Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

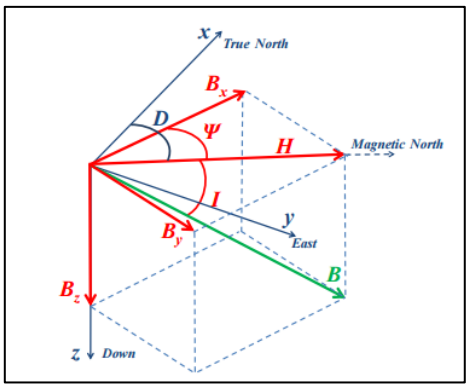
BAB II

LANDASAN TEORI

2.1 Medan Magnet Bumi

Medan magnet bumi adalah fenomena planet yang berlangsung secara alami (Azfal, 2011). Medan magnet bumi didasari atas teori dinamo dan terbentuk oleh pergerakan-pergerakan pada inti luar Bumi yang mengandung *molten iron* (Campbell, 2001).

Medan magnet bumi dapat direpresentasikan sebagai vektor tiga dimensi yang dapat dibaca menggunakan sistem koordinat Kartesian. Sumbu X merupakan sudut menuju utara geografis bumi, sumbu Y merupakan sudut menuju timur geografis bumi, dan sumbu Z mengarah ke bawah dan atas. Sudut deklinasi D merupakan sudut antara kutub utara bumi secara geografis dengan kutub utara magnet. Sudut inklinasi I merupakan sudut antara medan magnet B dan dataran horizontal (Afzal dkk., 2011). Gambar 2.1 menggambarkan gambaran sudut-sudut yang dapat didapat dari medan magnet bumi.



Gambar 2.1 Medan Magnet Bumi Sistem Koordinat Kartesian (Afzal dkk., 2011)

Berdasarkan Gambar 2.1, sudut deklinasi D dapat dihitung dengan rumus

$$D = \tan^{-1}(\frac{Y}{X}) \quad \ldots(2.1)$$

total kekuatan medan magnet F adalah

$$F = \sqrt{X^2 + Y^2 + Z^2}, \quad \ldots(2.2)$$

total intensitas horizontal medan magnet H adalah

$$H = \sqrt{X^2 + Y^2}, \quad \ldots(2.3)$$

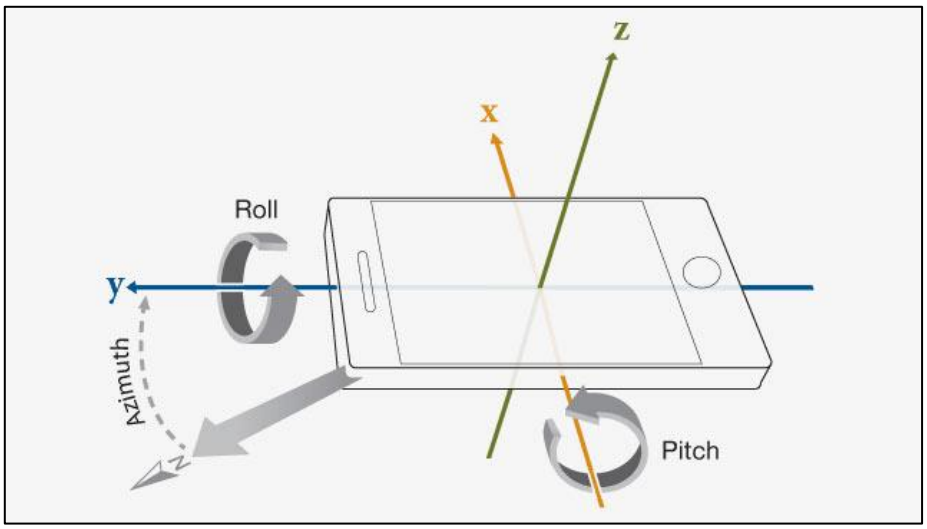
dan sudut inklinasi I dapat dihitung dengan rumus

$$I = \tan^{-1}(\frac{Z}{H}) \quad \ldots(2.4)$$

dimana H merupakan kekuatan magnetik pada bidang horizontal (Afzal dkk., 2011). Nilai D dan I dihitung dalam satuan *degree* atau derajat dan komponen lainnya dihitung dalam satuan *microtesla* (μT).

