Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR-IF184802

PENINGKATAN AKURASI INDOOR POSITIONING SYSTEM BERBASIS *TRILATERATION* DENGAN *KALMAN FILTER*

AGUNG DWI WICAKSONO
NRP 05111640000123

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

Dosen Pembimbing II
Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D

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

TUGAS AKHIR-IF184802

PENINGKATAN AKURASI INDOOR POSITIONING SYSTEM BERBASIS *TRILATERATION* DENGAN *KALMAN FILTER*

AGUNG DWI WICAKSONO
NRP 05111640000123

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

Dosen Pembimbing II
Tohari Ahmad, S.Kom., MIT., Ph.D.

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

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS-IF184802

IMPROVEMENT OF *TRILATERATION* BASED INDOOR POSITIONING SYSTEM WITH *KALMAN FILTER*

AGUNG DWI WICAKSONO
NRP 05111640000123

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

Second Advisor
Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D

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

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENINGKATAN AKURASI INDOOR POSITIONING SYSTEM BERBASIS *TRILATERATION* DENGAN *KALMAN FILTER*

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:
AGUNG DWI WICAKSONO
NRP: 05111540000123

Disetujui oleh Pembimbing tugas akhir:

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

2. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D .....
   (NIP. 198106202005011003) (Pembimbing 2)

SURABAYA
JUNI, 2020

(Halaman ini sengaja dikosongkan)

PENINGKATAN AKURASI INDOOR POSITIONING SYSTEM BERBASIS *TRILATERATION* DENGAN *KALMAN FILTER*

Nama Mahasiswa	**:**	**Agung Dwi Wicaksono**
NRP	**:**	**05111640000123**
Departemen	**:**	**Informatika FTIK ITS**
Dosen Pembimbing 1	**:**	**Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**:**	**Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D**

Abstrak

Indoor Positioning System (IPS) adalah sistem untuk menentukan posisi seseorang atau suatu benda pada ruangan. Berbda dengan *Global Positioning System (GPS),* yang menggunakan satelit untuk menentukan posisi, IPS menggunakan benda yang ada dalam ruangan seperti bluetooth atau *WiFi*.

Dari kekuatan sinyal *WiFi* dapat ditentukan jarak antara penerima dan pengirim sinyal dengan *signal propagation model*. Dibutuhkan tiga jarak dari tiga poin yang diketahui koordinatnya untuk mendapatkan posisi obyek dengan *trilateration.*

Namun, penentuan jarak dari sinyal ini cenderung tidak stabil karena flunktuasi sinyal sehingga menghasilkan koordinat yang tidak akurat. Dalam buku ini digunakan metode *Kalman filtering* untuk mengurangi ketidak stabilan flunktuasi sinyal untuk mendapatkan jarak yang lebih akurat yang pada hasilnya akan memberikan koordinat yang lebih akurat.

Kalman filter sendiri menggunakan data yang telah diobservasi atau dihitung sebelumnya untuk mengestimasi kondisi saat ini. Hal ini membuat estimasi yang dibuat cenderung lebih akurat daripada estimasi yang dibuat dalam sekali penghitungan.

Kata kunci: ***Indoor Positioning System, Trilateration, Kalman filter***

IMPROVEMENT OF *TRILATERATION* BASED INDOOR POSITIONING SYSTEM WITH *KALMAN FILTER*

Student's Name : Agung Dwi Wicaksono
Student's ID : 05111640000123
Department : Informatika FTIK-ITS
First Advisor : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
Second Advisor : Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D

Abstract

Indoor Positioning System (IPS) is a system for determining the position of a person or an object in a room. Unlike the Global Positioning System (GPS), which uses satellites to determine position, IPS uses objects in the room such as Bluetooth or WiFi.

From the strength of the WiFi signal the distance between the receiver and sender of the signal can be determined by signal propagation model. To determine the position of a person or object requires three distances from three points whose coordinates are known by the trilateration algorithm.

Determination of the distance from this signal tends to be unstable due to fluctuation of the signal so that it produces inaccurate coordinates. This book uses the Kalman filtering method to that unstable aspect of signal to produce more accurate coordinates.

Kalman filter itself uses data that has been observed or previously calculated to estimate the current conditions. This makes estimates made tend to be more accurate than estimates made in one calculation.

Keyword: Wireless Sensor Network, Cluster Head, Cluster node, nRF24.

(Halaman ini sengaja dikosongkan)

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 *CLUSTER-BASED* PROTOKOL UNTUK PENGIRIMAN DATA SECARA ADAPTIF DENGAN PENGATURAN KEKUATAN TRANSMISI DAN *MONITORING* KETERSEDIAN ENERGI PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN nRF24L01".

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 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 Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D selaku Dosen Pembimbing penulis yang telah membimbing, memberikan nasihat, dan memotivasi penulis sehingga penulis dapat menyelesaikan tugas akhir ini.
4. Ibu Dr.Eng. Chastine Fatichah, S.Kom, M.Kom. selaku kepala Departemen Informatika ITS.

5. Bapak dan Ibu Dosen yang telah memberikan ilmunya selama penulis berkuliah di Informatika ITS.
6. Sahabat baik penulis yang tidak perlu disebutkan Namanya yang selalu support dan mendukung penulis selama bertahun-tahun.
7. Teman-teman angkatan 2016 yang sudah menjadi saksi hidup perjalanan karir penulis selama berkuliah di Informatika ITS.
8. Untuk orang-orang yang tidak dapat disebutkan satu persatu oleh penulis dan pembaca buku tugas akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun 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 menjalani kehidupan, jangan menyerah, karena Allah masih ingin melihat kita berjuang. Semoga kita semua selalu diberi kebahagiaan lahir dan batin dan kesuksesan dunia akhirat. Aamiin.

Surabaya, 25 Juni 2020

Agung Dwi Wicaksono

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indoor Positioning System adalah sistem yang digunakan untuk memprediksi posisi seseorang atau obyek pada ruangan tertutup. Ada banyak cara dalam penentuan posisi ini, salah satunya adalah dengan *trilateration* dimana posisi suatu titik ditentukan oleh jarak titik tersebut kepada tiga titik lain yang diketahui koordinatnya.

Untuk mendapatkan jarak suatu perangkat dengan akses poin *WiFi* dibutuhkan kekuatan sinyal dan frekuensi dari *WiFi* itu sendiri yang terkadang tidak stabil yang akhirnya mempengaruhi akurasi dari posisi yang didapat. Terlebih lagi dengan *trilateration* yang menggunakan tiga jarak untuk mengestimasi posisi dimana kesalahan pada satu jarak dapat menghasilkan kesalahan estimasi yang signifikan.

Pada Tugas Akhir ini digunakan *kalman filter* untuk mengurangi ketidak stabilan sinyal tersebut untuk menghasilkan estimasi posisi yang lebih akurat.

1.2 Rumusan Masalah

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

1. Bagaimana cara menentukan jarak dari akses poin *WiFi* ke anroid?
2. Bagaimana cara mendapatkan kekuatan sinyal yang stabil?
3. Bagaimana cara mendapatkan posisi android berdasarkan jarak dari titik akses poin yang diketahui?
4. Apakah peletakan akses poin memiliki hubungan kepada akurasi estimasi posisi?

1.3 Batasan Permasalahan

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

1. Menggunakan android Android 4.4.4 (KitKat) sebagai perangkat untuk menerima kekuatan sinyal *WiFi*.
2. Menggunakan router sebagai pemancar sinyal *WiFi*.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Meningkatkan akurasi Indoor Positioning System dengan *Kalman filter*.

1.5 Manfaat

Manfaat yang diperoleh dari pengerjaan tugas akhir ini adalah menghasilkan peningkatan akurasi *indoor positioning system* berbasis *trilateration* dengan *kalman filtering*

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 *kalman filter* untuk *filtering* koordinat 2 dimensi dan *trilateration* berbasis *WiFi* RSSI untuk mencari koordinat.

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 dilakukan dengan menggunakan android studio dan bahasa java 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 percobaan lapangan yang berlokasi di 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 *Indoor Positioning System*, *Trilateration*, *Signal Propagation Model*, *Kalman filter*, *Android*, *Java*, *Android Studio*, *Python*, *Spyder*.
3. Bab III. Perancangan Perangkat Lunak dan Perangkat Keras

Bab ini berisi pembahasan mengenai desain dari jaringan sensor nirkable yang akan dibuat, meliputi arsitektur dan 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 kemampuan jaringan dan penggunaan energy pada jaringan sensor nirkabel.
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.

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 *Indoor Positioning System*

Indoor positioning system (IPS) digunakan untuk mengestimasi posisi orang dan obyek pada suatu ruangan. GPS atau *Global Positioning System* menggunakan satelit untuk mengestimasi lokasi seseorang atau obyek di bumi. Namun saat obyek atau seseorang tersebut berada dalam ruangan GPS tidak dapat mengestimasi keberadaan orang tersebut pada ruangan itu, namun hanya keberadaan orang atau obyek pada peta global.

Maka dari itu digunakan Indoor Positioning System yang dapat mengestimasi keberadaan obyek pada suatu ruangan. IPS yang digunakan pada ruangan tentunya menggunakan hal yang ada dalam ruangan tersebut untuk mengestimasi posisi dari obyek, bisa jadi melalui kekuatan sinyal *WiFi*, *Bluetooth* atau *NRF* [1].

WiFi sering digunakan pada ruangan pada tempat-tempat umum. Pada TA ini digunakan IPS berdasarkan sinyal *WiFi*.

2.2 *Signal Propagation Model* (estimasi jarak)

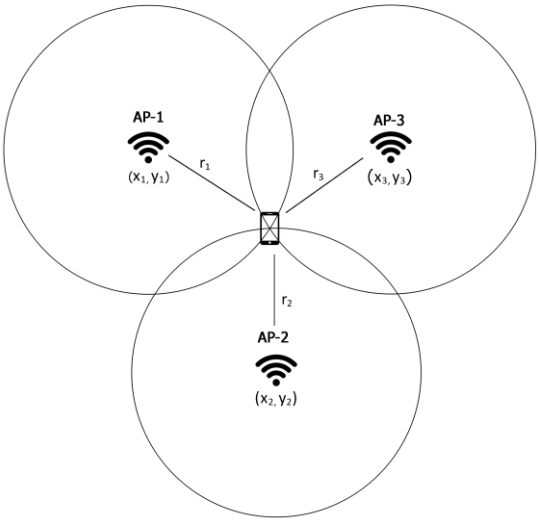
Gelombang radio sebagian besar dipengaruhi oleh dua fenomena yaitu kekuatan sinyal yang melalui atmosfir dan perpanjangan dari fenomena ini ke setiap obyek yang dilalui oleh sinyal [2].

Fenomena pertama adalah berkurangnya kekuatan sinyal berdasarkan jarak dari transmitter sedangkan fenomena kedua adalah berkurangnya atau bertambahnya sinyal karena refleksi, rerfaksi dan atenuasi. Dalam tugas akhir ini digunakan *signal*

propagation model yang telah dimodelkan untuk situasi dalam ruangan [2].

2.3 *Trilateration (estimasi posisi)*

Trilateration adalah metode yang digunakan untuk mendapatkan posisi obyek berdasarkan jarak dari beberapa titik yang diketahui posisinya. [1]



Gambar 2.1 Ilustrasi *trilateration*

Dari gambar 2.1 setiap lingkaran mempresentasikan semua kemungkinan perangkat android berada berdasarkan jarak antara masing-masing akses poin dan perangkat android. Tujuan *trilateration* adalah untuk mengestimasikan posisi perangkat android berdasarkan jaraknya dari tiga akses poin yang telah diketahui koordinatnya. [3]

2.4 *Kalman filter*

Kalman filtering atau juga dikenal dengan nama *Linear Quadratic Estimation* (LQE) adalah algoritma yang menggunakan

hasil pengamatan atau pengukuran dalam jangka waktu tertentu yang di dalamnya mengandung *noise* dan menghasilkan estimasi dari pengamatan yang cenderung lebih akurat dari pengukuran satu waktu.

Kalman filter telah lama digunakan untuk menggabungkan estimasi saat ini dengan pengukuran baru untuk mendapatkan hasil estimasi baru yang lebih akurat dengan cara meminimalkan varians kesalahan estimasi. Oleh karena itu *kalman filter* adalah metode kandidat yang bagus untuk penentuan posisi [4].

Pada tugas akhir ini *kalman filter* digunakan untuk menstabilkan kekuatan sinyal yang diterima oleh perangkat android.

2.5 Android

Android adalah sistem operasi terbuka (*open-source*) berbasis Linux yang dirancang untuk perangkat bergerak dengan layar sentuh, seperti telepon pintar dan komputer Tabelt. Antarmuka pengguna Android umumnya adalah layar sentuh di mana pengguna menggunakan gerakan sentuh untuk menggunakannya, serta papan ketik virtual untuk menulis teks [5].

Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google. Google kemudian membelinya pada tahun 2005. Google merilis kode sistem operasi ini di bawah Lisensi Apache. Kode dengan sumber terbuka ini memungkinkan perangkat lunak untuk dimodifikasi secara bebas dan didistribusikan oleh para pembuat perangkat, operator nirkabel, dan pengembang aplikasi. Selain itu, Android memiliki sejumlah besar komunitas pengembang aplikasi yang memperluas fungsionalitas perangkat, yang umumnya ditulis dalam bahasa pemrograman Java [5].

Dalam tugas akhir ini, Android digunakan sebagai basis sistem operasi *smartphone*. Aplikasi untuk pengambilan data RSSI akan dibuat akan dijalankan pada sistem operasi Android.

2.6 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems (saat ini merupakan bagian dari Oracle) dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaks yang terdapat pada C dan C++ namun dengan sintaks model objek yang lebih sederhana serta dukungan rutin-rutin yang minimal. Aplikasi-aplikasi berbasis Java umumnya dikompilasi ke dalam p-code (bytecode) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM) [6].

Java merupakan bahasa pemrograman yang bersifat umum (general purpose), dan secara khusus didesain untuk menggunakan ketergantungan implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi Java mampu berjalan di beberapa platform sistem operasi yang berbeda, Java dikenal dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini Java merupakan bahasa pemrograman yang paling populer digunakan dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi [6].

Dalam Tugas Akhir ini, aplikasi untuk pengambilan data RSSI akan dibuat dengan menggunakan bahasa pemrograman Java.

2.7 Android Studio

Android Studio adalah *Integrated Development Enviroment* (IDE) untuk sistem operasi Android, yang dibangun diatas perangkat lunak JetBrains IntelliJ IDEA dan didesain khusus untuk pengembangan Android. IDE ini merupakan pengganti dari Eclipse Android Development Tools (ADT) yang sebelumnya merupakan IDE utama untuk pengembangan aplikasi Android[7]..

Android Studio sendiri pertama kali diumumkan di konferensi Google I/O pada tanggal 16 Mei 2013. Ini merupakan tahap *preview* dari versi 0.1 pada Mei 2013, dan memasuki tahap uji coba (beta) sejak versi 0.8 dan mulai diliris pada Juni 2014. [7]

Versi rilis stabil yang pertama diliris pada December 2014, dimulai sejak versi 1.0. Sedangkan versi stabil yang sekarang adalah versi 3.6.3 yang diliris pada Februari 2020. [7]

Dalam Tugas Akhir ini, Android Studio digunakan untuk membuat aplikasi visualisasi yang dapat dijalankan pada *smartphone* dengan sistem operasi Android.

2.8 Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python juga didukung oleh komunitas yang besar. [8]

Python mendukung multi paradigma pemrograman, utamanya; namun tidak dibatasi; pada pemrograman bkesalahanientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operas. [8]

Pada tugas akhir ini python akan digunakan untuk analisis data dan pengukuran akurasi *indoor positioning system*.

2.9 Spyder (IDE)

Spyder adalah adalah *Integrated Development Enviroment* (IDE) *open source cross-platform* untuk bahasa pemrograman Python. Spyder terintegrasi dengan berbagai paket terkemuka

untuk Python seperti NumPy, SciPy, Matplotlib, pandas, IPython, SymPy, dan Cython dan berbagai perangkat lunak *open source* lainnya. [9]

Awalnya dibuat dan dikembangkan oleh Pierre Raybaut pada 2009, sejak 2012 Spyder telah dipertahankan dan terus ditingkatkan oleh tim pengembang Python ilmiah dan komunitas. [9]

Pada tugas akhir ini spyder akan digunakan untuk analisis data dan pengukuran akurasi *indoor positioning system.*

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 *indoor positioning system* berbasis *trilateration* untuk mengestimasi posisi *receiver* dengan *signal propagation model* untuk menentukan jarak antar *receiver* dan *transfimer* dan *kalman filter* untuk meningkatkan akurasi estimasi.

3.1 Deskripsi Umum

Pada tugas akhir ini akan dibuat sebuah implementasi *indoor positioning system* menggunakan kekuatan sinyal *WiFi* yang didapat dari 3 akses poin.

Dari setiap akses poin tersebut akan dicatat kekuatan sinyal (RSSI dalam dBm) yang diterima perangkat android. Kekuatan sinyal ini relatif tidak stabil dan memiliki *noise*. *Noise* ini membuat estimasi jarak yang dilakukan akan memiliki akurasi yang kurang bagus.

Untuk menghilangkan *noise* tersebut digunakan *kalman filter* untuk mencari nilai RSSI yang stabil dari *n* jumlah sampel yang diambil.

Setelah didapatkan kekuatan sinyal yang stabil, diestimasi jaraknya kepada masing masing akses poin *(transmitter)* dengan *signal propagation model*. Dari jarak-jarak yang telah diestimasi akan dicari koordinat dari receiver dengan mengestimasi koordinat posisi perangkat android berdasarkan koordinat posisi akses poin dengan algoritma *trilateration*.

3.2 Daftar Istilah

Daftar istilah yang sering digunakan pada buku tugas akhir ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	*Transmitter*	Perangkat yang digunakan untuk menyiarkan sinyal. Dalam konteks tugas akhir ini digunakan akses poin *WiFi*
2	*Receiver*	Perangkat yang digunakan untuk menerima sinyal. Dalam konteks tugas akhir ini digunakan perangkat android.
3	*Indoor positioning system*	Sistem yang digunakan untuk mengestimasi posisi seseorang atau suatu obyek dalam ruangan tertutup.
4	RSSI	*Received Signal Strength Indication*, Kekuatan sinyal yang diterima oleh perangkat android dalam dBm. RSSI ini memiliki rentang antara -30 sampai -90 dBm

3.3 Arsitektur Sistem

Pada sub bab ini akan di jelaskan mengenai arsitektur umum *indoor positioning system* yang akan dibangun. Sistem ini terdiri dari 3 tahapan yang meliputi *kalman filtering*, estimasi jarak dengan *signal propagation model*, dan estimasi posisi dengan *trilateration*.

Gambar 3.1 *Flowchart* sistem.

3.4 *Kalman filter*

Kekuatan sinyal akses poin (RSSI) memiliki banyak noise, tidak seperti idealnya dimana RSSI ini hanya dipengaruhi oleh jarak antara *receiver* dan akses poin. *Noise* ini terjadi karena banyak hal seperti *multi-path reflection* dimana sinyal radio yang diterima oleh *receiver* bisa saja datang dari hasil refleksi sinyal itu terhadap benda pada ruangan seperti tembok kursi dan lain sebagainya [10].

Untuk menghilangi *noise* ini digunakan *kalman filter*. *Kalman filter* adalah estimator yang menggunakan perhitungan - perhitungan yang telah dilakukan sebelumnya (histori perhitungan) untuk mengestimasi perhitungan saat ini untuk mengurangi *noise* dari perhitungan saat ini.

Kalman filter menggunakan model linear yang dimana perhitungan sekarang didapat dari perhitungan sebelumnya. Persamaan umum model transisi *kalman filter* ditulis sebagai berikut.

$$X_k = AX_{k-1} + Bu_{k-1} + w_{k-1} \quad (1)$$

Pada persamaan (1) X_k didefinisikan sebagai kombinasi pengukuran sebelumnya, yang merupakan pengukuran saat ini. State dari pengukuran sebelumnya ditulis sebagai X_{k-1} dengan matriks transformasi *A* untuk mengubah domain matriks dari state pengukuran sebelumnya menjadi domain pengukuran sekarang, kontrol input *u* sebagai variabel kontrol tambahan pada pengukuran (contohnya kecepatan untuk input tambahan pada pengukuran) dan w_{k-1} *process noise* yang merupakan *noise* dari sistem *kalman filter* itu sendiri.

Untuk *filtering* RSSI pada tugas akhir ini diasumsikan perangkat android tidak bergerak pada saat pengukuran RSSI dan waktu pengukuran statis. Dengan kata lain dalam *filtering* ini hanya dicari data RSSI yang stabil. Untuk itu nilai *u* dan nilai A tidak dianggap [10].

$$X_k = X_{k-1} + w_{k-1} \quad (2)$$

Step berikutnya adalah bagaimana mendefinisikan state *X* mempengaruhi pengukuran *z (prediction step)* [10].

$$z_k = HX_k + v_k \quad (3)$$

Untuk memprediksi state berikutnya dilakukan prediksi. Model prediksi untuk *kalman filter* ini cukup simpel dimana: [10]

$$\bar{u}_k = u_{k-1} \quad (4)$$
$$\bar{\Sigma}t = \Sigma t - 1 + Rt \quad (5)$$

Pada persamaan (4), $\bar{u}$ didefinisikan sebagai prediksi pengukuran yang dibuat oleh sistem, berbeda dengan *X* yang merupakan pengukuran dari sistem. Bar diatas $\bar{u}$ dan $\bar{\Sigma}t$ menunjukkan bahwa variabel tersebut merupakan variabel prediksi yang belum ditambah dengan nilai pengukuran dari RSSI sekarang.

Pada persamaan (5) Σ didefinisikan sebagai kepastian dari prediksi sistem. Σ diukur dari kepastian prediksi pada state

sebelumnya ditambah dengan *R* yang merupakan *noise* yang ditimbulkan dari sistem *kalman filter* sendiri. Hal ini berarti apabila prediksi pada state sebelumnya tidak pasti maka prediksi pada state saat ini juga tidak pasti. Pada *filtering* RSSI digunakan nilai *R* yang kecil, hal ini dilakukan dengan asumsi sebagian besar *noise* didapat dari kesalahan pengukuran RSSI.

Dari estimasi prediksi dihitung kalman gain untuk mengkorporasi prediksi yang telah dibuat dengan pengukuran riil *(update step)*, kalman gain ditulis sebagai berikut [10].

.

$$K_t = \bar{\Sigma} t(\bar{\Sigma} t Q_t)^{-1} \quad (6)$$

Kalman Gain merupakan fungsi bobot untuk menentukan seberapa pasti pengukuran yang dilakukan. Q didefinisikan sebagai *noise* dari pengukuran nyata, atau seberapa besar *noise* yang diharapkan pada pengukuran RSSI sesungguhnya.

Apabila kepastian dari prediksi ($\bar{\Sigma} t$ tinggi) maka pengukuran harus diutamakan. Dari bobot ini dibuat prediksi pengukuran dan kepastian sistem dengan persamaan sebagai berikut: [10]

$$\mu t = \bar{\mu} t + Kt(zt - \bar{\mu} t) \quad (7)$$

$$\Sigma t = \bar{\Sigma} t - (Kt\bar{\Sigma} t) \quad (8)$$

Pada step ini dibuat prediksi akhir dengan mengkorporasi pengukuran riil (*z)* kepada prediksi yang telah dibuat pada step sebelumnya. Pada step ini semakin besar *kalman gain* yang didapatkan berarti semakin besar pengaruh dari pengukuran kepada prediksi yang dibuat.

Pada tugas akhir ini akan digunakan *kalman filtering* untuk mendapatkan sinyal yang stabil dengan cara mengambil *n* jumlah sampel data RSSI dan dimasukkan ke dalam *kalman filter*.

3.5 Estimasi jarak

Dari kekuatan sinyal yang sudah distabilkan dengan *kalman filter* dicari jarak antara perangkat android dan akses poin yang memancarkan sinyalnya menggunakan *signal propagation model* yang diajukan oleh Chipcon [1] sebagai berikut:

$$RSSI = -(10n\ log_{10}\ d + A) \quad (9)$$

$$n = -\left(\frac{RSSI-A}{10\ log_{10}\ d}\right) \quad (10)$$

Dimana :

- RSSI adalah kekuatan sinyal yang diterima oleh perangkat android
- n adalah *signal propagation constant*
- d adalah jarak antara akses poin dan perangkat android
- A adalah RSSI yang diambil pada jarak 1 meter antar perangkat android dan akses poin

Signal propagation model sendiri memiliki konstanta (*n*) pada ekuasinya (9) yang membutuhkan penghitungan experiment lapangan untuk menentukannya. Konstanta tersebut adalah konstanta propagasi sinyal yang didapat dari ekuasi (10) dan konstanta kekuatan sinyal yang diambil pada jarak satu meter dari *transmitter,* konstanta *A.*

3.6 Estimasi posisi

Pada tahapan estimasi posisi akan dilakukan *trilateration* untuk mendapatkan estimasi posisi dari *receiver*. Dalam *trilateration* koordinat x dan y dari receiver akan didapatkan dari persamaan jarak berikut [3].

$$r_1^2 = (x - x_1)^2 + (y - y_1)^2 \quad (11)$$
$$r_2^2 = (x - x_2)^2 + (y - y_2)^2 \quad (12)$$
$$r_3^2 = (x - x_3)^2 + (y - y_3)^2 \quad (13)$$

Dari persamaan jarak tersebut masing masing dijabarkan.

$$r_1^2 = x^2 - 2x_1x + x_1^2 + y^2 - 2\,y_1y + y_1^2 \quad (14)$$
$$r_2^2 = x^2 - 2x_2x + x_2^2 + y^2 - 2\,y_2y + y_2^2 \quad (15)$$
$$r_3^2 = x^2 - 2x_3x + x_3^2 + y^2 - 2\,y_3y + y_3^2 \quad (16)$$

Persamaan (5) dikurangi persamaan (4) dan pengurangan persamaan (6) dikurangi persamaan (5).

$$r_1^2 - r_2^2 - x_1^2 + x_2^2 - y_1^2 + y_2^2 = (-2x_1 + 2x_2)x + (-2y_1 + 2y_2)x \quad (17)$$

$$r_3^2 - r_4^2 - x_3^2 + x_4^2 - y_3^2 + y_4^2 = (-2x_3 + 2x_4)x + (-2y_3 + 2y_4)x \quad (18)$$

Sutitusi dengan variabel A, B, C untuk persamaan (7) dan D, E, F untuk persamaan (8).

$$Ax + By = C \quad (19)$$
$$Dx + Ey = F \quad (20)$$

Didapatkan solusi dari *trilateration*.

$$x = \frac{CE - FB}{EA - BD} \quad (21)$$
$$y = \frac{CD - AF}{BD - AE} \quad (22)$$

(Halaman ini sengaja dikosongkan)

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 4.1 dibawah ini.

