TUGAS AKHIR – TE145561

PERANCANGAN SISTEM NAVIGASI ROV (*REMOTELY OPERATED VEHICLE)* MENGGUNAKAN MODUL KOMPAS *DIGITAL* DAN *ACCELEROMETER*

Gerdina Ika Wahyu
NRP. 10311500000064

Supervisor
Ir. Rusdhianto Effendie AK, MT.
Yunafi'atul Aniroh, ST., MSc.

PROGRAM STUDI KOMPUTER KONTROL
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

-----Halaman ini sengaja dikosongkan-----

TUGAS AKHIR – TE145561

ROV (REMOTELY OPERATED VEHICLE) NAVIGATION SYSTEM DESIGN USING DIGITAL COMPASS MODULE AND ACCELEROMETER

Gerdina Ika Wahyu
NRP. 10311500000064

Supervisor
Ir. Rusdhianto Effendie AK, MT.
Yunafi'atul Aniroh, ST., MSc.

PROGRAM STUDI KOMPUTER KONTROL
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

-----Halaman ini sengaja dikosongkan-----

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul **Perancangan Sistem Navigasi Rov (*Remotely Operated Vehicle*) Menggunakan Modul Kompas *Digital* Dan *Accelerometer*** adalah benar-benar hasil karya intelektual sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya orang lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, 24 Juli 2018

Gerdina Ika Wahyu
NRP. 10311500000064

-----Halaman ini sengaja dikosongkan-----

PERANCANGAN SISTEM NAVIGASI ROV (*REMOTELY OPERATED VEHICLE*) MENGGUNAKAN MODUL KOMPAS *DIGITAL* DAN *ACCELEROMETER*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Ahli Madya Teknik
Pada
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Ruschirianto Effendy A.K., MT.	Yunafi'atul Aniroh, S.T.M.Sc.
NIP. 196704241985021001	NIP. 2200201405001

SURABAYA
JULI, 2018

-----Halaman ini sengaja dikosongkan-----

PERANCANGAN SISTEM NAVIGASI ROV (*REMOTELY OPERATED VEHICLE*) MENGGUNAKAN MODUL KOMPAS *DIGITAL* DAN *ACCELEROMETER*

Nama : Gerdina Ika Wahyu
Pembimbing : Ir. Rusdhianto Effendy, A.K., M.T.
Yunafi'atul Aniroh, S.T.,M.Sc.

ABSTRAK

ROV (*Remotely Operated Vehicles*) merupakan salah satu wahana yang beroperasi di bawah laut dan dikendalikan oleh seorang operator dari darat. Pada pengoperasiaanya, ROV membutuhkan sebuah sistem navigasi guna mengetahui posisi dari ROV tersebut. Posisi ROV ini nantinya akan digunakan oleh operator untuk mengendalikan gerak ROV menuju tempat yang diinginkan.

Pada tugas akhir ini dilakukan pembuatan sistem navigasi menggunakan modul kompas *digital* dan *accelerometer*. Modul kompas *digital* bertujuan untuk menampilkan arah *heading* ROV, sedangkan sensor *accelerometer* untuk mengetahui jarak yang ditempuh ROV. Data ini diperoleh dari hasil integral ganda terhadap keluaran *accelerometer* yang berupa percepatan. Sistem navigasi ini terdiri dari modul kompas *digital* GY271, sensor *accelerometer* MPU6050, dan Arduino Mega 2560.

Berdasarkan data pembacaan sensor kompas, rata-rata error paling besar didapat pada arah Barat Daya dengan 3,81% dan paling kecil pada arah Barat Laut dengan 0,23%. Nilai *error* yang dihasilkan pada pembacaan secara *real time* sangat besar sehingga pengaplikasiannya pada ROV belum dapat dilakukan. Rata-rata nilai *error* mencapai 99,16% dalam waktu 100 detik pada sumbu X dan 51% pada sumbu Y. Untuk penelitian selanjutnya disarankan untuk menggunakan sensor *accelerometer* yang sudah terintegrasi dengan GPS dengan nilai kepresisian lebih tinggi.