2.2 Magnetometer

Magnetometer adalah alat atau sensor yang digunakan untuk mengukur nilai magnetik dari suatu objek atau pada suatu lokasi. Sensor magnetometer didasari atas *Hall effect sensor*, yang mengukur tingkat medan magnet bumi sepanjang sumbu X, Y, dan Z dengan menggunakan satuan *microtesla* (μT) (Nagpal, 2016). Sumbu X berada pada lintang horizontal dan mengarah ke kanan, Y berada pada bidang vertikal dan mengarah ke depan, dan Z mengarah ke bawah dan atas. Gambar 2.2 menggambarkan sumbu sensor magnetometer untuk perangkat *smartphone*.



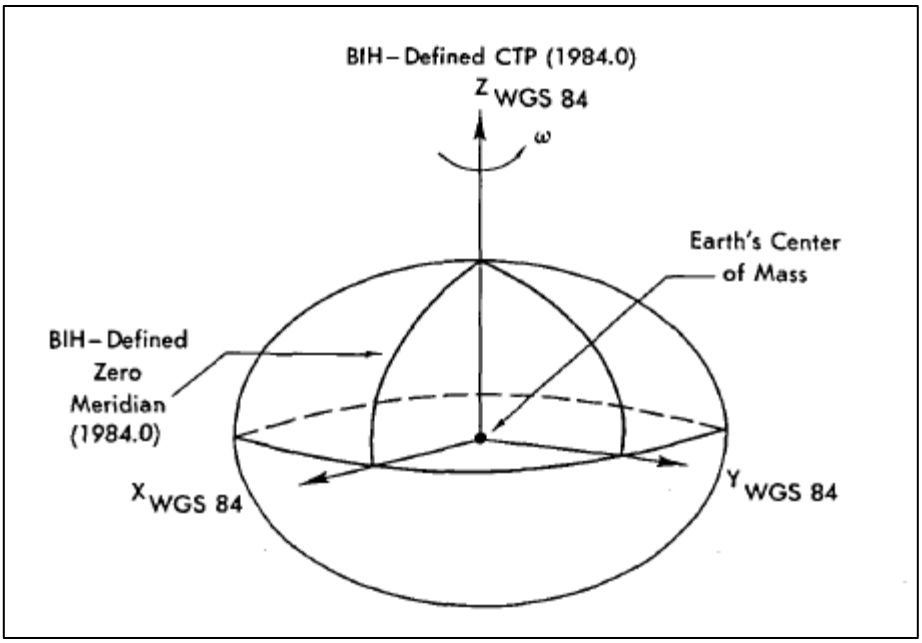
Gambar 2.2 Sumbu X, Y, dan Z Sensor Magnetometer (MathWorks, 2013)

2.3 World Geodetic System 1984

World Geodetic System 1984 (WGS 84) adalah sebuah sistem referensi terestrial dan datum berbasis parameter yang menjelaskan ukuran, bentuk, gravitasi, dan kekuatan medan magnet dari Bumi (Abidin, 2007). Datum ini adalah sistem referensi yang digunakan oleh Departemen Pertahanan Amerika Serikat (DoD) dan merupakan sebuah acuan dasar yang banyak digunakan dalam bidang katrografi (pembuatan peta) dan navigasi (Abidin, 2007). Salah satu teknologi yang memanfaatkan WGS 84 adalah *Global Positioning System* atau GPS untuk penentuan lokasi.

Sistem koordinat WGS 84 adalah Sistem Terestrial Konvensional (*Conventional Terrestrial System*), yang direalisasikan dengan memodifikasi kerangka referensi yang digunakan oleh sistem satelit Doppler (NSWC 9Z-2), yaitu parameter pusat (titik nol) sistem koordinat dan skalanya, serta merotasikannya sehingga meridian referensinya berimpit dengan meridian nol yang didefinisikan oleh Bureau

International de I'Heure (BIH) (DMA, 1991). Gambar 2.3 menggambarkan sistem koordinat *World Geodetic System* 1984.