Tabel 4.1 Tabel Lingkungan Implementasi Perangkat Keras

Perangkat	**Detail**
Perangkat Android	**Model:** • Lenovo A6000 **Launch :** • 2015, January 09. Rilis 2015, January 28 **OS:** • Android 4.4.4 (KitKat) **WLAN:** • Wi-Fi 802.11 b/g/n, hotspot

Akses Poin 1	**Model:** • TP Link TL-WR840N **Frequency :** • 2.4-2.4835GHz **Signal Rate:** • 11n: Up to 300Mbps(dynamic) • 11g: Up to 54Mbps(dynamic) • 11b: Up to 11Mbps(dynamic)
Akses Poin 2	**Model:** • Toto Link N210RE **Frequency :** • 2.4-2.4835GHz **Signal Rate:** • 11n: Up to 300Mbps(dynamic) • 11g: Up to 54Mbps(dynamic) • 11b: Up to 11Mbps(dynamic)
Akses Poin3	**Model:** • D Link DIR-612 N300 **Frequency :** • 2.4-2.4835GHz **Signal Rate:** • 11n: Up to 300Mbps(dynamic) • 11g: Up to 54Mbps(dynamic) 11b: Up to 11Mbps(dynamic)

4.1.2 Lingkungan Implementasi Perangkat Lunak

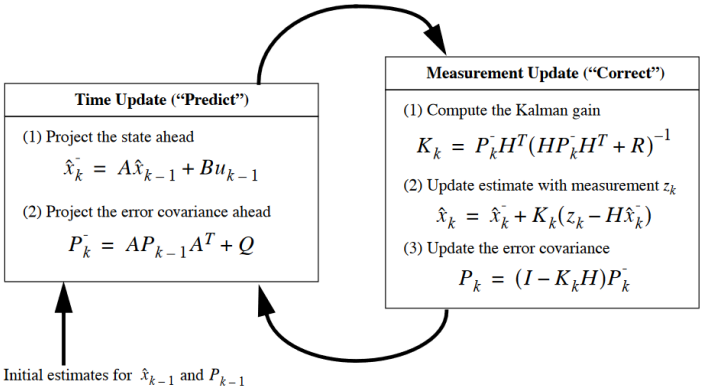
Pada bagian ini dibahas mengenai perangkat lunak apa saja yang dibutuhkan untuk membangun sitem. Lingkungan implementasi perangkat lunak dari sistem yang akan dibangun secara lebih lengkap di jelaskan pada Tabel 4.2 di bawah ini.

Tabel 4.2 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 adalah adalah *Integrated Development Enviroment* (IDE) *open source cross-platform* untuk bahasa pemrograman Python. Spyder terintegrasi dengan berbagai paket terkemuka untuk Python seperti NumPy, SciPy, Matplotlib, pandas, IPython, SymPy, dan Cython dan berbagai perangkat lunak *open source* lainnya.

4.2 Implementasi *kalman filter*

Pada bagian ini dijelaskan implementasi dari *kalman filter* yang dibagi menjadi 3 bagian yaitu inisilaisasi, fungsi *predict* dan fungsi *update*.



Gambar 4.1 *Flow kalman filter* [10].

4.2.1 Inisialisasi

Pertama tama diinisialisasi nilai estimasi *noise* untuk sistem (*R*) dan pengukuran (*Q*), juga diinisialisasi nilai keyakinan (*cov*) dan posisi saat ini (*x*).

```
1.  class KalmanFilter
2.      KalmanFilter(R, Q)
3.          this.R = R;
4.          this.Q = Q;
5.
6.          this.cov = -1;
7.          this.x = -1;
```

Kode Sumber 4.1 Inisialisasi *kalman filter*

4.2.2 Fungsi *predict*

Fungsi *predict* membuat prediksi pengukuran untuk state saat ini. Seperti yang telah dijelaskan pada bab IV fungsi *predict* ini secara langsung menggunakan prediksi dari state sebelumnya karena variabel kontrol pada *kalman filter* untuk RSSI ini tidak dianggap.

```
1.  predict()
2.      return this.x ;
3.
4.  uncertainty()
5.      return this.cov + this.R;
6.
```

Kode Sumber 4.2 Fungsi *predict kalman filter*

4.2.3 Fungsi *update*

Pada fungsi update dihitung *kalman gain* dan dilakukan pembaruan pada prediksi pengukuran dengan mengkorporasi pengukuran riil RSSI (z).

```
1.  filter(z )
2.      if (this.x == -1)
3.          this.x =  z;
4.          this.cov =  this.Q ;
5.
6.      else
7.          double predX = this.predict();
8.          double predCov = this.uncertainty();
9.
10.         double K = predCov * (1 / (predCov  + this.Q));
11.
12.         this.x = predX + K * (z - ( predX));
13.         this.cov = predCov - (K *  predCov);
14.     return this.x;
```

Kode Sumber 4.3 Fungsi update *kalman filter*

4.3 Implementasi estimasi jarak

Pada bagain ini dijelaskan implementasi dari pengambilan estimasi jarak antar akses poin dan perangkat android dengan *signal propagation model*.

4.3.1 Kalibrasi algoritma

Subbab ini membahas pengkalibrasian algoritma *signal propagation model* yang memiliki dua variabel konstan yaitu variabel *A* yang merupakan kekuatan RSSI pada jarak 1 meter antara akses poin dan perangkat android, dan variabel *n* yang merupakan *signal propagation constant.*

```
1.  getN(rssi, a, d)
2.      return -1* ((rssi +a)/ (10 * log10(d)))
```

Kode Sumber 4.4 Fungsi untuk mencari signal propagation constant

Pada kode sumber 4.1 ditentukan nilai *signal propagation constant* dengan menggunakan nilai rssi pada jarak d dan nilai a (nilai rssi pada jarak 1 meter). Penentuan nilai *signal propagation constant* dan nilai *A (*kekuatan RSSI pada jarak 1 meter) dilakukan untuk tiap-tiap akses poin.

4.3.2 *Signal propagation model*

Setelah didapatkan nilai *n* dan *A* diimplementasikan *signal propagation model* untuk mengestimasi jarak.

```
1.  getDistance(rssi, a, n):
2.      return pow(10, -1*((rssi-a)/(10*n)))
```

Kode Sumber 4.5 Fungsi untuk mencari signal propagation constant

4.4 Implementasi estimasi posisi

Pada bagain ini dijelaskan implementasi dari pengambilan estimasi posisi perangkat android dengan *trilateration. Trilateration* dilakukan dengan menggunakan 3 akses poin sebagai titik pnentu. Hal ini dilakukan karena *trilateration* membutuhkan minimal 3 buah titik yang diketahui untuk menentukan posisi.

```
1.  def getCoord(r1, x1,y1,  r2, x2, y2, r3, x3, y3) :
2.      A = 2*x2 - 2*x1
3.      B = 2*y2 - 2*y1
4.      C = r1*r1 - r2*r2 - x1*x1 + x2*x2 - y1*y1 + y2*y2
5.      D = 2*x3 - 2*x2
6.      E = 2*y3 - 2*y2
7.      F = r2*r2 - r3*r3 - x2*x2 + x3*x3 - y2*y2 + y3*y3
8.      x = (C*E - F*B) / (E*A - B*D)
9.      y = (C*D - A*F) / (B*D - A*E)
10.     return x,y
```

Kode Sumber 4.6 Fungsi *trilateration*

Pada kode sumber 4.3 diambil 9 parameter r, x, y yang masing-masing merupakan jarak antar akses poin dan perangkat android, koordinat x akses poin, koordinat y akses poin.

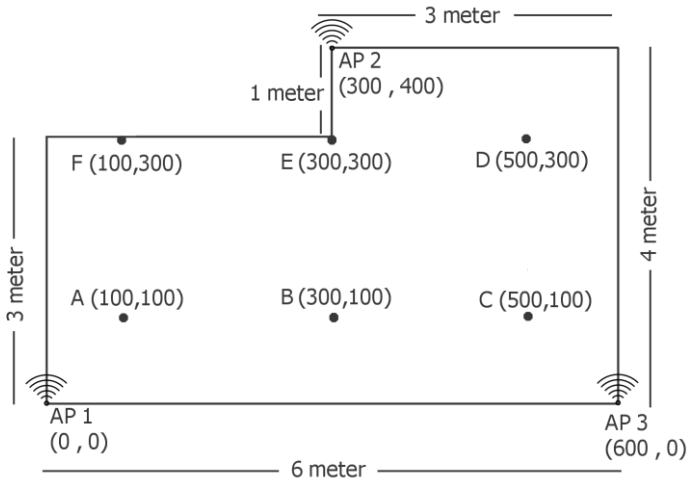
(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai penentuan variabel constant dan skenario uji coba yang dilakukan serta evaluasi perbandingan akurasi estimasi posisi dengan menggunakan *Kalman filter* dan tanpa *Kalman filter*. Hasil uji coba didapatkan dari implementasi yang dijelaskan pada Bab 4 dengan skenario 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 *indoor positioning system* dilakukan pada ruangan dengan panjang 6 meter dan lebar 4 meter. 3 buah akses poin diletakkan pada pojok pojok ruangan.



Gambar 5.1 Denah Uji Coba

Koordinat ditentukan dengan perhitungan sentimeter yang dimulai dari pojok kiri bawah ruangan. Akses poin (AP) 1 diletakkan pada posisi kiri bawah ruangan, pada koordinat (0, 0), akses poin 2 diletakkan di tengah atas ruangan pada koordinat (300, 400), dan akses poin 3 diletakkan pada kanan bawah ruangan pada koordinat (0, 600). Penentuan koordinat dilakukan dengan menyamakan ukuran ruangan dengan penghitungan 1 cm untuk satu poin koordinat.

Pengujian pertama dan kedua dilakukan dengan pengambilan sampel data pada jarak tertentu dari akses poin. Pengujian pertama ini dilakukan untuk mengukur variabel konstan (kalibrasi) untuk penentuan jarak (*signal propagarion model*) yang telah dibahas pada bab IV.

Pengujian ketiga dilakukan pada titik-titik ruangan dengan denah pada gambar 5.1. Pengujian kedua dilakukan untuk pengukuran akurasi estimasi posisi tanpa menggunakan *kalman filter*, sedangkan pengujian ketiga dilakukan dengan menggunakan *kalman filter*.

5.2 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam proses uji coba. Dengan skenario yang dibuat akan diuji apakah perangkat yang di buat sudah berjalan sesuai dengan yang di rancang dan benar, dan menentukan performa pada masing-masing skenario. Akan ditentukan akurasi performa penentuan jarak dengan *signal propagation model* dan penentuan posisi berdasarkan *trilateration* dengan kekuatan sinyal RSSI yang telah dilakukan *filtering* menggunakan *kalman filter*.

1. Pengujian akurasi jarak dengan *signal propagation model* dengan mengambil beberapa data pada jarak tertentu dari akses poin. Pada uji coba ini diambil sampel data yang besar untuk mendapatkan kekuatan sinyal absolut pada titik yang diuji.