Kata Kunci : *Accelerometer*, Kompas, ROV

-----Halaman ini sengaja dikosongkan-----

ROV (REMOTELY OPERATED VEHICLE) NAVIGATION SISTEM DESIGN USING DIGITAL COMPASS MODULE AND ACCELEROMETER

Name : Gerdina Ika Wahyu
Supervisor : Ir. Rusdhianto Effendy, A.K., M.T.
Yunafi'atul Aniroh, S.T.,M.Sc.

ABSTRACT

ROV (Remotely Operated Vehicles) is one of the robot that operate under the sea and controlled by an operator from the mainland. In operation, ROV required a navigation system to obtain the position of the ROV. This ROV position would be used by the operator to control the movement of ROV to the desired place.

In this final project, it was designed a navigation sistem using digital compass module and accelerometer. Digital compass module was aimed to display the direction of heading from ROV, while the accelerometer was used to know the distance traveled by ROV. Position data was obtained from the result of double integration to the accelerometer output in the form of acceleration. The navigation system consisted of GY271 digital compass module, MPU6050 accelerometer sensor, and Arduino Mega 2560.

Based on compass record data, the largest average error was occured when the sensor was directed to Southwest with 3.81% and the smallest is the Northwest with 0.23% error. The error value of accelerometer in real time data was so high, so it can not be used as a main navigation yet. The average error rate reaches 99.16% within 100 seconds on the X axis and 51% on the Y axis. For further research, it will be better to use accelerometer sensor that has been integrated with GPS with a higher precision value.

Keywords: *Accelerometer, Compass, ROV*

-----Halaman ini sengaja dikosongkan-----

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang selalu memberikan rahmat dan hidayah-Nya sehingga Tugas Akhir ini dapat terselesaikan dengan baik. Shalawat serta salam semoga selalu dilimpahkan kepada Rasulullah Muhammad SAW, keluarga, sahabat, dan umat muslim yang senantiasa meneladani beliau.

Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan guna menyelesaikan pendidikan Diploma 3 pada Bidang Studi Komputer Kontrol, Departemen Teknik Elektro Otomasi, Fakultas Vokasi, Institut Teknologi Sepuluh Nopember Surabaya dengan judul:

PERANCANGAN SISTEM NAVIGASI ROV (*REMOTELY OPERATED VEHICLE*) MENGGUNAKAN MODUL KOMPAS *DIGITAL*, DAN *ACCELEROMETER*

Dalam Tugas Akhir ini dirancang sistem navigasi yang diperuntukkan untuk ROV (*Remotely Operated Vehicle*) agar dapat terlacak keberadaannya didalam air.

Penulis mengucapkan terima kasih kepada Ibu dan Bapak penulis yang memberikan berbagai bentuk doa serta dukungan tulus tiada henti, Bapak Ir. Rusdhianto Effendy A.K., M.T. dan Ibu Yunafi'atul Aniroh, S.T. M.Sc. atas segala bimbingan ilmu, moral, dan spiritual dari awal hingga terselesaikannya Tugas Akhir ini, Penulis juga mengucapkan banyak terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung dalam proses penyelesaian Tugas Akhir ini.

Penulis menyadari dan memohon maaf atas segala kekurangan pada Tugas Akhir ini. Akhir kata, semoga Tugas Akhir ini dapat bermanfaat dalam pengembangan keilmuan di kemudian hari.

Surabaya, 26 Juli 2018

Penulis

-----Halaman ini sengaja dikosongkan-----

DAFTAR ISI

DAFTAR GAMBAR

-----Halaman ini sengaja dikosongkan-----

DAFTAR TABEL

-----Halaman ini sengaja dikosongkan-----

BAB I
PENDAHULUAN

1.1 Latar Belakang

Remotely Operated Vehicle (ROV) adalah robot bawah air yang dioperasikan oleh seseorang di atas kapal melalui kabel yang membawa sinyal elektrik secara bolak balik antara operator dan wahana ini. ROV digunakan untuk membantu penyelam atau memperluas kemampuan manusia untuk menjangkau laut dalam dimana penyelam sulit bekerja secara aman dan efektif. Biasanya ROV digunakan untuk melakukan beberapa pekerjaan seperti inspeksi, manipulasi, instalasi dan pemeliharaan peralatan bawah air dan survei dasar laut seperti survei karang.