Gambar 2.3 Sistem Koordinat World Geodetic System 1984 (Kumar, 1988)

Sumbu-sumbu dan karakteristik pada sistem koordinat WGS 84 sebagai berikut (Kumar, 1988):

1. *Origin*: Pusat massa bumi
2. Sumbu Z: Bersifat parallel dengan arah *Conventional Terrestrial Pole* (CTP)
3. Sumbu X: Persimpangan WGS 84 *Reference Meridian Plane* dengan garis khatulistiwa *Conventional Terrestrial Pole* (CTP)
4. Sumbu Y: 90 derajat ke arah timur dari sumbu X

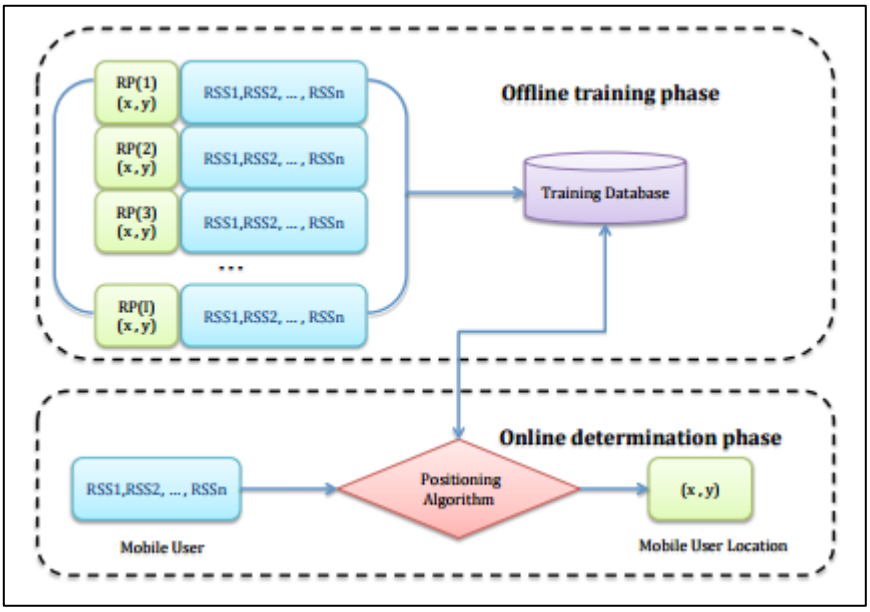
2.4 Fingerprinting

Indoor positioning system berbasis *fingerprinting* bekerja dengan mengambil pemetaan sinyal pada titik-titik tertentu lingkungan dan kemudian menentukan posisi dengan membandingkan hasil pemetaan dengan hasil pengukuran yang didapat saat sistem digunakan (Beder dan Klepal, 2012). Proses *fingerprinting* terdiri atas dua fase, yaitu fase *offline training* dan fase *online determination* (Jiang, 2012).

Dalam penggunaan medan magnet bumi, fase *offline training* digunakan untuk memetakan kekuatan medan magnet bumi pada titik-titik lingkungan yang juga disebut sebagai *grid points* atau *reference points* (Yeung dan Ng, 2006). Hasil pemetaan yang diperoleh akan disimpan pada *database* sebagai *training database*. Nilai yang ditampung ke dalam *database* berupa kekuatan medan magnet bumi pada sumbu X, Y, dan Z yang diperoleh dari sensor magnetometer pada *smartphone*. Dalam fase *online determination*, nilai kekuatan medan magnet bumi yang diperoleh pada lokasi pengguna dibandingkan dengan nilai pada *training database* hasil pemetaan dan dilakukan pencarian hasil yang paling mendekati untuk mendapatkan lokasi dari pengguna (Jiang, 2012).

Perbandingan antara nilai yang diperoleh oleh *device* dengan nilai yang tersimpan pada *training database* menggunakan sebuah algoritma penempatan atau *positioning algorithm*. Beberapa *positioning algorithm* yang sering digunakan antara lain *nearest neighbor* (NN), *k-nearest neighbors* (kNN), Bayesian *probabilistic algorithm*, dan *support vector machine* (Jiang dkk., 2015).

Gambar 2.4 menggambarkan tahapan proses *fingerprinting* untuk penggunaan *Received Signal Strength* (RSS) yang diperoleh dari sinyal Wi-Fi. Proses dan tahapan yang dilakukan sama dengan *fingerprinting* menggunakan medan magnet bumi, perbedaan hanya pada data yang digunakan untuk perbandingan, yakni data RSS dari *access point* Wi-Fi dengan nilai sumbu X, Y, Z medan magnet bumi.