2. Pengujian *kalman filtering* untuk estimasi jarak dilakukan dengan mengambil sampel data kecil pada titik uji dan dilakukan *kalman filtering* pada data tersebut. Dari RSSI yang sudah difilter ditentukan estimasi jarak dan dibandingkan dengan penentuan jarak tanpa menggunakan *kalman filtering*.
3. Pengujian akurasi estimasi posisi dengan *trilateration* berdasarkan jarak yang ditentukan dengan *signal propagation model* dan titik koordinat akses poin yang diketahui.
4. Skenario pengujian 4 dilakukan dengan melakukan estimasi posisi dengan *trilateration* dan *kalman filter* sama persis seperti skenario pengujuan 3 namun pada skenario pengujian 4 uji coba dilakukan dengan menaruh akses poin pada tengah ruangan.
5.

Pengujian pertama dilakukan dengan mengambil *n* data pada setiap jarak 1, 2, 3, dan 4 pada masing masing akses poin dan ditentukan *n* yang paling optimal.

Pengujian ketiga dan keempat dilakukan dengan mengambil 50 data pada masing masing titik pada ruangan, yang tertera gambar 5.1.

5.2.1 Skenario Uji Coba 1

Dalam skenario uji coba yang pertama akan dihitung variabel untuk signal propagation constant. Pada uji coba ini diambil 1000 data RSSI pada jarak 1 meter, 2 meter, 3meter dan 4 meter pada masing masing akses poin untuk *training set* guna menentukan nilai *n* dan *A* . Dan 1000 dataset pada jarak yang sama untuk menentukan akurasi dari nilai *n* dan *A* yang telah ditentukan (sebagai *test set*).

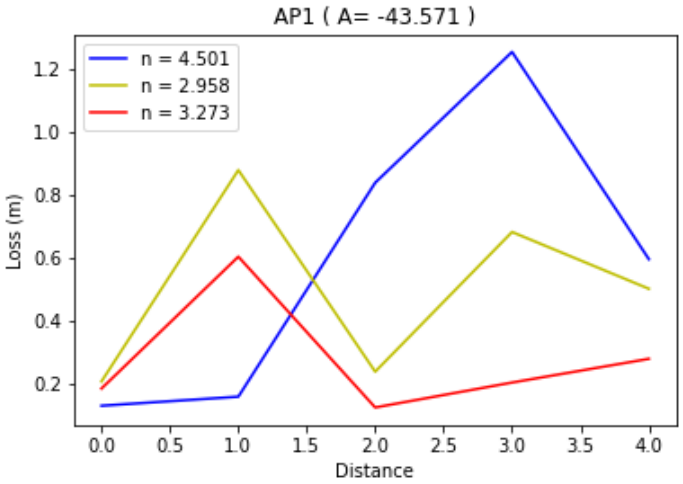
Nilai *A* diambil secara langsung dari rata-rata RSSI yang diambil dari jarak 1 meter, sedangkan nilai *n* diambil dari persamaan pada bab IV dengan input RSSI dari rata-rata 1000 data pada setiap jarak untuk setiap akses poin.

Dari variabel konstan yang telah ditentukan dilakukan pengetesan akurasi dengan mengambil 1000 data pada jarak satu

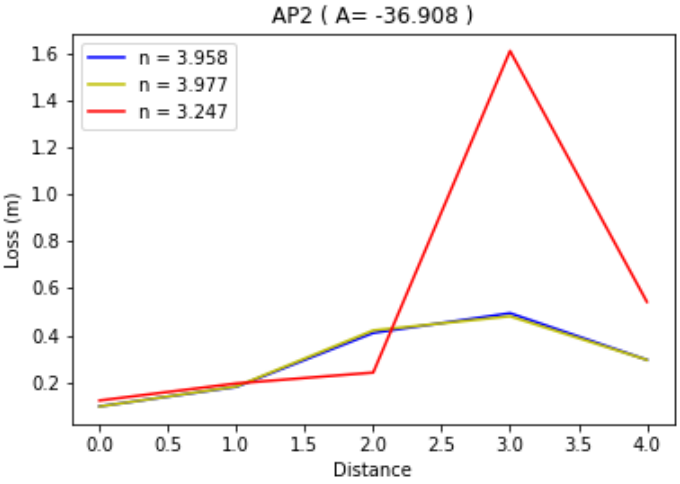
meter, dua meter, dan tiga meter untuk masing-masing akses poin. Pengetesan akurasi dilakukan dengan mengurangi jarak yang didapat dari *signal propagation model* dan jarak nyata, dengan menggunakan kesalahan perhitungan jarak.

Tabel 5.1 Perhitungan jarak dan variabel konstan *signal propagation model*

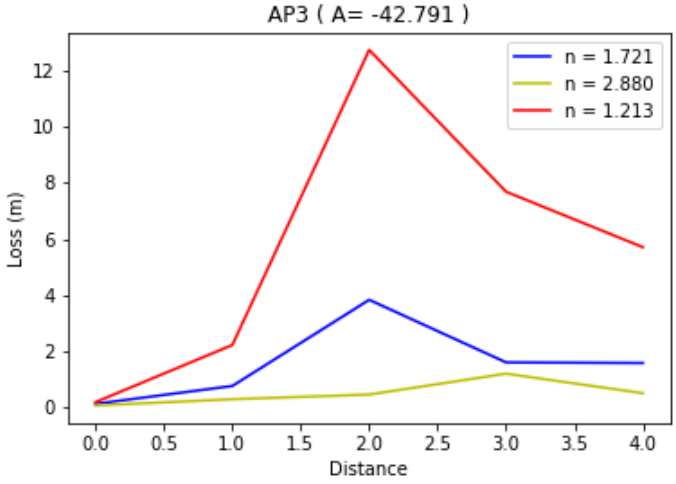
Akses Poin	*A*	Jarak (cm)	Rata-rata RSSI	*n*	Rata-rata kesalahan (cm)
AP1	-43.571	200	-57.121	4.5012	59.56
		300	-57.685	2.9582	50.13
		400	-63.274	3.2726	27.81
AP2	-36.908	200	-48.824	3.9584	29.54
		300	-55.881	3.9766	29.5
		400	-56.459	3.2474	54.27
AP3	-42.791	200	-47.972	1.7211	159.39
		300	-56.534	2.8804	51.87
		400	-50.094	1.2130	571.58



Gambar 5.2 Grafik akurasi variabel konstan *signal propagation model* untuk AP1



Gambar 5.3 Grafik akurasi variabel konstan *signal propagation model* untuk AP2



Gambar 5.4 Grafik akurasi variabel konstan *signal propagation model* untuk AP3

Dari Tabel 5.1 diambil nilai *n* yang memiliki akurasi paling kecil. Didapatkan akurasi dengan rata-rata kesalahan pengukuran kurang dari satu meter.

Diambil *n* dengan nilai 3.272 untuk AP1, 3.976 untuk AP2, 2.880 untuk ap3.

5.2.2 Skenario Uji Coba 2

Pada skenario uji coba 2 dihitung performa *kalman filter* dalam menstabilkan flunktuasi sinyal. Performa ini ditentukan dengan pengujian estimasi jarak kepada akses poin dengan *ground truth* berupa jarak sebenarnya data diambil.

Dilakukan uji coba dengan mengambil 50 data pada satu sampai lima meter terhadap akses poin menggunakan *kalman filter* dengan sampling 1 (tidak menggunakan *kalman filter*), 5 dan 10 dan di rata-rata.

Tabel 5.2 Estimasi jarak dari akses poin 2 dengan pengaplikasian *kalman filter*.

Ground truth (cm)	RSSI			Jarak (cm)		
	n = 1	n = 5	n = 10	n = 1	n = 5	n = 10
100	-40.66	-40.57	-40.33	125.26	124.21	122.10
200	-53.92	-50.23	-50.02	269.67	216.68	214.68
300	-52.12	-51.02	-53.78	241.85	226.78	266.94
400	-55.90	-58.53	-58.64	307.68	354.38	353.53
500	-63.08	-63.77	-63.08	463.49	476.62	458.92

Tabel 5.3 Kesalahan estimasi jarak dari akses poin 2 dengan pengaplikasian *kalman filter*.

Ground truth (cm)	*Error* (cm)		
	n = 1	n = 5	n = 10

100	25.25625	24.21399	22.09567
200	72.09766	17.92784	18.80448
300	58.1545	73.22206	35.79952
400	109.4969	64.30159	48.81642
500	81.19552	47.53677	64.05712
Rata - rata	69.24016	45.44045	37.91464

Dari hasil uji coba terbukti bahwa *kalman filter* dapat meningkatkan akurasi penentuan jarak. Terliat pada tabel 5.3 dimana kesalahan penentuan jarak semakin berkurang seiring dengan bertambahnya sampel yang digunakan di dalam *kalman filter*.

5.2.3 Skenario Uji Coba 3

Dalam skenario uji coba yang ke 3 dihitung performa estimasi dengan *kalman filter*. Diambil 50 data pada masing masing titik pada denah yang ada pada gambar 5.1. Data yang diambil tersebut meliputi RSSI dan estimasi jarak dari masing-masing akses poin, dan estimasi posisi dari data tersebut.

Pengambilan data ini dilakukan tiga kali dengan masing-masing pengujian dilakukan dengan nilai sampling *kalman filter* 0 (tidak menggunakan *kalman filter*), 5, 10.

Pengambilan data dilakukan dengan cara meletakkan perangkat android pada titik uji dan dipindah ke titik uji lain setelah pengambilan data.

Performa estimasi akan ditentukan dengan menghitung jarak estimasi ke *ground truth* dimana data diambil (misalnya pada label a yang berada pada koordinat 100,100). Jarak ini dihitung dengan rumus *euclidean distance*.

Tabel 5.4 Kesalahan estimasi jarak dan koordinat dengan pengambilan data tanpa *kalman filter*.

Label	*Error* AP 1	*Error* AP 2	*Error* AP 1	*Error* X	*Error* Y	*Error*

a	164.63	66.89	151.94	154.91	110.16	206.69
b	46.47	77.99	186.54	66.94	202.41	78.35
c	232.85	133.70	73.48	289.08	249.74	410.84
d	131.89	13.71	67.79	121.18	272.35	156.82
e	70.66	24.03	218.78	100.78	90.11	148.41
f	151.50	145.84	228.37	116.19	109.44	308.08
Avg	133.00	77.03	154.48	141.51	172.37	218.20

Tabel 5.5 Kesalahan estimasi jarak dan koordinat dengan pengambilan data *kalman filter* dengan sampling 5 data.

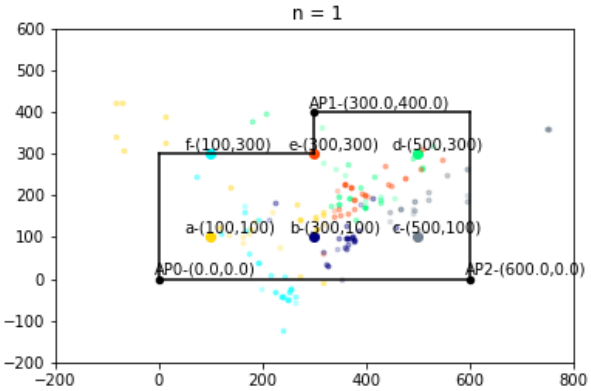
Label	*Error* AP 1	*Error* AP 2	*Error* AP 1	*Error* X	*Error* Y	*Error*
a	46.67	75.06	170.99	152.65	76.59	179.26
b	79.43	76.27	76.21	20.07	195.29	31.26
c	72.80	106.05	69.78	45.91	377.35	82.29
d	153.97	43.39	139.30	140.17	252.43	188.80
e	54.57	25.74	61.72	13.99	50.63	54.58
f	136.26	22.01	58.06	104.66	203.20	118.87
Avg	90.62	58.09	96.01	79.58	192.58	109.18

Tabel 5.6 Kesalahan estimasi jarak dan koordinat dengan pengambilan data *kalman filter* dengan sampling 10 data.