Dalam pegoperasiannya ROV memiliki banyak tantangan terutama pada sistem navigasinya. Karena pengoperasiannya yang berada dibawah laut, diperlukan sensor yang cocok untuk dapat mengirim data berupa lokasi ROV menembus permukaan air. Sistem navigasi ini sendiri digunakan agar pergerakan dari ROV tetap terpantau walaupun terkena arus laut, dan ROV dapat bergerak sesuai dengan kendali dari pengguna. Sistem navigasi yang biasa digunakan masih memanfaatkan sonar, dan GPS yang tergolong cukup mahal.

Pada tugas akhir ini topik yang dipilih adalah perancangan suatu sistem navigasi guna memonitoring pergerakan dan posisi ROV. Pembuatan sstem navigasi ini memanfaatkan modul kompas *digital* dan *accelerometer* dengan kontroler Arduino 2560. Modul kompas *digital* bertujuan untuk menampilkan arah *heading* dari ROV, sedangkan *accelerometer* untuk menampilkan posisi dari ROV. Data posisi diperoleh dari hasil proses integral ganda terhadap keluaran *accelerometer* yang berupa percepatan (grativasi). Tujuan dari penelitian ini diharapkan modul kompas dan *accelerometer* dapat menampilkan arah *heading* dan data posisi ROV berupa jarak, sehingga pengguna dapat melakukan *monitoring* terhadap ROV yang digunakan

serta menggerakkan ROV sesuai dengan titik yang dihendaki dengan menggunakan *joystick.*

1.2 Tujuan dan Manfaat

Tujuan yang akan dicapai dari tugas akhir ini adalah untuk mendapat rancangan sistem navigasi mandiri pada ROV sehingga pengguna mengetahui koordinat awal dari ROV dan pergerakan ROV. Adapun informasi yang diberikan berupa arah *heading* beserta jarak yang ditempuh oleh ROV sehingga mempermudah *user* dalam melakukan pemantauan dan mengontrol arah gerak ROV.

1.3 Permasalahan

Dalam melakukan pemantauan, ROV harus menyelam ke dalam air hingga kedalaman tertentu. Untuk dapat memonitoring gerak dan arah *heading* ROV dibutuhkan sebuah sistem navigasi. Untuk itu penulis memilih sebuah sensor *accelerometer* dan kompas *digital* agar dapat menampilkan data yang diharapkan tadi. Untuk mendapat data posisi dilakukan pengintegralan dua kali dari data percepatan pada *accelerometer*. Akan tetapi data percepatan sendiri memiliki *noise* yang cukup besar, dan apabila tidak direduksi *noise* tadi akan berkali lipat besarnya ketika diintegralkan. Untuk itu dibutuhkan filter yang sesuai agar nantinya *error* dari data (posisi) mendekati 0.

1.4 Batasan Masalah

Dalam pembuatan alat pada tugas akhir ini batasan masalah ada pada:

a. Menampilkan posisi berupa data jarak dan arah *heading* ROV.
b. Pergerakan ROV hanya pada kondisi linier yaitu sumbu X dan Y.
c. Nilai *offset* akibat *pitch* dan *roll* dihiraukan.
d. Titik awal dan akhir dari pergerakan sudah ditentukan terlebih dahulu.

1.5 Metodologi Penelitian

Dalam pelaksanaan tugas akhir yang berupa Perancangan Sistem Navigasi ROV Menggunakan Modul Kompas *Digital* dan

Accelerometer, ada beberapa kegiatan yang dapat diuraikan sebagai berikut:

a. Tahap persiapan

Pada tahap ini akan dilakukan studi literatur mengenai :

1. Mempelajari karakteristik sensor akselerometer dan kompas.

   Sensor akselerometer dan kompas digunakan untuk menampilkan data berupa percepatan dan arah mata angin.

