListening.

3.4.2.3 Diagram Alir Subproses writeOnCsv

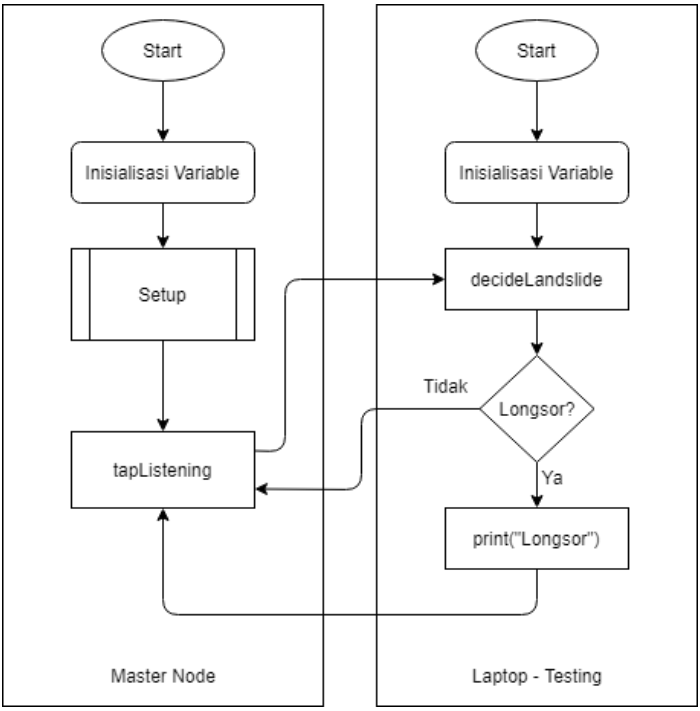


Gambar 3.4.5 *Diagram Alir Subproses writeOnCsv*

Subproses writeOnCsv berperan untuk membaca dan menyimpan data yang diterima oleh node master dari node slave ke dalam bentuk dataset. Dataset yang dibuat menyimpan id dari node serta nilai akselerasi pada axis x,y,z dari setiap node.
Perintah Arduino.readline() berfungsi untuk membaca data yang diterima oleh node master sementara writer.writerow untuk menyimpan data yang diterima. gambar 3.4.2.4 merupakan diagram alir subproses writeOnCsv.

3.4.3 Perancangan Diagram Alir Node Master - Testing

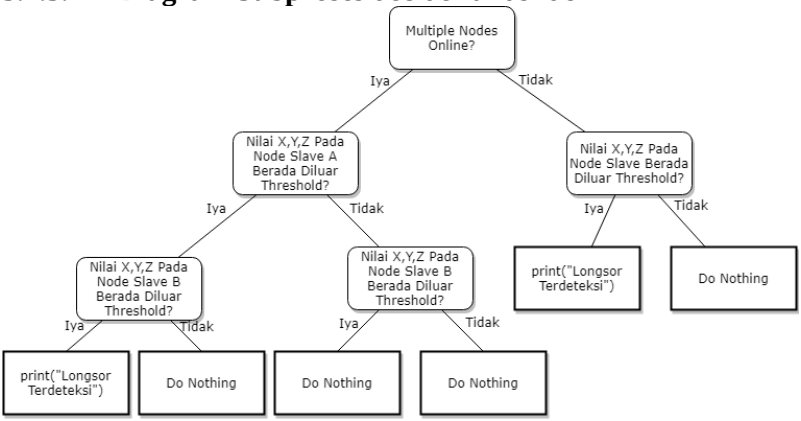
Diagram alir node master merupakan gambaran secara umum bagaimana node master bekerja ketika melakukan testing, yaitu cara kerja program yang dirancang untuk menentukan apakah kondisi yang terjadi adalah longsor atau tidak. Diagram alir Master Node sebagai Alat Testing dapat dilihat melalui gambar 3.4.3.1



Gambar 3.4.6 *Diagram Alir Node Master -Testing*

Alur jalannya program memiliki kemiripan dengan Node Master- Data collector. Perbedaannya adalah data yang diterima tidak disimpan di dataset namun dimasukkan ke fungsi decideLandslide untuk menentukan kondisi yang terjadi pada saat simulasi apakah longsor atau tidak.

3.4.3.1 Diagram Subproses decideLandslide



Gambar 3.4.3.3 *Diagram decision tree subproses decideLandslide*

Subproses decideLandslide berperan dalam menentukan kondisi pada saat simulasi apakah termasuk longsor atau tidak. Setelah mendapatkan nilai threshold, nilai tersebut diinisialisasikan untuk kemudian dilakukan pengecekan pada setiap node slave jika nodenya ada lebih dari satu. Ketika nilai pada semua node slave berada diluar threshold baik lebih rendah maupun lebih tinggi maka akan diputuskan bahwa kejadian tersebut adalah longsor. gambar 3.4.3.2 merupakan diagram decision tree subproses decideLandslide.

3.5 Perancangan Node Slave

Pada bagian ini akan dijelaskan mengenai node slave, node slave berfungsi sebagai node data kondisi lingkungan dan mengirimkannya ke node master. Node slave diimplementaskan menggunakan arduino yang terhubung dengan sensor ADXL345 untuk mengukur akselerasi pergerakan tanah. Proses pengiriman data ke node master menggunakan modul nRF24L01 sebagai media komunikasi dan powerbank sumber daya.

Pembahasan di sub bab ini juga menjelaskan mekanisme cara kerja node slave dalam mengumpulkan data dan mengirimkan data sensor dengan diagram alir.

3.5.1 Perancangan Rangkaian Node Slave

Rangkaian utama dari node slave dan beberapa komponen-komponen penting untuk rancangannya akan dijelaskan dengan tabel berikut.

Tabel 3.5.1.1 Tabel Komponen Node Slave

No	Nama Komponen	Deskripsi	Jumlah
1	Arduino UNO	Merupakan sebuah mikrokontroller yang berfungsi sebagai komponen utama dan bisa disebut sebagai otak dari komponen itu sendiri, arduino yang digunakan adalah arduino uno rev 3.	1 buah
2	Modul nRF24L01	Merupakan modul komunikasi secara nirkabel yang memanfaatkan frekuensi gelombang radio 2,4GHz.	1 buah
3	ADXL345	Merupakan sensor yang berfungsi untuk mengukur akselerasi pergerakan tanah.	1 buah
4	Powerbank	Merupakan sumber daya yang digunakan untuk menyalakan node slave.	1 buah
6	*Breadboard*	Sebuah papan yang biasa disebut juga sebagai *project board* yang	1 buah

		berfungsi untuk merangkai rangkaian elektronik.	
7	Kabel *Jumper*	Sebuah kabel elektrik yang menghubungkan kompenen dan komponen lainnya.	15-20 buah

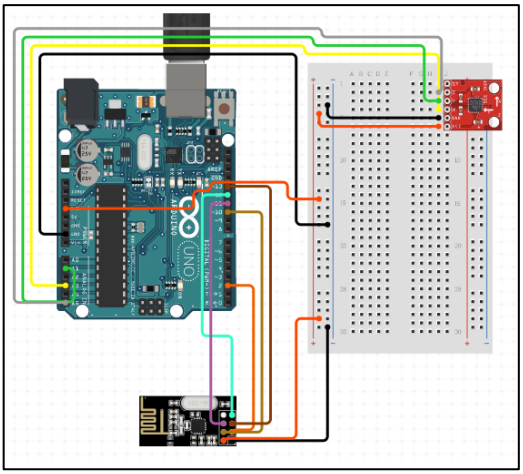
Setiap komponen-komponen akan dirangkai menjadi satu node slave, karena sistem membutuhkan 3 node slave maka, dari komponen tersebut membutuhkan 3 kali komponen tersebut. Berikut merupakan rancangan node slave setelah dirangkai menjadi satu node slave. Modul nRF24L01 digunakan untuk media komunikasi antara *node slave* dan *node master* dengan memanfaatkan frekuensi gelombang radio. Koneksi pin antar modul dapat dilihat pada tabel 3.5.1.2 berikut.