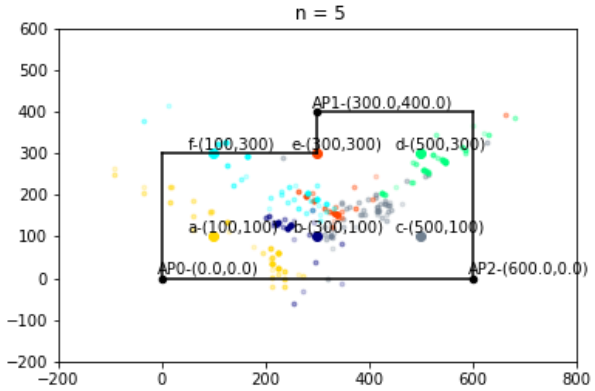
Label	*Error* AP 1	*Error* AP 2	*Error* AP 1	*Error* X	*Error* Y	*Error*
a	36.99	39.99	94.16	76.37	72.08	112.01
b	16.78	42.17	40.69	16.85	185.47	28.93
c	114.49	155.82	68.62	83.94	341.14	108.33
d	41.36	13.32	51.83	38.14	212.54	62.23
e	100.64	21.30	72.90	67.86	73.88	106.82
f	142.99	34.41	76.60	87.61	154.58	124.55

Avg	75.54	51.17	67.47	61.80	173.28	90.48

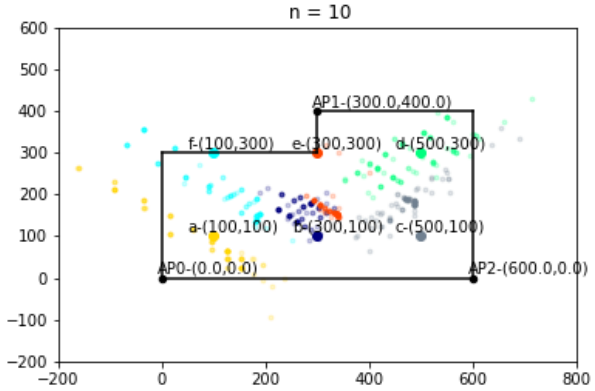
Dari hasil perhitungan kesalahan terlihat bahwa *kalman filter* dengan sampling 10 data memiliki kesalahan paling kecil dengan rata-rata kesalahan 109 cm. Dari uji coba yang dilakukan, dipetakan hasil estimasi posisi ke denah ruangan dengan data yang diambil pada suatu titik digambarkan dengan warna yang sama pada titiknya. Contohnya data yang diambil di titik a, digambarkan dengan warna kuning dan data yang diambil pada titik d digambarkan dengan warna hijau.



Gambar 5.5 Grafik plotting estimasi posisi untuk pengambilan data tanpa *kalman filter*



Gambar 5.6 Grafik plotting estimasi posisi untuk pengambilan data dengan *kalman filter* sampling 5 data



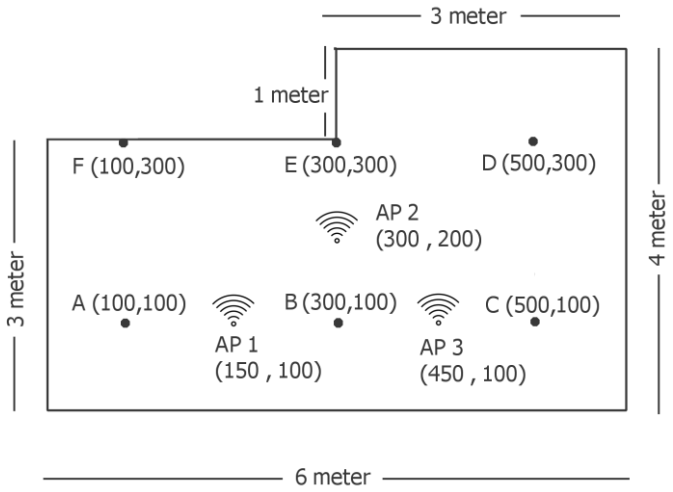
Gambar 5.7 Grafik plotting estimasi posisi untuk pengambilan data dengan *kalman filter* sampling 10 data

Pada gambar 5.5 sampai 5.7 terlihat bahwa semakin besar sampel yang digunakan untuk *kalman filter* semakin sedikit jarak diantara titik untuk label yang sama. Pada hambar 5.7 terlihat bahwa data posisi dengan warna yang sama cenderung berdekatan

satu sama lain dan cenderung berbentuk kotak, yang menandakan kestabilan sinyal. Dilain pihak pada gambar 5.5 titik-titik data tersebar, cenderung membentuk sebuah garis lurus yang diakibatkan karena tidak stabilnya jarak dari salah satu akses poin.

5.2.4 Skenario Uji Coba 4

Dalam skenario uji coba yang ke 4 diuji perubahan peletakan akses poin ke tengah ruangan. Dengan melakukan hal ini akan diuji kredibilitas *trilateration* dengan titik yang diketahui berdekatan satu sama lain.



Gambar 5.8 Peletakan akses poin untuk uji coba 4

Pengambilan data dan perhitungan performa dilakukan dengan cara yang sama pada uji coba 4. Diambil 50 data pada setiap titik dengan kalman filter dengan pengambilan data 1, 5, dan 10 data. Setelah pengambilan data pada satu titik dilakukan pengambilan berikutnya dengan memindahkan perangkat android ke titik berikutnya.

Tabel 5.7 Kesalahan estimasi jarak dan koordinat dengan pengambilan data tanpa *kalman filter*.

Label	*Error* AP 1	*Error* AP 2	*Error* AP 1	*Error* X	*Error* Y	*Error*
a	23.39	159.95	188.54	233.90	366.54	450.79
b	191.58	227.20	208.91	12.91	217.30	35.59
c	246.97	175.97	83.31	135.52	388.40	187.51
d	193.41	41.98	101.37	96.71	222.75	248.77
e	155.56	15.10	230.29	110.49	127.83	175.97
f	179.15	65.22	236.64	261.42	461.82	475.73
Avg	165.01	114.24	174.84	141.82	297.44	262.39

Tabel 5.8 Kesalahan estimasi jarak dan koordinat dengan pengambilan data *kalman filter* dengan sampling 5 data.

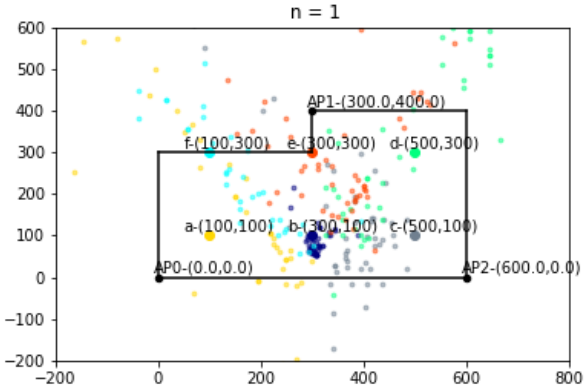
Label	*Error* AP 1	*Error* AP 2	*Error* AP 1	*Error* X	*Error* Y	*Error*
a	45.04	184.81	130.29	71.64	194.27	211.21
b	204.57	221.07	205.10	5.19	230.94	32.64
c	264.02	222.36	56.06	108.89	379.85	153.51
d	135.38	32.76	112.90	116.07	150.18	231.66
e	150.97	9.88	225.31	83.84	115.40	154.05
f	69.60	63.94	289.45	159.45	189.19	184.84
Avg	144.93	122.47	169.85	90.85	209.97	161.32

Tabel 5.9 Kesalahan estimasi jarak dan koordinat dengan pengambilan data *kalman filter* dengan sampling 10 data.

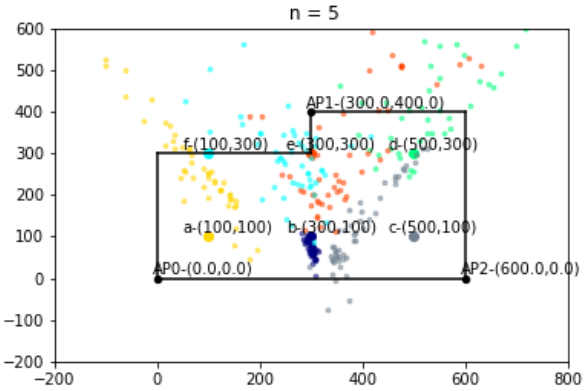
Label	*Error* AP 1	*Error* AP 2	*Error* AP 1	*Error* X	*Error* Y	*Error*
a	40.98	197.76	209.19	68.01	77.18	114.65
b	197.83	222.58	207.18	5.49	227.40	28.38

c	273.00	170.16	58.39	116.44	489.20	157.89
d	219.85	47.58	169.16	78.80	230.48	117.23
e	172.64	8.69	240.48	68.19	109.70	141.95
f	152.42	73.01	286.27	122.99	164.92	194.92
Avg	176.12	119.96	195.11	76.65	216.48	125.84

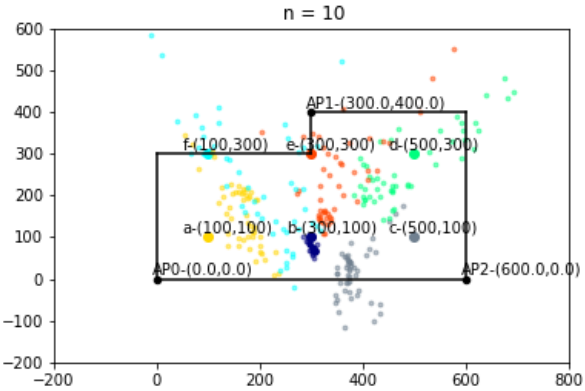
Dari hasil perhitungan kesalahan terlihat bahwa *kalman filter* dengan sampling 10 data memiliki kesalahan paling kecil dengan rata-rata kesalahan 109 cm. Dari uji coba yang dilakukan, dipetakan hasil estimasi posisi ke denah ruangan dengan data yang diambil pada suatu titik digambarkan dengan warna yang sama pada titiknya. Contohnya data yang diambil di titik a, digambarkan dengan warna kuning dan data yang diambil pada titik d digambarkan dengan warna hijau.



Gambar 5.9 Grafik plotting estimasi posisi untuk pengambilan data tanpa *kalman filter* untuk uji coba 4



Gambar 5.10 Grafik plotting estimasi posisi untuk pengambilan data dengan *kalman filter*, sampel data 5 untuk uji coba 4



Gambar 5.11 Grafik plotting estimasi posisi untuk pengambilan data dengan*kalman filter*, sampel data 5 untuk uji coba 4

Pada gambar 5.9 sampai 5.11 terlihat bahwa semakin besar sampel yang digunakan untuk *kalman filter* semakin sedikit jarak diantara titik untuk label yang sama. Pada titik *b* terlihat bahwa titik sangat berkumpul pada satu titik, hal ini dikarenakan jarak pada setiap akses poin sama kecilnya sehingga titik posisi yang didapatkan sangat stabil.

5.3 Evaluasi Umum Skenario Uji Coba

Berikut adalah evaluasi dari hasil uji coba dari masing skenario. Poin yang dievaluasi adalah performa *kalman filter* pada estimasi jarak, estimasi posisi, dan perbandingan estimasi posisi dengan peletakan akses poin yang berbeda dari uji coba yang telah dilakukan.

5.3.1 Estimasi jarak

Dari hasil uji coba 1 didapatkan nilai akurasi *signal propagation model* yang cukup bagus dengan kesalahan pengukuran jarak sampai dengan 29.5 cm. Akurasi ini didapatkan dari pengambilan seribu data pada setiap jarak untuk mendapatkan kekuatan sinyal yang pasti pada jarak tersebut.

Pada hasil uji coba 2 dilakukan pengujian jarak dengan sampel data kecil (1 sampai 10 data) yang dicerminkan di variabel *n* dari *kalman filter*. Untuk *n = 0* berarti pengambilan data tidak menggunakan *kalman filter*, data diambil satu kali. Utuk *n = 5* dan *n = 10* masing-masing diambil 5 dan 10 sampel data dan difilter dengan *kalman filter* untuk mendapatkan sinyal yang stabil.

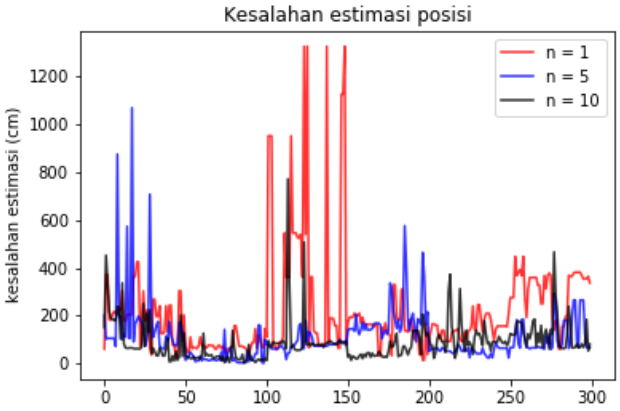
Dari hasil uji coba yang dilakukan, pada tabel 5.3 terlihat bahwa semakin besar sampel yang diambil maka semakin berkurang kesalahan estimasi jarak.

5.3.2 Estimasi posisi

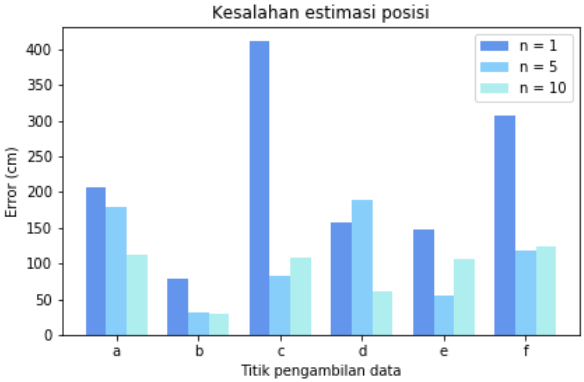
Dari hasil uji coba 3 terlihat bahwa estimasi posisi dengan menggunakan *kalman filter* dengan sampel yang lebih tinggi menghasilkan akurasi yang juga tinggi.