2. Mempelajari konsep integral ganda

   Nilai percepatan dari data akselerasi (percepatan) akan diintegralkan sebanyak dua kali untuk mendapatkan data berupa posisi atau jarak menggunakan hukum trapezoidal.

3. Mempelajari Metode *Filter* Kalman

Kalman *filter* merupakan jenis *filter* yang digunakan untuk mengestimasi posisi ROV dan juga untuk menghilangkan nilai *error* pada data.

b. Tahap identifikasi dan pemodelan sistem

Pada tahap ini akan dilakukan identifikasi dari sistem alat sesuai data yang telah didapatkan dari studi literatur serta dilakukan pemodelan dari alat yang akan dikerjakan.

c. Tahap perancangan

Pada tahap ini akan dilakukan perancangan dari sistem sesuai data yang telah didapatkan dari studi literatur. Dimulai dari perancangan program Arduino berupa pengambilan data kasar dari masing – masing sensor. Kemudian data tadi akan disaring menggunakan Kalman *filter*.

d. Tahap pembuatan alat

Pada tahap ini akan dilakukan pembuatan alat sesuai perancangan yang dibuat, berdasarkan data yang telah dikumpulkan melalui studi literatur.

e. Tahap pengujian dan Analisa

Pada tahap ini akan dilakukan pengujian alat, menganalisa kesalahan atau kegagalan pada alat dan mengatasi permasalahan tersebut. Pada tahap ini juga dilakukan analisa faktor penyebab alat tidak bekerja sesuai dengan keinginan atau terjadi *error*.

Tahapan ini dilakukan berdasarkan urutan di bawah ini:

- Pengujian program pembacaan sensor dari Arduino
- Pengujian Kalman *filter* pada nilai *error* pada sensor

f. Tahap penyusunan laporan

Setelah alat berhasil dibuat dan bekerja dengan baik tanpa adanya *error*, pengambilan data dan analisa data terpenuhi, maka tahap selanjutnya yaitu penyusunan laporan untuk buku tugas akhir. Diharapkan buku tugas akhir ini bermanfaat bagi semua orang dan dapat dijadikan pedoman dalam melanjutkan dan mengembangkan ide tugas akhir ini.

1.6 Sistematika Laporan

Untuk pembahasan lebih lanjut, laporan tugas akhir ini disusun dengan sistematika sebagai berikut:

Bab I PENDAHULUAN

Membahas tentang latar belakang, perumusan masalah, batasan masalah, maksud dan tujuan, sistematika laporan, metodologi, serta relevansi tugas akhir yang dibuat.

Bab II TEORI DASAR

Menjelaskan teori yang berisi teori-teori dasar yang dijadikan landasan dan mendukung dalam perencanaan dan pembuatan alat yang dibuat.

Bab III PERANCANGAN ALAT

Membahas perencanaan dan pembuatan tentang perencanaan dan perancangan s*oftware* yang meliputi program yang akan digunakan untuk menjalankan alat tersebut.

Bab IV PENGUKURAN DAN ANALISA

Membahas pengujian alat dan menganalisa data yang didapat dari pengujian tersebut serta membahas tentang pengukuran, pengujian, dan penganalisaan terhadap alat.

Bab V PENUTUP

Berisi penutup yang menjelaskan tentang kesimpulan yang didapat dari tugas akhir ini dan saran-saran untuk pengembangan alat ini lebih lanjut.

-----Halaman ini sengaja dikosongkan-----

BAB II
TEORI PENUNJANG

2.1 Tinjauan Pustaka

Sejauh ini sudah banyak dilakukan penelitian mengenai sistem navigasi dengan memanfaatkan sebuah unit pengukuran IMU (*Inertial Measurement Unit*) dimana pada pengukuran jenis ini memanfaatkan 2 buah sensor *accelerometer* dan *gyroscope*. Dan pada kasus lain juga memanfaatkan magnetometer.