Tabel 3.5.1.3 Tabel koneksi pin antar komponen node slave

PIN	**Modul/Komponen**	**Deskripsi**
POWER		
5V	VCC	Power input dari arduino
GND	GND	Grounding
ANALOG IN		
A4	ADXL345	Terhubung dengan sensor ADXL345 pin SDA
A5	ADXL345	Terhubung dengan sensor ADXL345 pin SCL
PIN		
9	nRF24L01	Terhubung dengan nRF24L01 pin CE

10	nRF24L01	Terhubung dengan nRF24L01 pin CSN
11	nRF24L01	Terhubung dengan nRF24L01 pin MOSI
12	nRF24L01	Terhubung dengan nRF24L01 pin MISO
13	nRF24L01	Terhubung dengan nRF24L01 pin SCK

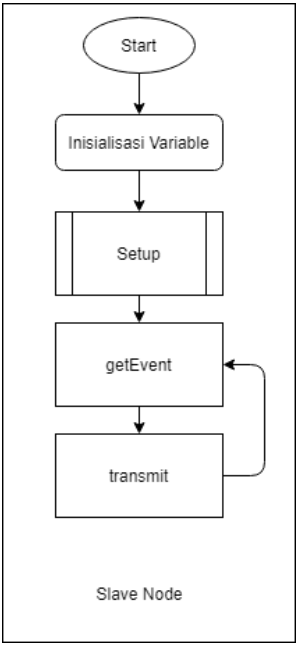
Setelah rangkaian dan komponen semuanya terhubung melalui rancangan seperti sebelumnya dijelaskan, maka rancangan rangkaian node slave, dapat digambarkan seperti gambar 3.5.1.1 berikut.



Gambar 3.5.1 *Gambar Rangkaian Node Slave*

3.5.2 Perancangan Diagram Alir Node slave

Diagram alir node slave merupakan gambaran secara umum bagaimana node slave bekerja, yaitu sebagai pengirim data yang didapatkan dari modul ADXL345. Diagram alir Slave Node dapat dilihat melalui gambar 3.5.2.1



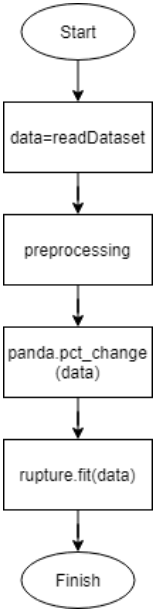
Gambar 3.5.2 *Diagram Alir slave node*

Subproses setup berfungsi sebagai menginisiasi jalannya mikrokontroller yang berjalan hanya satu kali saat arduino dihidupkan, sedangkan terdapat dua subproses utama yaitu subproses *getEvent* dan *transmit* yang berjalan pada fungsi loop untuk dieksekusi berulang kali. *updateSensor* yang berjalan di fungsi *loop* arduino berfungsi untuk mengeksekusi bagian-bagian program yang dijalankan secara berulang-ulang.

Dalam subproses getEvent terdapat 3 fungsi yang bertujuan untuk mendapatkan nilai akselerasi pada setiap axis dari

ADXL345. Sementara transmit berfungsi untuk mengirimkan data axis ke master node.

3.6 Perancangan Diagram Alir Fungsi findThreshold



Fungsi findThreshold digunakan untuk mencari nilai threshold axis x,y dan z untuk setiap slave node. Pertama, dataset di load kedalam suatu variable bernama data kemudian dilakukan preprocessing berupa data splitting. Hasil data splitting kemudian dimasukkan kedalam fungsi panda.pct_change untuk didapatkan perubahan nilai paling signifikan berdasarkan persentase perubahan nilai dari keseluruhan data. Lalu, dataset yang sebelumnya dimasukkan kedalam fungsi rupture.fit(data) untuk dilakukan cross-check nilai threshold menggunakan algoritma changing point detection dari library rupture. Berdasarkan hasil yang didapatkan dari

panda.pct_change dan rupture.fit inilah akan ditemukan nilai threshold untuk setiap axis

(***Halaman ini sengaja di kosongkan***)

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai implementasi sistem yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi berupa kode sumber untuk membangun program,spesifikasi hardwere. Implementasi yang dijelaskan dibagi menjadi lingkungan Implementasi perangkat keras dan lingkungan implementasi perangkat lunat pembangunan sistem.

4.1 Lingkungan Implementasi

Lingkungan Implementasi merupakan lingkungan dimana sistem akan dibangun. Lingkungan implementasi dibagi menjadi Lingkungan Implementasi Perangkat keras dan Lingkungan Implementasi Perangkat Lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Pada bagian ini dijelaskan perangkat keras yang digunakan untuk membangun sistem. Lingkungan implementasi perangkat keras yang akan dibangun secara lebih rinci dijelaskan pada Tabel 4.1dibawah ini.

Tabel 4.1.1.1 Tabel Lingkungan Implementasi Perangkat Keras

Perangkat	**Detail**
Perangkat Mikrokontroler	**Mikrokontroler:** • Atmega 328 **Model:** • Arduino UNO R3 **Tegangan:** • 5 – 12 V **Memory Flash:** • 32 KB **SRAM:**

	• 2KB
Perangkat Wireless Transceiver	**Model:** • nRF24L01+SingleChip2.4GHz Transceiver **Manufaktur:** • Nordic Semiconductor **Tegangan:** • 1.9-3.6V **Frekuensi:** • 2.4GHzISMBand **Interface:** • SPI **Ukuran:** • 20-pin4x4QFNPackage
Perangkat Monitoring Akselerasi Tanah	**Model:** • ADXL345 **Manufaktur:** • Sparkfun **Tegangan:** • Vcc 2.0 – 3.6 V **Ukuran:** • 3 mm × 5 mm × 1 mm

4.1.2 Lingkungan Implementasi Perangkat Lunak

Pada bagian ini akan dibahas mengenai perangkat lunak apa saja yang dibutuhkan untuk membangun sitem. Lingkungan implementasi perangkat lunak dari system yang akan dibangun secara lebih lengkap di jelaskan pada tabel 4.2 di bawah ini.

Tabel 4.1.2.1 Tabel Lingkungan Implemantasi Perangkat Lunak

Perangkat Lunak	Detail
Arduino IDE	Arduino IDE adalah sebuah IDE yang di gunakan untuk melakukan kompilasi terhadapa kode program Bahasa C untuk mikrokontroler Arduino serta digunakan juga untuk mengupload program ke dalam Arduino. Pada Arduino IDE juga bias melihat serial monitoring yang berfungsi untuk melihat hasil dari program yang berjalan pada arduino
Spyder	Spyder merupakan text editor yang digunakan untuk melakukan pemograman dengan Bahasa pemrograman python
Panda Library	Library yang digunakan untuk mencari changing point dari dataset yang telah dikumpulkan,
nRF24 Library	Library yang digunakan agar Arduino dapat berkomunikasi dengan modul nRF24L01. Library ini sudah mencakup fungsi-fungsi yang dibutuhkan seperti pengiriman data, pembacaan transmisi, ACK payloads, dan pengaturan kekuatan transmisi.

ADXL345 Library	Library yang digunakan agar Arduino dapat berkomunikasi dengan modul ADXL345. Library ini sudah mencakup fungsi fungsi yang di butuhkan untuk mendapatkan nilai accelerometer.
Panda Library	Library pada Bahasa pemrograman python yang berfungsi untuk analisis dan pengolahan data
Rupture Library	Library yang digunakan untuk melakukan cross-check hasil dari changing point value dari panda

4.2 Implementasi *Master Node*

Pada bagian ini akan dijelaskan implementasi mengenai Master node yang telah diimplementasikan melalui perancangan Master node yang di jelaskan di bab 3.4. Implementasi Master Node meliputi implementasi rangkaian dan implementasi fungsi master node yaitu program yang dibuat

4.3 Implementasi Code Program *getDataset*

Pada subbab ini akan di implementasikan code program *getDataset* ke master node. Dalam implementasi code program akan di bagi dalam beberapa bagian dan selanjutnya akan dijelaskan pada subbab-subbab tersendiri.