Akurasi yang tinggi ini datang dari minimnya *noise* pada RSSI yang menyebabkan akurasi penentuan jarak meningkat dan pada akhirnya juga meningkatkan akurasi estimasi posisi dengan *trilateration*.

Pada gambar 5.12 terlihat bahwa akurasi estimasi posisi dengan *kalman filter* (garis biru dan hitam) cenderung stabil daripada akurasi tanpa menggunakan *kalman filter* (garis merah).



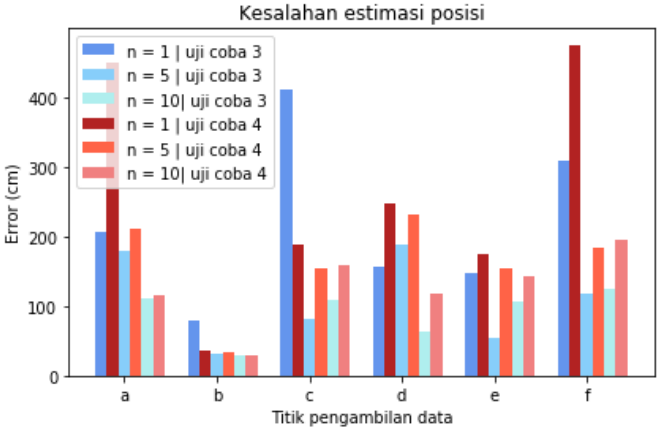
Gambar 5.12 Perbandingan akurasi estimasi posisi.



Gambar 5.13 Perbandingan akurasi estimasi posisi tiap titik uji.

5.3.3 Estimasi posisi dengan peletakan akses poin yang berbeda

Pada uji coba 3 dilakukan estimasi posisi dengan peletakan akses poin pada sudut-sudut ruangan. Hasil ini dibandingkan dengan uji coba 4 dimana akses poin diletakkan ditengah ruangan.



Gambar 5.14 Perbandingan akurasi estimasi posisi uji coba 3 dan 4.

Pada gambar 5.14 terlihat bahwa kesalahan estimasi uji coba 4 lebih tinggi 76% dari kesalahan estimasi uji coba 3. Ini menunjukkan bahwa peletakan akses poin memiliki hubungan dengan akurasi estimasi posisi.

Estimasi posisi memiliki akurasi yang lebih tinggi pada uji coba 3 dimana akses poin diletakkan pada sudut-sudut ruangan.

(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. Dari hasil uji coba didapatkan bahwa dengan menggunakan *signal propagation model* yang telah dikalibrasi didapatkan akurasi sampai dengan 1 meter.
2. Dengan menggunakan *kalman filter* terhadap RSSI yang diterima didapatkan nilai RSSI yang lebih stabil. Terbukti pada uji coba 2 dan 3 dimana akurasi data yang diambil dengan *kalman filter* memiliki rata-rata akurasi yang lebih tinggi dari data yang diambil tanpa *kalman filter*. Terjadi penurunan estimasi kesalahan sampai dengan 54 % dari dari pengambilan data tanpa menggunakan *kalman filter*.
3. Dengan *trilateration* dapat ditentukan posisi perangkat android. Tetapi metode ini tidak toleran terhadap *noise* yang berupa kesalahan estimasi jarak dari masing masing poin dikarnakan metode ini menggunakan 3 jarak dari 3 titik yang diketahui, sehingga dengan metode ini kesalahan penentuan jarak dari satu titik saja dapat mempengaruhi estimasi posisi secara signifikan. Untuk menanggulangi hal ini digunakan *kalman filter* untuk mengurangi *noise* dalam penerimaan sinyal untuk penentuan jarak. Dengan *kalman filter* kesalahan estimasi posisi turun sampai dengan 241%.
4. Peletakan akses poin mempengaruhi akurasi estimasi posisi. Uji coba dengan peletakan akses poin pada sudut-sudut ruangan memiliki kesalahan estimasi posisi yang lebih rendah 76%

daripada uji coba dimana akses poin diletakkan pada tengah - tengah ruangan

6.2 Saran

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

1. Menambahkan peningkatan akurasi pada estimasi posisi dengan mengurangi *noise* pada estimasi posisi itu sendiri, bukan pada fase estimasi jarak. Contohnya dengan menambahkan kalman filter 2 dimensi terhadap posisi yang telah diestimasi dengan menggunakan trilateration.
2. Membuat peta interaktif untuk menunjukkan posisi perangkat android saat ini.

DAFTAR PUSTAKA

[1] V. Ilci, R. M. Alkan, V. E. Gülal, H. Cizmec. (2015), "*Trilateration* Technique for *WiFi*-Based Indoor Localization," Paper presented at ICWMC 2015 : The Eleventh International Conference on Wireless and Mobile Communications.

[2] L. Frederic, C. Philippe, C. Pascal, S. François, B. Oumaya. (2005). "A Friis-based Calibrated Model for *WiFi* Terminal Positioning". Proceedings of IEEE Int. Symp. on A World of Wireless, Mobile and Multimedia Networks. 382 - 387. 10.1109/WOWMOM.2005.2.

[3] 101computing, "Cell Phone *Trilateration* Algorithm." [Online]. Available: https://www.101computing.net/cell-phone-*trilateration*-algorithm/. [Accessed: 30-May-2020].

[4] Yim, J., Jeong, S., Gwon, K. and Joo, J., 2010. "Improvement of *Kalman filter*s for WLAN based indoor tracking". Expert Systems with Applications, 37(1), pp.426-433.

[5] Wikipedia, "Android" [Online]. Available: https://id.wikipedia.org/wiki/Android_(sistem_operasi). [Accessed 2 june 2020].

[6] Wikipedia, "Java" [Online]. Available: https://id.wikipedia.org/wiki/ Java. [Accessed 2 june 2020].

[7] Wikipedia, "Arduino IDE" [Online]. Available: https://en.wikipedia .org/wiki/Arduino_IDE [Accessed 2 june 2020].

[8] Wikipedia, "Python (bahasa pemrograman)" [Online]. Available: https://id.wikipedia.org/wiki/Python_(bahasa_pemrograman) [Accessed 2 june 2020].

[9] Wikipedia, "Spyder (software) " [Online]. Available: https://en.wikipedia.org/wiki/Spyder_(software) [Accessed 2 june 2020].

[10] W. Bulten, A. C. V. Rossum and W. F. G. Haselager, "Human SLAM, Indoor Localisation of Devices and Users," 2016

IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), Berlin, 2016, pp. 211-222, doi: 10.1109/IoTDI.2015.19.

LAMPIRAN A

KODE SUMBER

```
1.  private class Scan WiFi extends AsyncTask<String, Void, String> {
2.      // daemon class to get rssi and freq of certain WiFi
3.      double level1,level2,level3;
4.
5.      @Override
6.      protected String doInBackground(String... params) {
7.          runOnUiThread(new Runnable() {
8.              @Override
9.              public void run() {
10.                 printConsole("scanning data;");
11.                 printConsole("---------- ");
12.
13.             }
14.         });
15.
16.         n_data = Integer.parseInt (t_n_data.getText().toString());
17.         sampling_rate = Integer.parseInt(t_n_KF.getText().toString());
18.         sleep_time = 2000 / sampling_rate;
19.
20.         for (int i = 0; i< n_data; i++) {
21.             if (started == false){
22.                 return "";
23.             }
24.             final int curent_count = i;
25.             runOnUiThread(new Runnable() {
26.                 @Override
27.                 public void run() {
28.                     printConsole("scanning data: " + curent_count);
29.
30.                 }
31.             });
32.             List<ScanResult> results;
33.             double[] RSSI = new double[BSSID.length];
34.             boolean[] is_found = new boolean[BSSID.length];
35.             KalmanFilter[] KF = new KalmanFilter[BSSID.length];
36.
37.
38.             for (int y = 0; y < BSSID.length; y++) {
```

```
39.                 KF[y]= new KalmanFilter(R_kf, Q_kf);
40.                 is_found[y] = false;
41.             }
42.
43.             for (int x = 0; x < sampling_rate; x++) {
44.
    //=================================================================
    ====================
45.                 //getting rssi
46.
    //=================================================================
    ====================
47.                 if (!WiFiManager.startScan()) {
48.                     x -= 1;
49.                     continue;
50.                 }
51.                 results = WiFiManager.getScanResults();
52.                 for (ScanResult scanResult : results) {
53.                     //assigning the rssis
54.                     int c = 0;
55.                     for (int y = 0; y < BSSID.length; y++) {
56.                         if (scanResult.BSSID.equals(BSSID[y])) {
57.                             RSSI[y] = KF[y].filter(scanResult.level); //applying
    kalman filter
58.                             is_found[y] = true;
59.                             final double temp = RSSI[y];
60.                             final double temp2 = scanResult.level;
61.                             final int y_temp = y;
62.                             runOnUiThread(new Runnable() {
63.                                 @Override
64.                                 public void run() {
65.                                     printConsole("y : "+y_temp+" | " + temp + " | "
    + temp2);
66.
67.                                 }
68.                             });
69.                         }
70.
71.                     }
72.                 }
73.                 final int c_temp = x;
74.                 runOnUiThread(new Runnable() {
75.                     @Override
```

```
76.                     public void run() {
77.                         printConsole("---------- "+c_temp+" ----------" );
78.
79.                     }
80.                 });
81.                 sleep(sleep_time);
82.             }
83.
84.
    //=================================================
    ====================
85.             //checking all rssi
86.
    //=================================================
    ====================
87.             boolean all_bssid_found = true;
88.             for (int y = 0; y < BSSID.length; y++) {
89.                 if (is_found[y] == false) {
90.                     all_bssid_found = false;
91.                     break;
92.                 }
93.             }
94.             if (all_bssid_found == false) {
95.                 runOnUiThread(new Runnable() {
96.                     @Override
97.                     public void run() {
98.
    printConsole("====================================");
99.                         printConsole("rssi missing");
100.
    printConsole("====================================");
101.                        double[] dummy = {-1, -1};
102.                        changeCoor(dummy);
103.                    }
104.                });
105.            }
106.
107.
    //=================================================
    ====================
108.            //process the rssi
```

```
109.
    //=================================================
    ====================
110.            double[] dist = new double[BSSID.length];
111.            final double distance1, distance2, distance3;
112.
113.            for (int y = 0; y < BSSID.length; y++) {
114.                dist[y] = getDistance(KF[y].lastMeasurement(), A_ap[y],
    N_ap[y]) * 100; //get distance
115.            }
116.
117.            final double[] coord = getCoord(dist); //apply trilateration
118.
119.
    //=================================================
    ====================
120.            //print to console
121.
    //=================================================
    ====================
122.            distance1 = dist[0];
123.            distance2 = dist[1];
124.            distance3 = dist[2];
125.            level1 = KF[0].lastMeasurement();
126.            level2 = KF[1].lastMeasurement();
127.            level3 = KF[2].lastMeasurement();
128.            runOnUiThread(new Runnable() {
129.                @Override
130.                public void run() {
131.                    changeDist(Math.round(distance1 * 100) / 100 + "",
    Math.round(distance2 * 100) / 100 + "", Math.round(distance3 * 100) /
    100 + "");
132.                    changeCoor(coord);
133.                    label_log = label.getText().toString();
134.                }
135.            });
136.
137.
    //=================================================
    ====================
138.            //append log
```

```
139.
    //==================================================
    ====================
140.            ArrayList<String> temp = new ArrayList<>();
141.            temp.add("" + label_log);
142.            temp.add("" + sampling_rate);
143.            for (int y = 0; y < BSSID.length; y++) {
144.                temp.add("" + KF[y].lastMeasurement());
145.            }
146.            for (int y = 0; y < BSSID.length; y++) {
147.                temp.add("" + dist[y]);
148.            }
149.            temp.add("" + coord[0]);
150.            temp.add("" + coord[1]);
151.            String[] temp2 = new String[temp.size()];
152.            temp2 = temp.toArray(temp2);
153.            log.add(temp2);
154.        }
155.        return "";
156.    }
157.
158.    @Override
159.    protected void onPostExecute(String result) {
160.        super.onPostExecute(result);
161.        started = false;
162.        start.setText("Test");
163.        runOnUiThread(new Runnable() {
164.            @Override
165.            public void run() {
166.                printConsole("---------- ");
167.                printConsole("scanning done;");
168.                printConsole("---------- ");
169.
170.            }
171.        });
172.
173.    }
174. }
```