Gambar 2.4 Proses Fingerprinting (Jiang, 2012)

Pada saat pengambilan sampel, posisi perangkat *smartphone* yang optimal adalah pada bagian bawah dada, dibandingkan dengan pundak, pergelangan tangan, kaki, dan punggung yang memiliki angka rata-rata perbedaan nilai magnetik mencapai 20% (Kai dkk., 2012).

2.5 Algoritma k-Nearest Neighbors

Algoritma *k-nearest neighbors* (kNN) adalah sebuah algoritma klasifikasi atau regresi yang bersifat *non-parametic* dan dapat diimplementasikan secara mudah, namun tetap efektif (Guo dkk., 2003). kNN melakukan klasifikasi berdasarkan *similarity measure*, seperti perhitungan jarak. Dalam *indoor positioning system*, kNN merupakan salah satu cara yang paling mudah untuk menentukan lokasi berdasarkan data hasil *fingerprinting*. Ide dasar dari algoritma kNN adalah membandingkan *fingerprint* pada *training database* dengan bacaan sensor dan memilih *k* jumlah *reference point* dengan nilai yang paling mendekati (Honkavirta, 2008).

Hasil dari algoritma kNN bergantung dengan nilai *k* yang digunakan. Terdapat beberapa cara untuk menentukan nilai *k*, namun yang paling sederhana adalah dengan menjalankan algoritma kNN beberapa kali dengan nilai *k* yang bervariasi dan memilih nilai *k* yang menghasilkan hasil yang paling baik (Guo dkk., 2003). Apabila nilai *k* yang digunakan adalah satu, maka setara dengan algoritma *nearest neighbor* (Jiang, 2012).

Penentuan *nearest neighbors* dapat menggunakan perhitungan perbedaan jarak antara *training database* dan nilai bacaan sensor. Euclidean Distance akan digunakan untuk mengukur jarak perbedaan dari bacaan pada fase *online* dengan *training database*. Rumus untuk menghitung Euclidean Distance adalah (Torres-Sospreda dkk., 2016):

$$EuDis(m_1, m_2) = \sqrt{\sum_{i=\{X,Y,Z\}} \left(m_{1,i} - m_{2,i}\right)^2} \qquad \ldots(2.5)$$

dimana m_1 adalah nilai bacaan pada fase *online* $m_1 = [m_{1X}, m_{1Y}, m_{1Z}]$ dan m_2 adalah nilai pada *training database* $m_2 = [m_{2X}, m_{2Y}, m_{2Z}]$ yang telah diperoleh pada fase *offline training* (Torres-Sospreda dkk., 2016). Lokasi dari *device* pengguna akan ditentukan dengan mengambil nilai rata-rata dari koordinat *k neighbors* dengan nilai Euclidean Distance minimum (Jiang, 2012).

2.6 Akurasi

Akurasi atau tingkat *location error* adalah kebutuhan yang paling penting untuk sebuah *positioning system*. Akurasi yang dihitung terbagi menjadi dua, yakni akurasi dalam menentukan posisi pada sebuah lantai atau *plane positioning* dan akurasi dalam menentukan lantai dimana pengguna berada atau *floor estimation* (Liu dan Yang, 2011).

Dalam *plane positioning*, nilai akurasi dapat dihitung dengan mengukur rata-rata *distance error* yang diperoleh saat melakukan pengujian (Adalja, 2013). Semakin rendah nilai *distance error* yang dihasilkan, maka tingkat akurasi dari *indoor positioning system* yang dibuat semakin baik. Namun, umumnya terdapat *tradeoff* antara akurasi dengan karakteristik-karakteristik lain dari sistem yang dikembangkan, sehingga harus ada tingkat prioritas antar karakteristik dalam *indoor positioning system* (Liu dkk., 2007). Akurasi untuk *plane positioning* dapat dihitung menggunakan rumus (Adalja, 2013):

$$\text{Akurasi} = \frac{\sum_{i=1}^{n} ER_i}{n} \qquad \ldots(2.6)$$

dimana ER adalah *distance error* yang didapat dan n adalah jumlah sampel yang digunakan.

Berdasarkan Sun, dkk. (2015), akurasi untuk *floor estimation* atau *floor identification* diuji dengan mengambil sampel pada lokasi lingkungan penelitian secara acak, dan menghitung persentase berapa kali lantai dapat diidentifikasi dengan benar dari banyak sampel yang diambil saat percobaan yang dapat direpresentasikan dalam rumus:

$$\text{Akurasi} = \frac{B}{n} \times 100\% \qquad \ldots(2.7)$$

dimana B adalah jumlah dimana posisi lantai dapat diperkirakan dengan benar dan *n* adalah jumlah sampel yang diambil untuk menguji tingkat akurasi *floor estimation*.