4.3.1 Global Variabel dan Type

Subbab ini membahas variable dan type data yang digunakan oleh *master node* untuk menunjang fungsi-fungsi lain dalam program.

```
Radio ← new RF24(9,10)


rxAddr ← 5000
```

Kode Sumber 4.1 Global dan type variable master node

Pada kode sumber 4.1 terdapat deklarasi address yang digunakan dalam transmisi yaitu rxAddr , Juga terdapat deklarasi untuk CE dan CSN yang disesuaikan dengan pin nRF24l01 pada board Arduino.

4.3.2 Fungsi Setup

Fungsi ini adalah fungsi yang akan selalu dijalankan minimal satu kali setiap Arduino dinyalakan. Fungsi ini berguna untuk setup variabel atau apapun sebelum arduino menjalankan fungsi loop yang akan selalu dijalankan selama Arduino hidup.

```
Function setup ():
        call serial. begin ()
        call radio. begin ()
    call radio. openreadingpipe (0,rxAddr)
    call radio. Startlistening()
```

Kode Sumber 4.2 Fungsi Setup

Saat Arduino menyala Arduino akan menjalankan fungsi setup maka akan mengaktifkan radio nRF.

4.3.3 Fungsi Transmit

Fungsi transmisi adalah fungsi yang berisi instruksi untuk nRF agar melakukan transmisi data ke alamat yang ditujukan.

```
Function transmit (msg) :

        call radio. stopListening ()
        call radio. openWritingPipe (addr)

        call radio. write(msg, call sizeof (msg))
    call radio. openreadingpipe (0,rxAddr)
    call radio. Startlistening()
```

Kode Sumber 4.3 Fungsi Transmisi

Pada fungsi transmisi, sebelum melakukan pengiriman nRF diinstruksikan untuk berhenti melakukan *listening* dikarenakan nRF tidak bias melakukan *listening* dan *transmit* diwaktu yag sama. Kemudian nRF diberikan alamat node yang akan dikirimkan pesan, dan fungsi *write* menginstruksikan nRF untuk menulis pesan dan melakukan *transmisi*.setelah selesai melakukan transmisi nrf diinstruksikan untk melakukan listening kembali

4.3.4 Fungsi *tapListening*

Fungsi tap*listening* adalah fungsi yang berisi instruksi untuk nRF agar menerima pesan yang telah dikirimkan oleh node lain.

```
function taplistening () :

        if radio. available :
        call radio. read (msg, call sizeof (msg))

        return true
```

```
return false
```

Kode Sumber 4.4 Fungsi *Listening*

Pada Kode Sumber 4.3 fungsi *listening*, Program akan melakukan pengechekan apakah ada transmisi yang masuk, fungsi akan mengembalikan nilai *True* apabila dia menerima transmisi, kemudian akan dibaca data yang di terimanya, sebaliknya jika tidak menerima transmisi fungsi akan mengembalikan nilai *false*.

4.3.5 Fungsi loop

Fungsi ini adalah fungsi yang akan selalu dijalankan berulang-ulang selama Arduino beroperasi. Fungsi loop merupakan fungsi utuma pada Arduino

```
Function loop():

        Call tapListening()
    Call transmit(msg)
```

Kode Sumber 4.5 Fungsi loop

Fungsi loop memanggil fungsi taplistening untuk menerima data yang dikirim dari slave node. Sementara fungsi loop hanya dipanggil ketika akan melakukan testing koneksi antara node slave dan master

4.4 Implementasi *slave node*

Pada bagian ini akan di jelaskan implementasi slave node. Seperti yang di jelaskan sebelumnya *slave node* berperan sebagai pengumpul data dan alat testing pada saat simulasi. *slave node* melakukan pemanggilan fungsi *getEvent* dan *transmit* . Rangkaian yang di gunakan oleh *slave node* mengikuti rancangan yang telah dijelaskan pada bagian 3.5. Rangkaian akan terlihat seperti pada gambar 4.2

4.5 Implementasi Code Program *slave node*

Pada bab ini akan di jelaskan implementasi code program yang di gunakan pada *slave node*. Dalam implementasi code pada *slave node* fungsi yang digunakan lebih banyak dari pada master node, dikarenakan pada *slave node* perlu melakukan getEvent untuk mendapatkan data accelerometer dan *transmit* untuj mengirimkan datanya ke master node, code dibagi dalam 3 bagian yaitu bagian setup,bagian transmit dan bagian loop. Setiap implementasi code program akan dibahas dalam subbab berikut.

4.5.1 Global Variabel dan Type

Subbab ini membahas variable dan type data yang digunakan oleh *slave node* untuk menunjang fungsi-fungsi lain dalam program.

```
rxAddr ← 5000
accel ← new Adafruit_ADXL345_Unified(12345)
radio ← new RF24 (9,10)
```

Kode Sumber 4.6 Global Variable Slave Node

Pada kode sumber diatas rxAddr adalah alamat yang akan digunakan sebagai channel komunikasi, sementara untuk variable accel untuk menginisialisasikan modul ADXL345. Radio digunakan untuk menginisialisasikan modul *wireless* nRF24L01 berada pada pin 9 dan 10 pada *mikrokontroller* Arduino.

4.5.2 Fungsi Setup

Fungsi ini adalah fungsi yang pasti akan selalu dilanja minimal satu kali setiap Arduino dinyalakan. Fungsi ini berguna untuk setup variable atau apapun sebelum Arduino menjalankan fungsi loop yang akan selalu berjalan selama Arduino hidup.

```
function setup ():
```

```
call radio. begin ()
call accel.setRange(ADXL345_RANGE_16_G)
call radio. openReadingPipe (0, sink_addr)
call radio. startListening ()
```

Kode Sumber 4.7 Fungsi Setup Pada Slave node

Fungsi ini bertujuan untuk inisialisasi variable,nRF dan range untuk ADXL345. Karena *slave node* mendapat data dari ADXL345 setelah itu mengirimkannya ke master node. Maka pada code node slave di set untuk melakukan *transmit* ke alamat yang telah di set pada global variable sejak *slave node* dihidupkan.

4.5.3 Fungsi Loop

Fungsi ini adalah fungsi yang akan selalu dijalankan berulang-ulang selama Arduino beroperasi. Fungsi loop merupakan fungsi utuma pada Arduino. Semua perintah yang akan dilaksanakan di instruksikan dalam fungsi ini.

```
function loop () :
    sensors_event_t event
    call accel.getEvent(event)
        current_x <− event.acceleration.x
     current_y <− event.acceleration.y
     current_z <− event.acceleration.z
                    if radio . available :
                    transmit (current_x, current_y, current_z)
```

Kode Sumber 4.6 Fungsi Loop pada slave node

Fungsi ini bertujuan agar *slave node* mengirimkan kondisi pada axis x,y dan z dari ADXL345 ke *master node.* Event.acceleration.x berfungsi untuk mendapat nilai sumbu x pada ADXL345 begitu juga dengan sumbu y dan z. Hal ini akan dilakukan selama Arduino masih beroperasi.

4.6 Implementasi code program findThreshold

Pada subbab ini akan di implementasikan code program *findThreshold* ke master node. Dalam implementasi code program akan di bagi dalam beberapa bagian dan selanjutnya akan dijelaskan pada subbab-subbab tersendiri.

4.6.1 Global Variabel dan Type

Subbab ini membahas variable dan type data yang digunakan oleh *findThreshold* untuk menunjang fungsi-fungsi lain dalam program.

```
dataset ← csv.DictReader(open(“dataset.csv”))
```

Kode Sumber 4.9 Global dan type variable findThreshold

Pada kode sumber 4.1 terdapat deklarasi dataset yang digunakan dalam pencarian nilai threshold yaitu variable dataset.

4.6.2 Preprocessing

Subbab ini membahas variable dan type data yang digunakan oleh *findThreshold* untuk menunjang fungsi-fungsi lain dalam program.

```
for row in dataset:
    for column, value in row.items():
        data_split.setdefault(column, []).append(value)

x_axis ← list(data_split ["x"])
y_axis ← list(data_split ["y"])
z_axis ← list(data_split ["z"])
```

Kode Sumber 4.7 Preprocessing findThreshold

Pada kode sumber 4.1 terdapat metode preprocessing yang diterapkan pada dataset yang sebelumnya telah diload yaitu dengan memasukkannya kedalam bentuk list kemudian dipisahkan sesuai dengan axisnya. Hal ini dilakukan untuk mempermudah proses pengolahan data di tahap selanjutnya.