Kode Sumber A.1 *Class* untuk mengambil sampel data RSSI (Android)

```
1.  import math
2.  import csv
3.  from os.path import join
4.  import numpy as np
5.  import pandas as pd
6.  import matplotlib.pyplot as plt
7.
8.  def loadDataset(name):
9.      data = []
10.     with open(name, newline='') as f:
11.         reader = csv.reader(f)
12.         reader = list(reader)
13.     data = reader.copy()
14.     return data
15.
16. def getN(rssi, a,d):
17.     n =( -1 * (rssi -a)/ (10 * math.log10(d)))
18.     return   float("{:.4f}".format(n))
19.
20. def averageDataset(filename, BSSID):
21.     #load dataset
22.     dataset = loadDataset(filename)
23.     #extract rssi
24.     rssi  = []
25.     for item in dataset:
26.         count = 0
27.         for atom in item:
28.             if atom == BSSID:
29.                 rssi.append(int(item[count+1]))
30.             count+=1
31.     #get average rssi
32.     avg = sum(rssi) / len (rssi)
33.     #return average rssi
34.     return avg
35.
36. def printCSV(filename, mylist):
37.     with open(filename, 'w', newline="") as myfile:
38.         wr = csv.writer(myfile)
39.         for item in mylist:
40.             wr.writerow(item)
41.
42.
43. def getDistance(rssi, a, n):
```

```
44.     return pow(10, -1*((rssi-a)/(10*n)))
45.
46.  def doDist(filename ,a , n, BSSID):
47.     data = loadDataset(filename)
48.
49.     #extract rssi
50.     rssi  = []
51.     for item in data:
52.        count = 0
53.        for atom in item:
54.           if atom == BSSID:
55.              rssi.append(int(item[count+1]))
56.           count+=1
57.
58.     dist = []
59.     for item in rssi:
60.        dist.append(getDistance(item,a,n))
61.     return dist
62.
63.  def doAcc(dataset, trueDist):
64.     acc = []
65.     for item in dataset:
66.        acc.append(abs(item - trueDist))
67.     return acc
68.
69.  def getStatistic(acc):
70.     dictElement = {}
71.     for item in acc:
72.        if item not in dictElement:
73.           dictElement[item] = 1
74.        else:
75.           dictElement[item] += 1
76.     avg = sum(acc) / len(acc)
77.     return dictElement, avg
78.
79.  def getAcc (filename, Tdist, a, n, bssid):
80.     dist = doDist(filename, a, n, bssid)
81.     acc = doAcc(dist, Tdist)
82.     stat, avg = getStatistic(acc)
83.
84.     return float("{:.4f}".format(avg))
85.
86.
```

```
87.  if __name__ == "__main__":
88.
89.      bssid = ["c0:25:e9:7a:e6:2f", "14:4d:67:98:2b:64",
     "18:0f:76:91:f2:72"]
90.      ap = [1,2,3]
91.      ssid = ["TP-LINK_E630", "TOTOLINK_N210RE", "D-Link_DIR-
     612"]
92.      x_ap = [0, 300, 600 ]
93.      y_ap = [0, 400, 0]
94.      """
95.      name format: datasetxx/apy-z-w.csv
96.      """
97.      datasetSerial = ""
98.      z= [1,2,3,4]
99.      w_train = 1
100.     w_test = 2
101.
102.
103.
104.     """
105.     ================================
106.     Geting A
107.     ========================================
108.     """
109.     a = []
110.     count = 0
111.     for item in range(len(bssid)):
112.         filename = (join("dataset{}".format(datasetSerial),
113.                          "ap{}-{}-{}.csv".format(ap[count], 1, w_train)))
114.         a.append(averageDataset(filename, bssid[count]))
115.         count+=1
116.
117.     """
118.     ================================
119.     Geting n
120.
121.     ....|dist
122.     ap
123.
124.     ========================================
125.     """
126.
127.     avg2 = []
```

```
    n = []
    count = 0
    countZ = 1
    for count in range(len(ap)):
        n_now = []
        for countZ in range(1, len(z)):

            filename = (join("dataset{}".format(datasetSerial),
                             "ap{}-{}-{}.csv".format(ap[count], z[countZ],
w_train)))
            avg = averageDataset(filename, bssid[count])
            avg2.append(avg)

            n_now.append(getN(avg, a[count], z[countZ]))
        n.append(n_now)



    """
    ================================
    testing

    ....|dist
    ap

    ======================================
    """
    #ap1
    stat = []
    stat_avg = []
    for ap_now in ap: #itterating each ap

        stat_now = []
        stat_now_avg= []
        for zCount in range (len(z)-1): #each n for current ap
            n_now = n[ap_now-1][zCount]
            avg= []

            for dCount in range(len(z)): #calculating the accuracy of a and n
to data in various distance
                filename = (join("dataset{}".format(datasetSerial),
                             "ap{}-{}-{}.csv".format(ap_now, z[dCount], w_test)))
```

```
168.                avg.append(getAcc(filename, z[dCount], a[ap_now-1], n_now,
    bssid[ap_now-1]))
169.
170.            avg_avg =float("{:.4f}".format(sum(avg)/len(avg)))
171.            avg.append(avg_avg)
172.            stat_now_avg.append(avg_avg)
173.            stat_now.append(avg)
174.        stat.append(stat_now)
175.        stat_avg.append(stat_now_avg)
176.
177.
178.
179.    """
180.    ================================
181.    Visualizing
182.    ========================================
183.    """
184.    lines = ["b-", "y-", "r-", "g-"]
185.    titles = ["AP1", "AP2", "AP3"]
186.    xlabel = "Distance"
187.    ylabel = "Loss (m)"
188.    titleC = 0
189.    for ap_now in range(len(stat)): #each ap
190.        lineC = 0
191.        for n_now in range(len(stat[ap_now])): #each n
192.            x = list(range(len(stat[ap_now][n_now])))
193.            plt.plot(x, stat[ap_now][n_now], lines[lineC], label= "n = %.3f"
    % n[ap_now][n_now])
194.            plt.legend()
195.            lineC+=1
196.        plt.xlabel(xlabel)
197.        plt.ylabel(ylabel)
198.        plt.title(titles[titleC] + " ( A= {} )".format(a[ap_now]))
199.        plt.savefig(join( join( "output", "plot"), titles[titleC] + " - signal
    propagation"))
200.        plt.show()
201.        titleC+=1
202.
203.    df = []
204.    for item in  range(len(ap)):
205.        for atom in range(0,3):
206.            df_n=["AP{}".format(item+1), a[item], (atom+1) *100]
207.            df_n.append(avg2[item*3 + atom])
```

```
208.            df_n.append(n[item][atom])
209.            df_n.append(sum( stat[item][atom]) / len( stat[item][atom]) *
    100)
210.            df.append(df_n)
211.
212.    df = pd.DataFrame(columns = ['Akses Poin', 'A', 'Jarak (cm)', "Rata -
    rata RSSI",
213.                                 "n", 'Rata - rata kesalahan'], data = df)
214.
215.
216.    """
217.    ================================
218.    Selecting best data
219.    ========================================
220.    """
221.    bestN = [0 for i in range(len(ap))]
222.    cur_ap = 0
223.    bestLoss = [0 for i in range(len(ap))]
224.    for each_ap in stat:
225.        min_loss = np.inf
226.        cur_n = 0
227.        for each_n in each_ap:
228.            if(each_n[-1] < min_loss):
229.                bestN[cur_ap] = n[cur_ap][cur_n]
230.                min_loss = each_n[-1]
231.                bestLoss[cur_ap] = each_n[-1]
232.            cur_n+=1
233.        cur_ap+=1
234.    """
235.    ================================
236.    Printing
237.    ========================================
238.    """
239.    printed = [["SSID", "BSSID", "A", "n", "x", "y"]]
240.
241.    for i in range(len(ap)):
242.        printed.append([ssid[i], bssid[i], a[i], bestN[i], x_ap[i], y_ap[i]])
243.
244.    printCSV(join ("output", "constant variables.csv"),printed)
245.    df.to_csv(join("output", "acc signal propagation.csv"))
246.
```

```
247.
```

Kode Sumber A.2 Kode sumber untuk menentukan konstan *signal propagation model* (python)

```
1.  package com.example.kalmanfilterforrssi;
2.
3.  public class KalmanFilter {
4.      /**
5.       * Create 1-dimensional kalman filter
6.       * @param  {Number} options.R Process noise
7.       * @param  {Number} options.Q Measurement noise
8.       * @param  {Number} options.A State vector
9.       * @param  {Number} options.B Control vector
10.      * @param  {Number} options.C Measurement vector
11.      * @return {KalmanFilter}
12.      */
13.     double R;
14.     double Q;
15.     double cov;
16.     double x;
17.     double count;
18.
19.     public KalmanFilter(double R, double  Q) {
20.         this.R = R; // noise power desirable
21.         this.Q = Q; // noise power estimated
22.         this.count =0;
23.         this.cov = -1;
24.         this.x = -1; // estimated signal without noise
25.     }
26.
27.     /**
28.      * Filter a new value
29.      * @param  {Number} z Measurement
30.      * @param  {Number} u Control
31.      * @return {Number}
32.      */
33.     public double filter(int z ) {
34.         this.count+=1;
35.         if (this.x == -1) {
36.             this.x =   z;
37.             this.cov =  this.Q ;
```

```
38.         }
39.         else {
40.
41.             // Compute prediction
42.             double predX = this.predict();
43.             double predCov = this.uncertainty();
44.
45.             // Kalman gain
46.             double K = predCov * (1 / (predCov  + this.Q));
47.
48.             // Correction
49.             this.x = predX + K * (z - ( predX));
50.             this.cov = predCov - (K *  predCov);
51.         }
52.
53.         return this.x;
54.     }
55.
56.     /**
57.      * Predict next value
58.      * @param  {Number} [u] Control
59.      * @return {Number}
60.      */
61.     public double predict() {
62.         return this.x ;
63.     }
64.
65.     /**
66.      * Return uncertainty of filter
67.      * @return {Number}
68.      */
69.     public double uncertainty() {
70.         return this.cov + this.R;
71.     }
72.
73.     /**
74.      * Return the last filtered measurement
75.      * @return {Number}
76.      */
77.     public double lastMeasurement() {
78.         return this.x;
79.     }
80.
```

```
81.     /**
82.      * Set measurement noise Q
83.      * @param {Number} noise
84.      */
85.     public void setMeasurementNoise(double noise) {
86.         this.Q = noise;
87.     }
88.
89.     /**
90.      * Set the process noise R
91.      * @param {Number} noise
92.      */
93.     public void setProcessNoise(double noise) {
94.         this.R = noise;
95.     }
96. }
```