Haversine Distance Formula digunakan untuk menentukan jarak lokasi antara dua titik *latitude* dan *longitude* dengan memperhitungkan lengkungan bumi (Ivis, 2006). Rumus untuk menghitung Harversine Distance adalah (Mwemezi dan Huang, 2011):

$$d = 2r \sin^{-1}\left(\sqrt{\sin^2\left(\frac{lat_2 - lat_1}{2}\right) + \cos(lat_1)\cos(lat_2)\sin^2\left(\frac{lon_2 - lon_1}{2}\right)}\right) \qquad \ldots(2.8)$$

dimana lat_1 dan lat_2 adalah nilai *latitude* titik satu dan dua dalam *radian*, lon_1 dan lon_2 adalah nilai *longitude* titik satu dan dua dalam *radian*, r adalah radius dari bumi sepanjang 6372.8 km.

2.7 Presisi

Akurasi hanya menghitung nilai rata-rata *distance error*. Presisi menghitung tingkat konsistensi dari sistem dengan melihat variasi dari performa melalui beberapa kali percobaan (Adalja, 2013). Beberapa literatur mendefinisikan presisi sebagai standar deviasi dari *distance error* untuk melihat distribusi dari *distance error* antara lokasi yang telah diestimasi atau diprediksi dengan lokasi yang sebenarnya (Adalja, 2013). Nilai standar deviasi dari hasil penelitian dapat dihitung dengan menggunakan rumus (Supranto, 2009):

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \qquad \ldots(2.9)$$

dimana s adalah nilai standar deviasi, n adalah jumlah sampel, x adalah *distance error*, dan $\bar{x}$ adalah rata-rata *distance error* dari seluruh sampel.

2.8 Library Osmdroid

Osmdroid adalah pengganti lengkap untuk *class* Android Mapview untuk API versi satu. *Library* osmdroid menyediakan *modular tile provider* yang mendukung berbagai *tile source* yang bersifat *online* maupun *offline* beserta dukungan fitur-fitur seperti *plotting icons*, *tracking location*, dan menggambar bentuk-bentuk pada *overlay* tampilan peta (spyhunter99, 2017). Osmdroid menyediakan cara yang praktis dan efisien untuk mengubah sebuah aplikasi Android yang didasari oleh Google Maps menjadi OpenStreetMap (MKer, 2016).

OSMBonusPack adalah *library* pihak ketiga yang digunakan untuk menambahkan fitur-fitur pada *library* osmdroid. *Library* OSMBonusPack melengkapi *library* dasar osmdroid dengan fitur-fitur seperti (MKer, 2016):

- *Routes* dan *directions*
- *Points of Interests* (POI)
- *Marker clustering*
- Dukungan untuk *Keyhole Markup Language* (KML) dan GeoJSON
- *Geocoding* dan *reverse geocoding*
- dan lain-lain

2.9 Keyhole Markup Language

Keyhole Markup Language (KML) adalah bahasa pemrograman berbasis XML, yang pada awalnya dikembangkan untuk mengatur tampilan data *geospatial* pada aplikasi Google Earth oleh Google. Pada tahun 2008, KML menjadi standar internasional untuk *Open Geospatial Consortium* (Bacharach, 2008).

KML adalah standar untuk menampilkan *overlay* pada peta, dimulai dari beberapa *marker* sederhana sampai struktur-struktur yang kompleks. Google Maps dan Bing Maps mengizinkan orang-orang untuk membuat *personal maps* dengan membentuk garis, poligon, dan *marker* yang kemudian dapat di-*export* dalam format KML (MKer, 2016).

Beberapa kegunaan untuk KML antara lain (Open Geospatial Consortium, 2014):

- Memberikan label untuk mengidentifikasi lokasi-lokasi pada Bumi
- Mendefinisikan *overlay* untuk dihubungkan ke peta
- Mendefinisikan tampilan-tampilan dari fitur KML
- Menentukan lokasi dan orientasi dari objek-objek 3D seperti garis dan poligon
- Mendeskripsikan objek sesuai dengan fitur KML
- Dan lain-lain