4.6.3 Fungsi findThreshold

Subbab ini membahas fungsi utama dari *findThreshold* untuk menunjang fungsi-fungsi lain dalam program.

```
x_series ← pd.Series(x_axis)
y_series ← pd.Series(y_axis)
z_series ← pd.Series(z_axis)

changes_x ← x_series.pct_change()
changes_y ← y_series.pct_change()
changes_z ← z_series.pct_change()

changes_x_list←x_series[x_series.pct_change() > 1].index.tolist()

changes_y_list←y_series[y_series.pct_change() > 1].index.tolist()

changes_z_list←z_series[x_series.pct_change() > 1].index.tolist()
```

Kode Sumber 4.8 Fungsi utama findThreshold

Pada kode sumber 4.1 terdapat fungsi utama findThreshold. Dengan menggunakan fungsi pct_change pada library panda dapat ditemukan perubahan nilai paling signifikan

pada setiap axis dalam dataset. Setelah itu titik perubahan atau changing point dibuat menjadi sebuah list berisi index data tersebut pada dataset berdasarkan perubahan yang paling signifikan, dalam hal ini perubahan diatas 100% dari nilai yang lain.

Kemudian dataset yang sama dijalankan menggunakan library rupture untuk dilakukan pengecekan ulang titik changing point. Jika hasilnya sama antara menggunakan panda dan rupture baru kemudian nilai tersebut diambil sebagai nilai threshold. Nilai threshold juga perlu menyesuaikan dengan offset dari modul ADXL sehingga nilai didapatkan perlu dilakukan pengurangan atau penjumlahan.

4.7 Fungsi decideLandslide

Subbab ini membahas fungsi *decideLandslide* untuk menentukan apakah suatu kondisi termasuk longsor atau bukan.

```
data ← arduino.readline()[:-2] #ignore eol
data←data.split(",")
flag_a←0
flag_b←0
slave_id←data[0]
current_x←data[1]
current_y←data[2]
current_z←data[3]

if  slave_a in slave_id:
            if (Nilai X,Y dan Z Diluar Threshold):
                        flag_a=1
if  slave_b in slave_id:
            if (Nilai X,Y dan Z Diluar Threshold):
                        flag_b=1
if flag_a == 1 and flag_b ==1:
            print("longsor")
```

Kode Sumber 4.9 Fungsi decideLandslide

Pada kode sumber 4.1 dapat dilihat setelah nilai threshold didapatkan, nilai tersebut digunakan untuk batas atas dan batas bawah nilai sensor untuk setiap axis dan slave node. Dalam contoh kode diatas menggunakan kasus dua slave node yaitu slave_a dan slave_b. Ketika nilai yang didapatkan slave_a dan slave_b berada diluar threshold yang didapatkan barulah peringatan longsor didapatkan. Maksud dari berada diluar nilai threshold adalah Ketika nilai sensor lebih kecil daripada batas bawah atau lebih besar daripada batas atas. Program ini mengimplementasi decision tree yang telah dirancangan pada bab sebelumnya.

BAB V
UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai scenario uji coba yang dilakukan dan evaluasi terhadap nilai threshold yang telah didapatkan. Hasil uji coba didapatkan dari implementasi yang dijelaskan pada Bab 4 dengan scenario yang berbeda. Bab ini berisikan pembahsan mengenai lingkungan pengujian data pengujian, uji kinerja dan hasil pengujian yang digunakan untuk Bab selanjutnya.

5.1 Lingkungan Pengujian

Lingkungan pengujian *system* pengiriman data dan simulasi monitoring tanah longsor dilakukan dalam 5 skenario ujicoba yaitu longsor, node slave digerakkan secara manual dengan lambat, node digerakkan manual dengan cepat dan node slave digerakkan secara manual dengan tempo yang berubah-ubah juga pada saat sensor terjatuh..Skenario dimulai dengan pengambilan data hingga simulasi dilakukan dengan menggunakan metode seperti yang sudah dibahas pada bab III dalam tugas akhir ini. Untuk mensimulasikan tanah longsor digunakan pasir yang dipasangkan dengan karung semen dibagian bawahnya untuk nantinya akan ditarik dengan begitu pergerakan pasir akan menyerupai tanah longsor sungguhan. Juga dengan memastikan pergerakan tanah lebih besar atau sama dengan 1g yg merupakan akselerasi dari tanah longsor sungguhan.

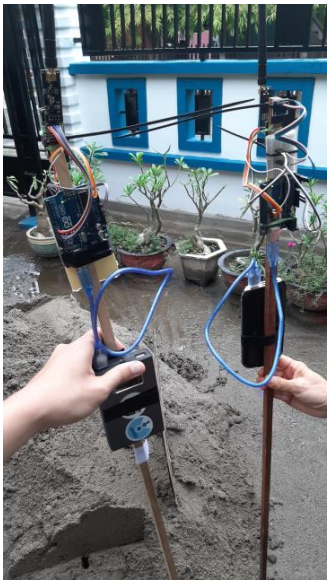
Gambar 5.1.1 Simulasi longsor

Gambar 5.1.2 Lokasi dan Posisi Uji Coba Dataset A

Gambar 5.1.3 Lokasi dan Posisi Uji Coba Dataset B

Setiap pengujian scenario dilakukan di halaman depan rumah penulis, jarak antar *slave node* dengan *master node* kurang lebih 10 meter. Dalam setiap skenario uji coba akan menggunakan dataset yang sama kemudian dilakukan berulang kali dengan dataset yang berbeda setelah melakukan semua skenario.Dan masing-masing *node* di beri jarak kurang lebih 30 sampai dengan 50 centimeter.

Gambar 5.1.4 Lingkungan pengujian node slave A



Gambar 5.1.5 Node Slave A dan Slave B

Gambar 5.1.6 Lingkungan pengujian *master node*

Gambar 5.1.7 Powerbank

Dalam lingkungan uji coba *slave node* akan memakai powerbank berbeda namun satu jenis yang sama yaitu powerbank berbentuk kotak dengan voltase 5V dan berkapasitas 10.000mAh. Powerbank ini dipilih karena mampu mentenagai Arduino beserta

modul nRF24L01 cukup lama dan dapat di charger ulang sehingga dapat digunakan berulang.

5.2 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan scenario yang akan digunakan dalam proses uji coba. Dengan scenario yang dibuat kita akan menguji apakah perangkat yang di buat sudah berjalan sesuai dengan yang di rancang dan benar, dan menentukan performa pada masing-masing scenario. Pada uji coba ini kita akan menguji akurasi dari threshold yang telah didapatkan. Terdapat 4 macam scenario uji coba sebagai berikut:

1. Pengujian akurasi saat kondisi yang disimulasikan adalah benar tanah longsor
2. Pengujian akurasi saat kondisi yang disimulasikan node yang digerakkan secara manual dengan tempo yang lambat
3. Pengujian akurasi saat kondisi yang disimulasikan node yang digerakkan secara manual dengan tempo yang cepat
4. Pengujian akurasi saat kondisi yang disimulasikan node yang digerakkan secara manual dengan tempo yang berubah ubah
5. Pengujian akurasi saat kondisi yang disimulasikan node yang sedang dalam posisi tidak tegak (terjatuh)

Dalam melakukan uji coba akan dilakukan pergantian dataset setelah melakukan semua scenario juga dilakukan kalibrasi nilai sensor sebelum melakukan pengujian ulang. Sehingga sebelum melakukan simulasi akan dilakukan pengambilan dataset dan pencarian nilai threshold terlebih dahulu. Pengambilan dataset sendiri dilakukan dengan cara memasangkan node slave pada tumpukan pasir kemudian mensimulasikan longsor pada pasir tersebut.

Pencarian nilai threshold dilakukan berulang ulang dikarenakan karakteristik sensor ADXL345 yang sangat sensitif sehingga setiap akan melakukan perubahan posisi nilainya akan selalu berubah, maka dari itu perlu dilakukan pengambilan dataset berulang kali setiap memindahkan posisi sensor.