Pada papernya yang berjudul Pendeteksi Posisi Menggunakan Sensor *Accelerometer* MMA7260Q Berbasis Mikrokontroler Atmega 32, Muhammad Riyadi menggunakan sensor *accelerometer* MMA7260Q untuk mendapatkan data berupa percepatan gravitasi dan . Untuk mendapatkan mengurangi *error* pada data beliau menggunakan beberapa *filter* seperti *filter* eksponensial dan *filter* Kalman. Berbeda dengan alat milik Muhammad Riyadi penulis yang menggunakan sensor MPU6050 untuk sensor *accelerometer*nya dengan *filter Low pass Digital* dan *filter* Kalman. [1]

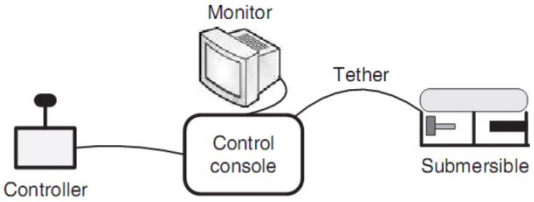
Hafizhuddin Zul Fahmi dan kawan – kawan pada papernya menulis tentang Implementasi *Complementary Filter* Menggunakan Sensor *Accelerometer* dan Gyroscope pada Keseimbangan Gerak Robot Humanoid. Disini dia dan teman – teman memanfaatkan sensor MPU6050 untuk kemudian diolah datanya agar mendapatkan nilai berupa posisi sudut. Selain itu dia menggunakan *complementary* dan Kalman *filter* sebagai *filter error* pada datanya. Pada bahasan tugas akhir ini penulis menggunakan sensor dan *filter* dan tujuan yang berbeda yaitu data posisi linier. [2]

2.2 ROV

ROV adalah wahana bawah air yang bertenaga listrik dan dikontrol melalui pusat, dapat bermanuver sesuai perintah manusia dengan pendorong (*thruster*) hidrolik atau elektrik. [3]

Definisi lain disampaikan oleh Christ dan Wernli dimana ROV adalah kamera yang dipasang dalam wadah tahan air, dengan pendorong untuk bermanuver, yang melekat pada kabel ke permukaan dimana sinyal video yang dikirim. Sebuah ROV menerima energi dan informasi perubahan dengan panel kontrol yang terletak di permukaan melalui kabel pusat. Dari panel kontrol, operator dapat merencanakan pekerjaan

atau menggunakan satu *joystick* untuk manuver wahana secara langsung Gambar 2.1. [3]



Gambar 2. 1 Komponen dasar sistem ROV

2.3 *Accelerometer*

Accelerometer adalah alat yang digunakan untuk mengukur percepatan, mendeteksi dan mengukur getaran (vibrasi), dan mengukur percepatan akibat gravitasi (inklinasi). *Accelerometer* dapat digunakan untuk mengukur getaran pada mobil, mesin, bangunan, dan instalasi pengamanan. *Accelerometer* juga dapat diaplikasikan pada pengukuran aktivitas gempa bumi dan peralatan-peralatan elektronik, seperti permainan 3 dimensi, mouse komputer, dan telepon. Untuk aplikasi yang lebih lanjut, sensor ini banyak digunakan untuk keperluan navigasi.

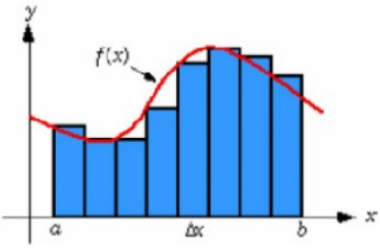
Percepatan merupakan suatu keadaan berubahnya kecepatan terhadap waktu. Bertambahnya suatu kecepatan dalam suatu rentang waktu disebut percepatan (*acceleration*). Namun jika kecepatan semakin berkurang daripada kecepatan sebelumnya, disebut perlambatan (*deceleration*). Percepatan juga bergantung pada arah/orientasi karena merupakan penurunan kecepatan yang merupakan besaran vektor. Berubahnya arah pergerakan suatu benda akan menimbulkan percepatan pula. Untuk memperoleh data jarak dari sensor *accelerometer*, diperlukan proses integral ganda terhadap keluaran sensor. [4]

$$\bar{s} = \int (\int (\bar{a})\, dt)\, dt \tag{2.1}$$

$\bar{s}$ = jarak
$\bar{a}$ = percepatan
dt = jeda waktu cuplik data

Proses penghitungan ini dipengaruhi oleh waktu cuplik data, sehingga jeda waktu cuplik data (dt) harus selalu konstan dan dibuat sekecil. [5]