Kode Sumber A.3 Kode sumber *kalman filter* (android)

```
1.  from os.path import join
2.  import csv
3.  import math
4.  import pandas as pd
5.  import matplotlib.pyplot as plt
6.  import numpy as np
7.
8.  def loadDataset(name):
9.      data = []
10.     with open(name, newline='') as f:
11.         reader = csv.reader(f)
12.         reader = list(reader)
13.     data = reader.copy()
14.     return data
15.
16.
17. def getStatistic(acc):
18.     dictElement = {}
19.     for item in acc:
20.         if item not in dictElement:
21.             dictElement[item] = 1
22.         else:
23.             dictElement[item] += 1
```

```
24.     return dictElement
25.
26.  def avg(lists):
27.     return sum(lists) / len(lists)
28.
29.  def euclidDist(v1, v2):
30.     dist = 0
31.     for item in range(len(v1)):
32.        dist += (v1[item] - v2[item]) **2
33.     return math.sqrt(dist)
34.
35.  def drawRoomOnly():
36.     import matplotlib.pyplot as plt
37.     #drawing the room
38.     for item in range(len(x)): #drawing the AP
39.        plt.scatter(x[item], y[item], s = 20, color = "k",)
40.        plt.annotate("AP{}-({},{})".format(item, x[item], y[item]), (x[item]
     -10, y[item] + 10))
41.
42.     lines = [[[  0,   0], [600,   0]], #room lines
43.              [[600,   0], [600, 400]],
44.              [[600, 400], [300, 400]],
45.              [[300, 400], [300, 300]],
46.              [[300, 300], [  0, 300]],
47.              [[  0, 300], [  0,   0]],
48.              ]
49.
50.     trueLoc = [ [100, 100], [300,100],
51.                 [500, 100], [500,300],
52.                 [300, 300], [100,300],]
53.     label_point = ['a', 'b' , 'c', 'd','e', 'f']
54.
55.     for item in lines: #drawing room lines
56.        plt.plot([item[0][0], item[1][0]], [item[0][1], item[1][1]], color =
     "k")
57.
58.     color = ["gold", "navy", "slategrey", "springgreen", "orangered",
     "aqua"]
59.     for item in range(len(trueLoc)):#Drawing the point
60.        plt.scatter(trueLoc[item][0],trueLoc[item][1], color = color[item])
61.        plt.annotate("{}-({},{})".format(label_point[item],
62.                                         trueLoc[item][0],trueLoc[item][1]),
63.                                         (trueLoc[item][0] - 50,trueLoc[item][1] + 10))
```

```
64.
65.  def drawRoom(scattered, title):
66.      import matplotlib.pyplot as plt
67.      drawRoomOnly()
68.      color = ["gold", "navy", "slategrey", "springgreen", "orangered",
     "aqua"]
69.      est = scattered
70.      for item in est:#Drawing the estimated points
71.
72.          plt.scatter(float(item[1]), float(item[2]),
73.                      color = color[ord(item[0]) - ord('a')],
74.                      alpha = "0.2",
75.                      s = 8)
76.      plt.title(title)
77.      plt.xlim([-200, 800])
78.      plt.ylim([-200, 600])
79.      plt.savefig(join(join("output", "plot"), title + "-room accuracy"))
80.      plt.show()
81.
82.
83.  def trilateration(r,x,y):
84.      x1 = x[0]
85.      x2 = x[1]
86.      x3 = x[2]
87.      r1 = r[0]
88.      r2 = r[1]
89.      r3 = r[2]
90.      y1 = y[0]
91.      y2 = y[1]
92.      y3 = y[1]
93.      A = 2*x2 - 2*x1;
94.      B = 2*y2 - 2*y1;
95.      C = r1*r1 - r2*r2 - x1*x1 + x2*x2 - y1*y1 + y2*y2;
96.      D = 2*x3 - 2*x2;
97.      E = 2*y3 - 2*y2;
98.      F = r2*r2 - r3*r3 - x2*x2 + x3*x3 - y2*y2 + y3*y3;
99.      x = (C*E - F*B) / (E*A - B*D);
100.     y = (C*D - A*F) / (B*D - A*E);
101.     return x,y
102.
103. def getDistance(rssi, a, n):
104.     return pow(10, -1*((rssi-a)/(10*n)))
105.
```

```
106. if __name__ == "__main__":
107.     """
108.     ==================================
109.     loading constant variables
110.     ==================================
111.     """
112.
113.     df = pd.read_csv(join('dataset', 'dataset.csv'))
114.     constant =loadDataset(join(join(join("..","Getting constant
        variables"),"output"), "constant variables.csv"))
115.     a = []
116.     n = []
117.     x = []
118.     y = []
119.     bssid = []
120.     for item in constant[1:]:
121.         bssid.append(item[1])
122.         a.append(float(item[2]))
123.         n.append(float(item[3]))
124.         x.append(float(item[4]))
125.         y.append(float(item[5]))
126.
127.     trueLocX = {'a' : 100,
128.                 'b' : 300,
129.                 'c' : 500,
130.                 'd' : 500,
131.                 'e' : 300,
132.                 'f' : 100}
133.
134.     trueLocY = {'a' : 100,
135.                 'b' : 100,
136.                 'c' : 100,
137.                 'd' : 300,
138.                 'e' : 300,
139.                 'f' : 300}
140.
141.     """
142.     ==================================
143.     Calculating accuracy
144.     ==================================
145.     """
146.     acc = [[],[],[],[],[],[]]
147.     for item in df.values:
```

```
148.        acc[0].append(abs(item[5] - euclidDist([trueLocX[item[0]],
    trueLocY[item[0]]], [x[0],y[0]]) ))
149.        acc[1].append(abs(item[6] - euclidDist([trueLocX[item[0]],
    trueLocY[item[0]]], [x[1],y[1]]) ))
150.        acc[2].append(abs(item[7] - euclidDist([trueLocX[item[0]],
    trueLocY[item[0]]], [x[2],y[2]]) ))
151.        acc[3].append(abs(item[8] - trueLocX[item[0]]))
152.        acc[4].append(abs(item[9] - trueLocX[item[0]]))
153.        acc[5].append(euclidDist([item[8], item[9]], [trueLocX[item[0]],
    trueLocY[item[0]]]))
154.
155.    df['acc dist 1'] = acc[0]
156.    df['acc dist 2'] = acc[1]
157.    df['acc dist 3'] = acc[2]
158.
159.    df['acc x'] = acc[3]
160.    df['acc y'] = acc[4]
161.    df['acc coord'] = acc[5]
162.
163.    for i in range(0,11,5):
164.        cur = df[df["n"] == i]
165.        est = cur[['label','x', 'y']].values
166.        drawRoom(est, " n = {}".format(i ))
167.
168.    df.to_csv(join('output', 'analisis ips.csv'))
169.
170.    acc_point= []
171.    for item in range(0,11,5):
172.        cur = df[df['n'] == item].sort_values(by=['label'])
173.
174.        for i in  ['a', 'b' , 'c', 'd','e', 'f']:
175.            means_n = [i, item]
176.            cur_label = cur[cur['label'] == i]
177.            means_n.append(sum(cur_label['acc dist 1']) / len(cur_label['acc
    dist 1']))
178.            means_n.append(sum(cur_label['acc dist 2']) / len(cur_label['acc
    dist 2']))
179.            means_n.append(sum(cur_label['acc dist 3']) / len(cur_label['acc
    dist 3']))
180.            means_n.append(sum(cur_label['acc x']) / len(cur_label['acc x']))
181.            means_n.append(sum(cur_label['acc y']) / len(cur_label['acc y']))
182.            means_n.append(sum(cur_label['acc coord']) / len(cur_label['acc
    coord']))
```

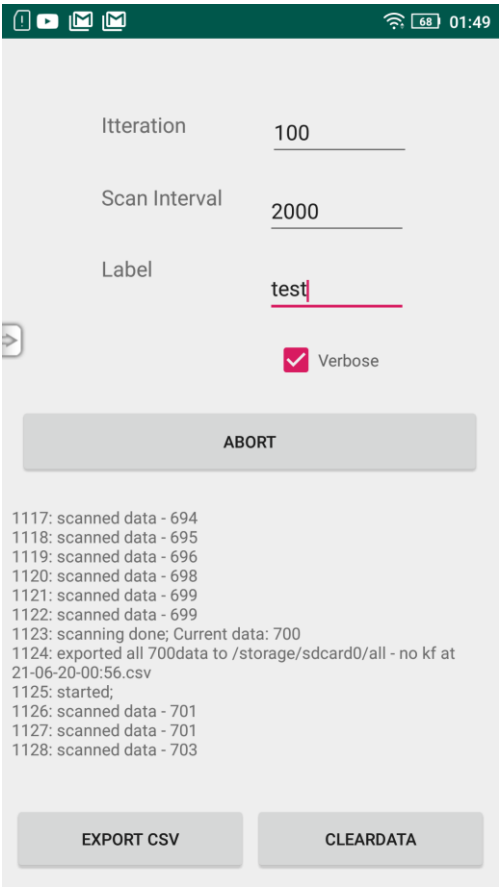
```
183.            acc_point.append(means_n)
184.
185.
186.    acc_point = pd.DataFrame(columns = ['label', 'n',
187.                                        'acc dist 1' , 'acc dist 2', 'acc dist 3',
188.                                        'acc x', 'acc y',
189.                                        'acc coord'],
190.                             data = acc_point)
191.    acc_point.to_csv(join('output', 'analisis ips per point.csv'))
192.
193.
194.    """
195.    ==================================
196.    Visualizing
197.    ==================================
198.    """
199.
200.    color = ["red", "blue", "black"]
201.    for item in range(0,11,5):
202.        cur = df[df['n'] == item].sort_values(by=['label'])
203.        plt.plot(list(range(599)), cur['acc coord'].values[:599],
204.                 color = color[int(item/5)], alpha =0.8, label = "n =
    {}".format(item))
205.    plt.ylabel("kesalahan estimasi (cm)")
206.    plt.legend()
207.
208.    plt.title("Kesalahan estimasi posisi")
209.    plt.savefig(join(join('output','plot'), 'kesalahan estimasi posisi'))
210.    plt.show()
211.
212.    #kesalahan per poin
213.
214.    labels = ['a', 'b' , 'c', 'd','e', 'f']
215.
216.    x = np.array(list(range(0,11,2)))
217.    width = 0.5  # the width of the bars
218.
219.    fig, ax = plt.subplots()
220.    rects1 = ax.bar(x  - width , acc_point[acc_point['n'] ==0]['acc coord'],
    width, label='n = 0', color = "cornflowerblue")
221.    rects2 = ax.bar(x , acc_point[acc_point['n'] ==5]['acc coord'], width,
    label='n = 5', color = "lightskyblue")
```

```
222.    rects3 = ax.bar(x  + width, acc_point[acc_point['n'] ==10]['acc
        coord'], width, label='n = 10',  color = "paleturquoise")
223.
224.    ax.set_ylabel('Error (cm)')
225.    ax.set_title('Kesalahan estimasi posisi')
226.    ax.set_xlabel('Titik pengambilan data')
227.    ax.set_xticks(x)
228.    ax.set_xticklabels(labels)
229.    ax.legend()
230.
231.    fig.tight_layout()
232.    plt.savefig(join(join('output','plot'), 'kesalahan estimasi posisi bar'))
233.    plt.show()
234.
235.
```

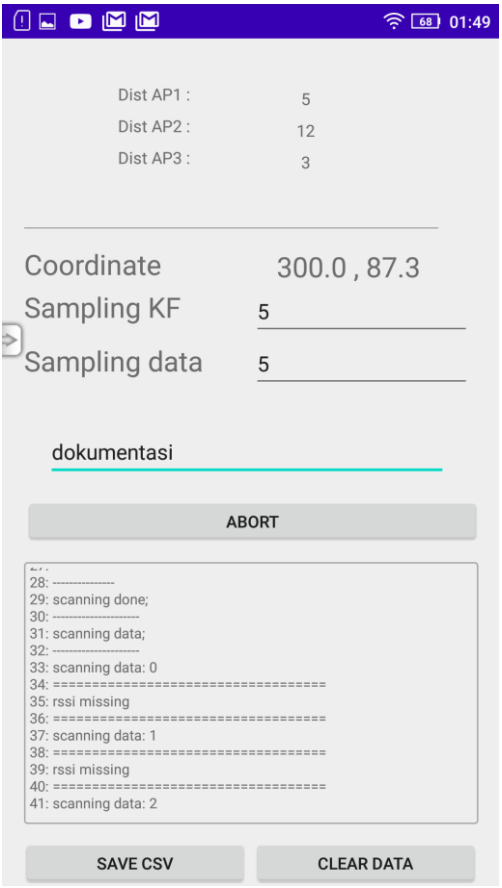
Kode Sumber A.4 Kode sumber untuk menentukan akurasi IPS

LAMPIRAN B

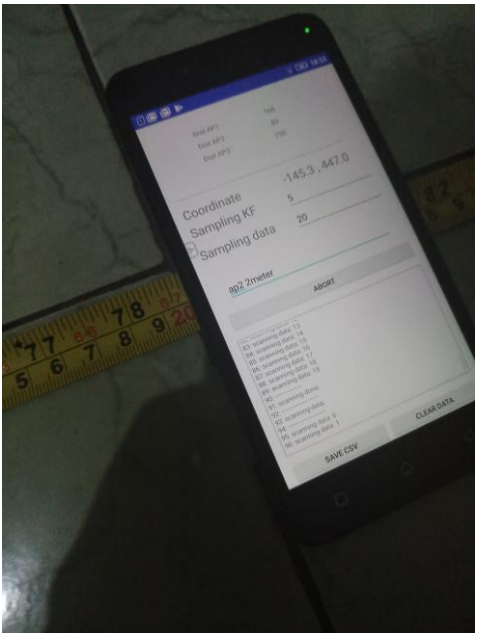
DOKUMENTASI IMPLEMENTASI



Gambar B.1 Aplikasi untuk mengambil sampel data untuk kalibrasi *signal propagation model*



Gambar B.2 Aplikasi *indoor positioning system* dengan kalman filter



Gambar B.3 Ujicoba *kalman filter* untuk penentuan jarak.

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Agung Dwi Wicaksono lahir di Malang pada tanggal 8 Desember 1998. Penulis menempuh Pendidikan formal di MI Al-Huda (2004-2010), SMPN 3 Malang (2010-2013), SMAN 1 Kota Solok (2013-2016), Sesudah lulus dari SMAN 3 Malang, Penulis melanjutkan program studi sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember, Surabaya (2016-2020).

Selama kuliah di Departemen Informatika ITS, Penulis mengambil bidang minat Komputasi Berbasis Jaringan (KBJ). Penulis pernah menjadi asisten dosen untuk mata kuliah Dasar Pemrograman (2018-2019), Bahasa Pemrograman C# PIKTI (2018-2019), Struktur Data (2019-2020) dan Dasar Pemrograman (2019-2020). Selama menempuh perkuliahan, Penulis juga aktif mengikuti organisasi seperti Schematics (2016-2017). Selama masa kuliah penulis pernah mendapat juara satu pada lomba hackaton BRI YoungHackr 2020. Penulis dapat dihubungi melalui surel di agung.dwi.temp@gmail.com.