5.2.1 Skenario Uji Coba 1 pada Dataset A

Dalam scenario uji coba yang pertama kita akan menghitung akurasi dari threshold yang didapatkan dengan kondisi simulasi benar adalah longsor. Setelah melakukan pencarian nilai threshold, didapatkan nilai threshold sesuai pada tabel berikut.

Tabel 5.2.1.1 *Tabel Nilai Threshold Dataset A*

	Slave_A	**Slave_B**
Upper_X_Threshold	-5,29	0,08
Lower_X_Threshold	-5,53	-1,96
Upper_Y_Threshold	-10,3	-10,23
Lower_Y_Threshold	-10,71	-10,45
Upper_Z_Threshold	7,59	2,2
Lower_Z_Threshold	6,51	0,55

Dari hasil trial yang dilakukan dengan scenario kondisi longsor yang sebenarnya dapat dilihat pada tabel hasil uji coba dibawah ini.

Tabel 5.2.1.2 *Tabel Hasil Uji Coba Skenario 1 - Dataset A*

Trial	**Kondisi Sebenarnya**	**Kondisi yang didapatkan**
1	Longsor	Longsor
2	Longsor	Longsor
3	Longsor	Longsor

4	Longsor	Tidak Longsor
5	Longsor	Tidak Longsor
6	Longsor	Tidak Longsor
7	Longsor	Longsor
8	Longsor	Tidak Longsor
9	Longsor	Longsor
10	Longsor	Longsor

5.2.2 Skenario Uji Coba 2 pada Dataset A

Dalam scenario uji coba yang kedua kita akan menghitung akurasi dari threshold yang didapatkan dengan kondisi simulasi node slave digerakkan manual dengan tempo lambat. Setelah melakukan pencarian nilai threshold, didapatkan nilai threshold sesuai pada tabel berikut.

Tabel 5.2.2.1 *Tabel Nilai Threshold Dataset A*

	Slave_A	**Slave_B**
Upper_X_Threshold	-5,29	0,08
Lower_X_Threshold	-5,53	-1,96
Upper_Y_Threshold	-10,3	-10,23
Lower_Y_Threshold	-10,71	-10,45
Upper_Z_Threshold	7,59	2,2
Lower_Z_Threshold	6,51	0,55

Dari hasil trial yang dilakukan dengan scenario kondisi tidak longsor namun node digerakkan dengan tempo lambat secara manual dapat dilihat pada tabel hasil uji coba dibawah ini.

Tabel 5.2.2.2 *Tabel Hasil Uji Coba Skenario 2 - Dataset A*

Trial	**Kondisi Sebenarnya**	**Kondisi yang didapatkan**
1	Tidak Longsor	Tidak Longsor

2	Tidak Longsor	Tidak Longsor
3	Tidak Longsor	Longsor
4	Tidak Longsor	Tidak Longsor
5	Tidak Longsor	Tidak Longsor
6	Tidak Longsor	Longsor
7	Tidak Longsor	Longsor
8	Tidak Longsor	Tidak Longsor
9	Tidak Longsor	Tidak Longsor
10	Tidak Longsor	Tidak Longsor

5.2.3 Skenario Uji Coba 3 pada Dataset A

Dalam scenario uji coba yang ketiga kita akan menghitung akurasi dari threshold yang didapatkan dengan kondisi simulasi node slave digerakkan manual dengan tempo cepat. Setelah melakukan pencarian nilai threshold, didapatkan nilai threshold sesuai pada tabel berikut.

Tabel 5.2.3.1 *Tabel Nilai Threshold Dataset A*

	Slave_A	**Slave_B**
Upper_X_Threshold	-5,29	0,08
Lower_X_Threshold	-5,53	-1,96
Upper_Y_Threshold	-10,3	-10,23
Lower_Y_Threshold	-10,71	-10,45
Upper_Z_Threshold	7,59	2,2
Lower_Z_Threshold	6,51	0,55

Dari hasil trial yang dilakukan dengan scenario kondisi tidak longsor namun node digerakkan dengan tempo lambat secara manual dapat dilihat pada tabel hasil uji coba dibawah ini.

Tabel 5.2.3.2 Tabel Hasil Uji Coba Skenario 3 - Dataset A

Trial	Kondisi Sebenarnya	Kondisi yang didapatkan
1	Tidak Longsor	Longsor
2	Tidak Longsor	Longsor
3	Tidak Longsor	Longsor
4	Tidak Longsor	Tidak Longsor
5	Tidak Longsor	Longsor
6	Tidak Longsor	Longsor
7	Tidak Longsor	Longsor
8	Tidak Longsor	Longsor
9	Tidak Longsor	Tidak Longsor
10	Tidak Longsor	Longsor

5.2.4 Skenario Uji Coba 4 pada Dataset A

Dalam scenario uji coba yang keempat kita akan menghitung akurasi dari threshold yang didapatkan dengan kondisi simulasi node slave digerakkan manual dengan tempo berubah-ubah. Setelah melakukan pencarian nilai threshold, didapatkan nilai threshold sesuai pada tabel berikut.

Tabel 5.2.4.1 Tabel Nilai Threshold Dataset A

	Slave_A	Slave_B
Upper_X_Threshold	-5,29	0,08
Lower_X_Threshold	-5,53	-1,96
Upper_Y_Threshold	-10,3	-10,23
Lower_Y_Threshold	-10,71	-10,45
Upper_Z_Threshold	7,59	2,2
Lower_Z_Threshold	6,51	0,55

Dari hasil trial yang dilakukan dengan scenario kondisi tidak longsor namun node digerakkan dengan tempo berubah-ubah secara manual dapat dilihat pada tabel hasil uji coba dibawah ini.

Tabel 5.2.4.2 *Tabel Hasil Uji Coba Skenario 4 - Dataset A*

Trial	Kondisi Sebenarnya	Kondisi yang didapatkan
1	Tidak Longsor	Tidak Longsor
2	Tidak Longsor	Longsor
3	Tidak Longsor	Longsor
4	Tidak Longsor	Longsor
5	Tidak Longsor	TidakLongsor
6	Tidak Longsor	Longsor
7	Tidak Longsor	Longsor
8	Tidak Longsor	Tidak Longsor
9	Tidak Longsor	Longsor
10	Tidak Longsor	Longsor

5.2.5 Skenario Uji Coba 1 pada Dataset B

Dalam scenario uji coba yang pertama kita akan menghitung akurasi dari threshold yang didapatkan dengan kondisi simulasi benar adalah longsor. Setelah melakukan pencarian nilai threshold, didapatkan nilai threshold sesuai pada tabel berikut.

Tabel 5.2.5.1 *Tabel Nilai Threshold Dataset B*

	Slave_A	Slave_B
Upper_X_Threshold	-6,86	0,16
Lower_X_Threshold	-7,14	-0,15
Upper_Y_Threshold	-10,3	-10,35

Lower_Y_Threshold	-10,75	-10,47
Upper_Z_Threshold	7,53	1.80
Lower_Z_Threshold	6,67	1.10

Dari hasil trial yang dilakukan dengan scenario kondisi longsor yang sebenarnya dapat dilihat pada tabel hasil uji coba dibawah ini.

Tabel 5.2.55.2.5.2 *Tabel Hasil Uji Coba Skenario 1 - Dataset B*

Trial	Kondisi Sebenarnya	Kondisi yang didapatkan
1	Longsor	Tidak Longsor
2	Longsor	Longsor
3	Longsor	Longsor
4	Longsor	Longsor
5	Longsor	Tidak Longsor
6	Longsor	Longsor
7	Longsor	Longsor
8	Longsor	Tidak Longsor
9	Longsor	Longsor
10	Longsor	Tidak Longsor

5.2.6 Skenario Uji Coba 2 pada Dataset B

Dalam scenario uji coba yang kedua kita akan menghitung akurasi dari threshold yang didapatkan dengan kondisi simulasi node slave digerakkan manual dengan tempo lambat. Setelah melakukan pencarian nilai threshold, didapatkan nilai threshold sesuai pada tabel berikut.