Secara sederhana, integral merupakan luas daerah di bawah suatu sinyal selama rentang waktu tertentu. Untuk lebih jelasnya dapat dilihat pada Gambar 2.2. [5]



Gambar 2. 2 Proses integral terhadap suatu sinyal

Berikut adalah persamaan yang digunakan:

$$\int_a^b f(x)dx = \lim_{x=\infty} \sum_{i=1}^{x} f(x_t)\Delta x \tag{2.2}$$

$$\Delta x = \frac{b-a}{n} \tag{2.3}$$

f(x) = fungsi f dari variabel x
a = inisial posisi
b = posisi final
Δx = panjang kurva
n = banyaknya luasan persegi
dx = diferensial variable x
x_t = x terhadap waktu

Persamaan pengintegralan pada persamaan (2.2) masih memiliki *error* yang cukup besar. Untuk lebih mengoptimalkan hasil pengintegralan maka dapat digunakan metode trapezoidal seperti pada persamaan (2.4).[5]

$$x_k = x_{k-1} + \frac{h}{2}[f(x_k, t_k) + (f(x_{k-1}, t_{k-1})] \tag{2.4}$$

x_k = x terhadap k
x_{k-1} = x terhadap k-1
t_k = waktu terhadap k
t_{k-1} = waktu terhadap k-1

2.4 Kalman *Filter*

Kalman *filter* merupakan salah satu solusi optimal dalam menyaring data dari sinyal pada suatu proses yang linier. Kalman *filter* digunakan pada proses yang dapat dinyatakan dalam bentuk persamaan state linier. Kalman *filter* digunakan pada proses yang dapat dinyatakan dalam bentuk persamaan *state* linier seperti pada pada persamaan (2.5).

$$x_{k+1} = A_k X_k + Bu_k + w_k \tag{2.5}$$

$$Z_k = H_k X_k + V_k \tag{2.6}$$

$X_{(k+1)}$: vektor *state space* dari proses pada saat $k + 1$
A_k : matriks transformasi *state space* saat k
Z_k : vektor output dari proses yang diukur pada saat $k + 1$
H_k : matriks hubungan *state* – output
Bu_k : *determind* input
w_k : *noise* proses pada saat k
V_k : *noise* pengukuran pada saat k

Persamaan (2.5) dapat diobservasi dengan model pengukuran yang memetakan *state* x ke keluaran z seperti dituliskan pada persamaan (6). *Noise* proses (w) dan *noise* pengukuran (v) merupakan *noise* yang saling bebas. Nilai estimasi state x_k pada Kalman *filter* ditentukan dari estimasi posteriori x_k serta selisih antara pengukuran sebenarnya z_k dan estimasi pengukuran $H_k X_k$.

$$x_k = x_k^- + K_k(H_k x_k + V_k - H_k x_k^-) \tag{2.7}$$

x_k = vektor *state space* dari proses pada saat k
x_k^- = vektor pra *state space* dari proses pada saat k
K_k = *Gain* Kalman

Selisih nilai antara pengukuran sebenarnya z_k dan estimasi pengukuran disebut sebagai residual atau pengukuran *innovation*. Jika nilai residual adalah nol, maka hal itu menunjukkan bahwa hasil estimasi sama dengan hasil pengukuran. Nilai K_k adalah faktor *gain* pada Kalman filter.

$$P_k = (I - K_k H_k) P_k^- \quad (2.8)$$

P_k = Kovarian estimasi *error* Kalman
P_k^- = Kovarian pra-estimasi *error* Kalman

Pada Kalman *filter* dipilih nilai K_k sehingga estimasi posteriori adalah optimal atau mempunyai *error* yang minimum. Nilai P_k minimum diperoleh jika nilai K_k dapat menyediakan estimasi yang mempunyai *covariance* minimum.