Tabel 5.2.6.1 *Tabel Nilai Threshold Dataset B*

	Slave_A	Slave_B
Upper_X_Threshold	-6,86	0,16

Lower_X_Threshold	-7,14	-0,15
Upper_Y_Threshold	-10,3	-10,35
Lower_Y_Threshold	-10,75	-10,47
Upper_Z_Threshold	7,53	1.80
Lower_Z_Threshold	6,67	1.10

Dari hasil trial yang dilakukan dengan scenario kondisi tidak longsor namun node digerakkan dengan tempo lambat secara manual dapat dilihat pada tabel hasil uji coba dibawah ini.

Tabel 5.2.6.2 *Tabel Hasil Uji Coba Skenario 2 - Dataset B*

Trial	Kondisi Sebenarnya	Kondisi yang didapatkan
1	Tidak Longsor	Longsor
2	Tidak Longsor	Tidak Longsor
3	Tidak Longsor	Tidak Longsor
4	Tidak Longsor	Longsor
5	Tidak Longsor	Tidak Longsor
6	Tidak Longsor	Tidak Longsor
7	Tidak Longsor	Longsor
8	Tidak Longsor	Tidak Longsor
9	Tidak Longsor	Tidak Longsor
10	Tidak Longsor	Longsor

5.2.7 Skenario Uji Coba 3 pada Dataset B

Dalam scenario uji coba yang ketiga kita akan menghitung akurasi dari threshold yang didapatkan dengan kondisi simulasi node slave digerakkan manual dengan tempo cepat. Setelah melakukan pencarian nilai threshold, didapatkan nilai threshold sesuai pada tabel berikut.

Tabel 5.2.7.1 Tabel Nilai Threshold Dataset B

	Slave_A	Slave_B
Upper_X_Threshold	-6,86	0,16
Lower_X_Threshold	-7,14	-0,15
Upper_Y_Threshold	-10,3	-10,35
Lower_Y_Threshold	-10,75	-10,47
Upper_Z_Threshold	7,53	1.80
Lower_Z_Threshold	6,67	1.10

Dari hasil trial yang dilakukan dengan scenario kondisi tidak longsor namun node digerakkan dengan tempo lambat secara manual dapat dilihat pada tabel hasil uji coba dibawah ini.

Tabel 5.2.7.2 Tabel Hasil Uji Coba Skenario 3 - Dataset B

Trial	Kondisi Sebenarnya	Kondisi yang didapatkan
1	Tidak Longsor	Longsor
2	Tidak Longsor	Tidak Longsor
3	Tidak Longsor	Longsor
4	Tidak Longsor	Longsor
5	Tidak Longsor	Longsor
6	Tidak Longsor	Longsor
7	Tidak Longsor	Tidak Longsor
8	Tidak Longsor	Longsor
9	Tidak Longsor	Longsor
10	Tidak Longsor	Longsor

5.2.8 Skenario Uji Coba 4 pada Dataset B

Dalam scenario uji coba yang keempat kita akan menghitung akurasi dari threshold yang didapatkan dengan kondisi simulasi node slave digerakkan manual dengan tempo berubah-ubah.

Setelah melakukan pencarian nilai threshold, didapatkan nilai threshold sesuai pada tabel berikut.

Tabel 5.2.8.1 *Tabel Nilai Threshold Dataset B*

	Slave_A	**Slave_B**
Upper_X_Threshold	-6,86	0,16
Lower_X_Threshold	-7,14	-0,15
Upper_Y_Threshold	-10,3	-10,35
Lower_Y_Threshold	-10,75	-10,47
Upper_Z_Threshold	7,53	1.80
Lower_Z_Threshold	6,67	1.10

Dari hasil trial yang dilakukan dengan scenario kondisi tidak longsor namun node digerakkan dengan tempo berubah-ubah secara manual dapat dilihat pada tabel hasil uji coba dibawah ini.

Tabel 5.2.8.2 *Tabel Hasil Uji Coba Skenario 4 - Dataset B*

Trial	**Kondisi Sebenarnya**	**Kondisi yang didapatkan**
1	Tidak Longsor	Tidak Longsor
2	Tidak Longsor	Tidak Longsor
3	Tidak Longsor	Longsor
4	Tidak Longsor	Longsor
5	Tidak Longsor	Longsor
6	Tidak Longsor	Longsor
7	Tidak Longsor	Longsor
8	Tidak Longsor	Longsor
9	Tidak Longsor	Tidak Longsor
10	Tidak Longsor	Longsor

5.2.9 Skenario Uji Coba 5

Dalam scenario uji coba yang pertama kita akan menghitung akurasi dari threshold yang didapatkan dengan kondisi simulasi benar adalah longsor. Setelah melakukan pencarian nilai threshold, didapatkan nilai threshold sesuai pada tabel berikut.

Tabel 5.2.9.1 *Tabel Nilai Threshold Uji Coba 5*

	Slave_A	**Slave_B**
Upper_X_Threshold	-6,86	-0.94
Lower_X_Threshold	-13,73	-1.57
Upper_Y_Threshold	-2,71	-3,10
Lower_Y_Threshold	-3,65	-3,30
Upper_Z_Threshold	17,61	-7.25
Lower_Z_Threshold	13,49	-7,65

Dari hasil trial yang dilakukan dengan scenario kondisi longsor yang sebenarnya dapat dilihat pada tabel hasil uji coba dibawah ini.

Tabel 5.2.9.2 *Tabel Hasil Uji Coba 5*

Trial	**Kondisi Sebenarnya**	**Kondisi yang didapatkan**
1	Longsor	Tidak Longsor
2	Longsor	Longsor
3	Longsor	Longsor
4	Longsor	Longsor
5	Longsor	Tidak Longsor
6	Longsor	Longsor
7	Longsor	Tidak Longsor

8	Longsor	Longsor
9	Longsor	Longsor
10	Longsor	Tidak Longsor

5.3 Evaluasi Umum Skenario Uji Coba

Berikut adalah evaluasi dari hasil uji coba dari masing skenario. Terdapat point yang akan di evaluasi yaitu akurasi juga pengaruh lingkungan terhadap hasil yang didapatkan.

5.3.1 Akurasi

Dari uji coba yang di lakukan terjadi penurunan akurasi Ketika sensor digerakkan dengan cepat hal ini disebabkan karena pergerakan tersebut melewati batas thresholding yang telah ditetapkan.Juga karena kedua slave digerakan juga dapat melewati threshold yang mengharuskan kedua node yang berada pada kondisi yang sama untuk mendapatkan kondisi “longsor” yang benar. Maka dari itu, uji coba sebaiknya dilakukan pada kondisi lingkungan yang minim gangguan dari luar.

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang diperoleh dari tugas akhir yang telah dikerjakan dan saran terkait pengembangan dari tugas akhir ini yang dapat dilakukan pada masa yang akan datang.

6.1 Kesimpulan

Kesimpulan yang diperoleh dari hasil uji coba dan evaluasi pada tugas akhir ini adalah sebagai berikut:

1. Dengan menggunakan metode pencarian thresholding menggunakan changing point dengan bantuan panda dan rupture dapat dengan cepat ditemukan setelah melakukan pengambilan data.
2. Metode yang digunakan penulis memiliki akurasi maksimal 60% dikarenakan tidak diterapkannya adaptive thresholding. hal ini menyebabkan pencarian threshold harus dilakukan berulang kali untuk setiap perubahan posisi slave node.
3. Metode monitoring tanah longsor yang digunakan penulis dapat diterapkan pada lingkungan yang minim interferensi dari luar apabila jarak antar node masih dalam jangkauan radio modul nRF24L01.

6.2 Saran

Saran yang diberikan dari hasil uji coba dan evaluasi pada tugas akhir ini adalah sebagai berikut:

1. Mengimplementasikan *adaptive thresholding.* Dengan begitu thresholding yang didapatkan dapat menyesuaikan setiap kali node dipindahkan dan tidak perlu melakukan pencarian thresholding ulang
2. Akurasi dapat ditingkatkan dengan menambahkan jumlah slave node juga menggunakan decision tree yang sama. Dengan begitu setiap node dapat saling mengoreksi hasil decision making

(Halaman ini sengaja dikosongkan)

Daftar Pustaka

[1] H. Z. Kotta, K. Rantelobo, S. Tena and G. Klau, "Wireless Sensor Network for Landslide Monitoring," *TELKOMNIKA*, vol. 9, no. 1, pp. 9-18 , 2011.