Penyelesaian untuk mendapatkan P_k minimum ditunjukkan pada persamaan dibawah. [6]

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \quad (2.9)$$

R_k = *Matriks* kovarian R

Nilai *covariance* dari *error* diberikan pada persamaan (2.10).

$$P_{k+1}^- = A_k P_k A_k^T + Q_k \quad (2.10)$$

P_{k+1}^- = Kovarian pra-estimasi *error* Kalman pada saat k+1
Q_k = *Matriks* kovarian Q

2.5 Arduino

Arduino Mega 2560 adalah sebuah papan mikrokontroler berbasis Atmega 2560. Arduino ini mempunyai 54 pin *digital* input/output (I/O) dengan 14 pin yang dapat digunakan sebagai keluaran PWM, 16 pin input analog, 2 UARTs *(Hardware serial ports*), sebuah *crystal oscillator* 16 MHz, sebuah penghubung USB, sebuah colokan listrik, ICSP header, dan 1 tombol *reset*.

Setiap isi dari Arduino Mega 2560 membutuhkan dukungan mikrokontroler; koneksi mudah antara Arduino mega 2560 ke komputer adalah dengan menggunakan sebuah kabel USB atau daya AC to DC adaptor. Tegangan operasinya sebesar 5 V, dengan tegangan input sebesar 6 – 20 V.

Arus DC pin I/O sebesar 40 mA sedangkan arus DC untuk pin 3.3V sebesar 50 mA. Selain itu Arduino ini mempunyai SRAM 8 *Kbyte*, EEPROM 4 *Kbyte* dan *flash memory* sebesar 156 Kb yang mana 8

Kbnya digunakan oleh *bootloader*. Bentuk fisik dari Arduino Mega 2560 ditunjukkan pada Gambar 2.3. [7]

Gambar 2. 3 Bentuk fisik Arduino Mega 2560

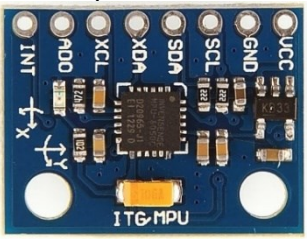
2.6 MPU6050

GY-521 MPU-6050 *Module* adalah sebuah modul berinti MPU-6050 yang merupakan 6 axis *Motion Processing Unit* dengan penambahan regulator tegangan dan beberapa komponen pelengkap lainnya yang membuat modul ini siap dipakai dengan tegangan *supply* sebesar 3-5 VDC. Modul ini memiliki *interface* I2C yang dapat disambungkan langsung ke MCU yang memiliki fasilitas I2C.

Sensor MPU-6050 berisi sebuah MEMS *Accelerometer* dan sebuah MEMS Gyro yang saling terintegrasi. Sensor ini sangat akurat dengan fasilitas *hardware* internal 16 bit ADC untuk setiap kanalnya. Sensor ini akan menangkap nilai kanal *axis* X, Y dan Z bersamaan dalam satuan waktu.

Sensor ini berbasis c*hip* MPU-6050 dengan *range* Gyroscope + 250 500 1000 2000 ° / s dan *range* Akselerasinya ± 2 ± 4 ± 8 ± 16 g. Jarak antar pin headernya 2.54 mm dan dimensi modulnya 20.3mm x 15.6mm. [8]

Gambar 2.4 adalah tampilan fisik dari sensor MPU6050.



Gambar 2. 4 Bentuk fisik sensor MPU6050

2.7 GY271

GY271 merupakan sensor kompas dengan 3 sumbu dimana didalamnya ditanamkan sebuah *chip* magnetometer yang bisa berupa HMC5883L dan QMC5883L. Perbedaan dari kedua *chip* terdapat pada *address* I2C yang digunakan. Pada tugas akhir kali ini penulis menggunakan GY271 dengan *chip* QMC5883L didalamnya.