[2] C. Truong, L. Oudre and N. Laurent , "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, no. 107299, 2020.

[3] B. K. Raharjo, Rancang Bangun Sistem Monitoring dan Notifikasi Fall Detection Menggunakan Metode Quad Threshold Berbasis Dual Sensor Akselerasi Dan Orientasi Multiposisi, Surabaya: -, 2018.

[4] M. I. Mirza A, Rancang Bangun Sistem Monitoring Struktur Bangunan Berbasis Jaringan Sensor Nirkabel dengan Analisis Nilai Modal Struktur (Studi Kasus Prototype Jembatan), Surabaya, 2017.

[5] A. Kadir, Buku Pintar Pemrograman Arduino. Yogyakarta: Mediakom, 2014.

[6] Quinlan, J. R. (1987). "Simplifying decision trees". International Journal of Man-Machine Studies. 27 (3): 221–234. CiteSeerX 10.1.1.18.4267. doi:10.1016/S0020-7373(87)80053-6

[7] Aminikhanghahi S, Cook DJ. A Survey of Methods for Time Series Change Point Detection. Knowl Inf Syst. 2017;51(2):339-367. doi:10.1007/s10115-016-0987-z

[8] "nRF24L01" [Daring]. Tersedia: http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01[Diakses 04-November-2019]

[9] Arduino, "Arduino Software (IDE)." [Daring]. Tersedia: https://www.arduino.cc/en/Guide/Environment [Diakses 01-November-2019]

[10] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comput. networks*, vol. 38, no. 4, pp. 393–422, 2002

LAMPIRAN A

KODE SUMBER

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9,10);

const int nodeID=1;
const int chId=1;
const int statusCH=0;
const int rxAddr = 5000;
const int rxAddr1 = 2000;
const int rxAddr2 = 3000;
const int rxAddr3 = 4000;
int sendAddr3 = 5000;


int a = 0, xMili = 0, xSecond = 0, xMinute = 0, xHour = 0, currentSecond = 0, currentMilis=0;
unsigned long wTime;
int flag = 1;
void setup()
{

  xMili = 0; xSecond = 0; xMinute = 0; xHour = 0;


  while (!Serial);
  Serial.begin(9600);


  radio.begin();
  radio.openReadingPipe(0, rxAddr);
  radio.startListening();

}

```

```
37.  void countTime() {
38.    if (xMili == 999) {
39.      ++xSecond;
40.      xMili = 0;
41.    }
42.    if ((xSecond > 59)) {
43.      ++xMinute;
44.      xSecond = 0;
45.      currentSecond = -1;
46.    }
47.    if (xMinute > 59) {
48.      ++xHour;
49.      xMinute = 0;
50.
51.    }
52.    ++xMili;
53.  }
54.
55.  void countMillis() {
56.
57.    wTime=millis();
58.    if (currentMilis==0){
59.      currentMilis=wTime;
60.    }
61.
62.    if ((wTime - currentMilis > 999)) {
63.      ++xSecond;
64.      currentMilis = wTime;
65.    }
66.    if ((xSecond > 59)) {
67.      ++xMinute;
68.      xSecond = 0;
69.      currentSecond = -1;
70.    }
71.    if (xMinute > 59) {
72.      ++xHour;
73.      xMinute = 0;
74.
75.    }
76.   // ++xMili;
77.  }
78.
79.
```

```
80.  void tapListening() {
81.
82.    if (radio.available())
83.    {
84.       char rtext[100];
85.      radio.read(&rtext, sizeof(rtext));
86.
87.      Serial.println(rtext);
88.    }
89.  }
90.
91.  void transmit(String Message)
92.  {
93.
94.    radio.setRetries(15, 15);
95.    radio.stopListening();
96.    radio.openWritingPipe(rxAddr);
97.
98.    String c = Message;
99.    char pesan[25];
100.   c.toCharArray(pesan, 25);
101.
102.   int  rNumber  = random(1,5);
103.   delay(rNumber);
104.
105.   radio.write(&pesan, sizeof(pesan));
106.   radio.openReadingPipe(0, rxAddr);
107.   radio.startListening();
108.
109.
110. }
111.
112.
113.
114.
115. void transmit(int sendAddr, String Message)
116. {
117.
118.   radio.setRetries(15, 15);
119.   radio.stopListening();
120.   radio.openWritingPipe(rxAddr);
121.
122.   String c = Message;
```

```
123.   char pesan[25];
124.   c.toCharArray(pesan, 25);
125.
126.   int  rNumber  = random(1,5);
127.   delay(rNumber);
128.
129.   radio.write(&pesan, sizeof(pesan));
130.   radio.openReadingPipe(0, sendAddr);
131.   radio.startListening();
132.
133.
134. }
135.
136.
137. void loop()
138. {
139.
140.   tapListening();
141.
142.
143.   delay(1);
144. }
```

```
1.   #include <SPI.h>
2.   #include <nRF24L01.h>
3.   #include <RF24.h>
4.   #include <Wire.h>
5.   #include <Adafruit_Sensor.h>
6.   #include <Adafruit_ADXL345_U.h>
7.
8.   Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(
     12345);
9.   RF24 radio(9,10);
10.
11.  const int nodeID=1;
12.  const int chId=1;
13.  const int statusCH=0;
14.  const int rxAddr = 5000;
15.  const int rxAddr1 = 2000;
16.  const int rxAddr2 = 3000;
```

```
17.  const int rxAddr3 = 4000;
18.  int sendAddr3 = 5000;
19.
20.
21.  int a = 0, xMili = 0, xSecond = 0, xMinute = 0, xHour = 0, currentS
     econd = 0, currentMilis=0;
22.  unsigned long wTime;
23.  int flag = 1;
24.
25.
26.
27.  void displaySensorDetails(void)
28.  {
29.   sensor_t sensor;
30.   accel.getSensor(&sensor);
31.   delay(500);
32.  }
33.
34.  void displayDataRate(void)
35.  {
36.   Serial.print ("Data Rate: ");
37.
38.   switch(accel.getDataRate())
39.   {
40.   case ADXL345_DATARATE_3200_HZ:
41.   Serial.print ("3200 ");
42.   break;
43.   case ADXL345_DATARATE_1600_HZ:
44.   Serial.print ("1600 ");
45.   break;
46.   case ADXL345_DATARATE_800_HZ:
47.   Serial.print ("800 ");
48.   break;
49.   case ADXL345_DATARATE_400_HZ:
50.   Serial.print ("400 ");
51.   break;
52.   case ADXL345_DATARATE_200_HZ:
53.   Serial.print ("200 ");
54.   break;
55.   case ADXL345_DATARATE_100_HZ:
56.   Serial.print ("100 ");
57.   break;
58.   case ADXL345_DATARATE_50_HZ:
```

```
59.   Serial.print ("50 ");
60.   break;
61.   case ADXL345_DATARATE_25_HZ:
62.   Serial.print ("25 ");
63.   break;
64.   case ADXL345_DATARATE_12_5_HZ:
65.   Serial.print ("12.5 ");
66.   break;
67.   case ADXL345_DATARATE_6_25HZ:
68.   Serial.print ("6.25 ");
69.   break;
70.   case ADXL345_DATARATE_3_13_HZ:
71.   Serial.print ("3.13 ");
72.   break;
73.   case ADXL345_DATARATE_1_56_HZ:
74.   Serial.print ("1.56 ");
75.   break;
76.   case ADXL345_DATARATE_0_78_HZ:
77.   Serial.print ("0.78 ");
78.   break;
79.   case ADXL345_DATARATE_0_39_HZ:
80.   Serial.print ("0.39 ");
81.   break;
82.   case ADXL345_DATARATE_0_20_HZ:
83.   Serial.print ("0.20 ");
84.   break;
85.   case ADXL345_DATARATE_0_10_HZ:
86.   Serial.print ("0.10 ");
87.   break;
88.   default:
89.   Serial.print ("???? ");
90.   break;
91.   }
92.   Serial.println(" Hz");
93.  }
94.
95.  void displayRange(void)
96.  {
97.  // Serial.print ("Range: +/- ");
98.
99.   switch(accel.getRange())
100.  {
101.  case ADXL345_RANGE_16_G:
```

```
  Serial.print ("16 ");
 break;
 case ADXL345_RANGE_8_G:
 Serial.print ("8 ");
 break;
 case ADXL345_RANGE_4_G:
 Serial.print ("4 ");
 break;
 case ADXL345_RANGE_2_G:
 Serial.print ("2 ");
 break;
 default:
 Serial.print ("?? ");
 break;
 }
 Serial.println(" g");
}


void setup()
{

  xMili = 0; xSecond = 0; xMinute = 0; xHour = 0;


  while (!Serial);
  Serial.begin(9600);
   radio.begin();
  radio.openReadingPipe(0, rxAddr);
  radio.startListening();
 if(!accel.begin())
 {
 Serial.println("Ooops, no ADXL345 detected ... Check your wirin
g!");
 while(1);
 }

 accel.setRange(ADXL345_RANGE_16_G);


}

```

```
144.
145. //-----------------------------------------------CONNECTION-----------
     ----------------
146. void countTime() {
147.   if (xMili == 999) {
148.     ++xSecond;
149.     xMili = 0;
150.   }
151.   if ((xSecond > 59)) {
152.     ++xMinute;
153.     xSecond = 0;
154.     currentSecond = -1;
155.   }
156.   if (xMinute > 59) {
157.     ++xHour;
158.     xMinute = 0;
159.
160.   }
161.   ++xMili;
162. }
163.
164. void countMillis() {
165.
166.   wTime=millis();
167.   if (currentMilis==0){
168.     currentMilis=wTime;
169.   }
170.
171.   if ((wTime - currentMilis > 999)) {
172.     ++xSecond;
173.     currentMilis = wTime;
174.   }
175.   if ((xSecond > 59)) {
176.     ++xMinute;
177.     xSecond = 0;
178.     currentSecond = -1;
179.   }
180.   if (xMinute > 59) {
181.     ++xHour;
182.     xMinute = 0;
183.
184.   }
185. }
```

```
186.
187.
188. void tapListening() {
189.
190.
191.   if (radio.available())
192.   {
193.     char rtext[100];
194.    radio.read(&rtext, sizeof(rtext));
195.
196.    Serial.println(rtext);
197.   }
198. }
199.
200. void transmit(String Message)
201. {
202.
203.   radio.setRetries(15, 15);
204.   radio.stopListening();
205.   radio.openWritingPipe(rxAddr);
206.
207.   String c = Message;
208.   char pesan[30];
209.   c.toCharArray(pesan, 30);
210.
211.   int  rNumber  = random(1,5);
212.   delay(rNumber);
213.
214.   radio.write(&pesan, sizeof(pesan));
215.   radio.openReadingPipe(0, rxAddr);
216.   radio.startListening();
217.
218.
219. }
220.
221.
222.
223.
224. void transmit(int sendAddr, String Message)
225. {
226.
227.   radio.setRetries(15, 15);
228.   radio.stopListening();
```

```
229.  radio.openWritingPipe(rxAddr);
230.
231.  String c = Message;
232.  char pesan[25];
233.  c.toCharArray(pesan, 25);
234.
235.  int  rNumber  = random(1,5);
236.  delay(rNumber);
237.
238.  radio.write(&pesan, sizeof(pesan));
239.  radio.openReadingPipe(0, sendAddr);
240.  radio.startListening();
241.
242.
243. }
244. //-------------------------------------CONNECTION-----------------------
245.
246.
247.
248.
249. void loop()
250. {
251.
252. sensors_event_t event;
253.  accel.getEvent(&event);
254. float current_x= event.acceleration.x;
255. float current_y= event.acceleration.y;
256. float current_z= event.acceleration.z;
257.
258.
259. String myMessage = "slavea,"+String(current_x)+ ","+String(current_y)+ ","+String(current_z);
260. transmit(myMessage);
261.
262.  Serial.println(myMessage);
263. }
```

```
1.  import serial
2.  import time
```

```
import csv
arduino = serial.Serial('COM11', 9600, timeout=2000)

slave_a="slavea"
slave_b="slaveb"
while True:
        data = arduino.readline()[:-2] #the last bit gets rid of the new-line chars
        data=str(data,'utf-8')
        seconds = time.time()
        data=data.split()
        slave_id=data[0]
        if  slave_a in slave_id:
            with open("testing_wsn_slave_A_Skenario_1.csv","a", newline='') as f:
                writer = csv.writer(f,delimiter=",")
                writer.writerow(data)
        if  slave_b in slave_id:
            with open("testing_wsn_slave_B_Skenario_1.csv","a", newline='') as f:
                writer = csv.writer(f,delimiter=",")
                writer.writerow(data)
```

```
import serial
import time
import csv
arduino = serial.Serial('COM11', 9600, timeout=2000)


slave_a="slavea"
slave_b="slaveb"

flag_a=0
flag_b=0

Threshold_atas_x_slave_A=-5.29
Threshold_bawah_x_slave_A=-5.53

Threshold_atas_y_slave_A=-10.30
Threshold_bawah_y_slave_A=-10.71

```

```
19.  Threshold_atas_z_slave_A=7.59
20.  Threshold_bawah_z_slave_A=6.51
21.
22.
23.
24.  Threshold_atas_x_slave_B=0.08
25.  Threshold_bawah_x_slave_B=-1.96
26.
27.  Threshold_atas_y_slave_B=-10.23
28.  Threshold_bawah_y_slave_B=-10.45
29.
30.  Threshold_atas_z_slave_B=2.20
31.  Threshold_bawah_z_slave_B=0.55
32.  while True:
33.      # try:
34.          data = arduino.readline()[:-2] #the last bit gets rid of the new-
     line chars
35.          data=str(data,'utf-8')
36.          seconds = time.time()
37.          # print(data)
38.          data=data.split(",")
39.          # print(data)
40.          slave_id=data[0]
41.          current_x=data[1]
42.          current_y=data[2]
43.          current_z=data[3]
44.          current_x=float(current_x)
45.          current_y=float(current_y)
46.          current_z=float(current_z)
47.          # print(str(current_x+","+current_y+","+current_z))
48.
49.          if  slave_a in slave_id:
50.              if (current_x>Threshold_atas_x_slave_A or current_x<Thr
     eshold_bawah_x_slave_A) and (current_y>Threshold_atas_y_slav
     e_A or current_y<Threshold_bawah_y_slave_A) and (current_z>T
     hreshold_atas_z_slave_A or current_z<Threshold_bawah_z_slave_
     A):
51.                  print("potensi_pada_node_a")
52.                  flag_a=1
53.          if  slave_b in slave_id:
54.              if (current_x>Threshold_atas_x_slave_B or current_x<Thre
     shold_bawah_x_slave_B) and (current_y>Threshold_atas_y_slave
     _B or current_y<Threshold_bawah_y_slave_B) and (current_z>Th
```

```
    reshold_atas_z_slave_B or current_z<Threshold_bawah_z_slave_
    B):
55.             print("potensi_pada_node_b")
56.             flag_b=1
57.     if flag_a==1 and flag_b==1:
58.         print("Longsor")
59.         flag_a=0
60.         flag_b=0
61.
```

LAMPIRAN B

DOKUMENTASI IMPLEMENTASI

BIODATA PENULIS

Muhammad Ryanda Nugraha M lahir di Ujung Pandang Pada tanggal 21 November 1997. Penulis menempuh Pendidikan formal di SDN 187 Kabupaten Pinrang (2003-2009), SMPN 1 Kabupaten Pinrang (2009-2012), SMAN 11 Kabupaten Pinrang (2012-2015), dan Teknik Informatika ITS Surabaya (2016-2020). Bidang studi yang diambil oleh penulis saat berkuliah di Departemen Teknik Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi UKM Robotika ITS (2016-2018) dan Kanda ITS (2016-2020), Penulis juga aktif dalam kegiatan kepanitian seperti SCHEMATICS 2017 - 2018 Divisi Humas, dan Try Out Ewako Sepuluh November 2017 sebagai salah satu pembicara. Penulis pernah menjalani kerja praktik di Center of IoT Innovation (CITI) NTUST, Taiwan Juni-Agustus 2019, Penulis dapat dihubungi melalui nomor *handphone* 095399071938 atau di email mryandanm@gmail.com.