QMC5883L adalah sensor magnetik tiga-sumbu multi-*chip* yang terintegrasi dengan kondisi sinyal ASIC, yang ditargetkan untuk aplikasi presisi tinggi seperti kompas, navigasi dan game di drone, robot, ponsel dan perangkat genggam pribadi. QMC5883L ini berlisensi dari teknologi Honeywell AMR. Seiring dengan 16-bit ADC ASIC yang dirancang khusus, ia menawarkan keuntungan dengan *noise* yang rendah, akurasi 1 ° hingga 2 °, tegangan operasi 2.16 V sampai 3.6 V dan konsumsi daya rendah (75μA), serta *interface* menggunakan I2C. Gambar fisik dari sensor GY271 adalah seperti pada Gambar 2.5: [9]

Gambar 2. 5 Bentuk fisik GY271

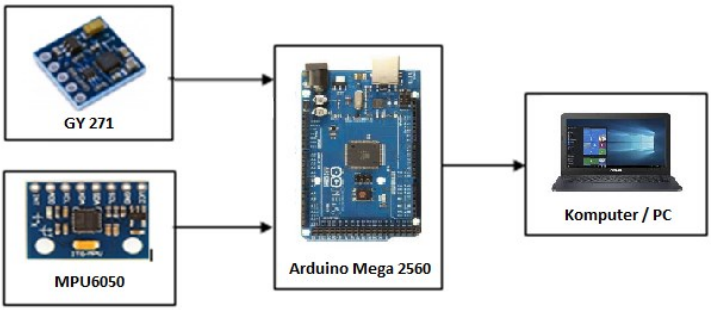
-----Halaman ini sengaja dikosongkan-----

BAB III
PERANCANGAN DAN IMPLEMENTASI

Pada Bab ini akan dijelaskan mengenai Perancangan Sistem Navigasi ROV (*Remotely Operated Vehicle*) Menggunakan Modul Kompas *Digital* dan *Accelerometer* meliputi blok fungsional sistem yang akan menjelaskan proses kerja alat dalam bentuk alur diagram maupun perancangan software.

3.1 Diagram Fungsional Sistem

Terdapat 3 komponen penting pada sistem navigasi ini yaitu sensor *accelerometer* MPU6050, sensor kompas *digital* GY271 dan Arduino Mega 2560 sebagai kontrolernya. Gambar 3.1 merupakan tampilan blok fungsional sistem pada tugas akhir ini.



Gambar 3. 1 Diagram fungsional sistem

Dari Gambar 3.1, dijelaskan bahwa dalam tugas akhir ini, dibuat sebuah sistem navigasi menggunakan sensor *accelerometer* dan kompas *digital*. Data keluaran dari kedua sensor utama diambil menggunakan Arduino Mega 2560 dan nantinya akan difilter menggunakan *digital low pass filter* dan Kalman *filter*.

Data *accelerometer* disaring terlebih dahulu menggunakan *digital low pass filter* yang terdapat didalam sensor MPU6050. Sensor MPU6050 ini digunakan untuk menghilangkan *noise* dari pembacaan sensor yang cukup besar. Setelah difilter *raw* data tadi difilter kembali menggunakan Kalman *filter* untuk menghilangkan nilai *error* pembacaan dan *error* akibat pengintegralan ganda. Setelah dilakukan

pemfilteran dengan Kalman *filter* dilakukan *allignment* untuk membuat keluaran daripada sensor mendekati nilai *real*.

Berbeda dengan sensor MPU6050, pada sensor GY 271 tidak mengalami tahapan penfilteran. Pengurangan nilai *error* pada sensor dilakukan dengan menambahkan *error* sudut deklinasi pada pembacaan data. Arduino mega digunakan sebagai memproses data untuk kemudian melakukan komunikasi secara serial antara sensor dan komputer.

3.2 Perancangan Perangkat Elektrik

Pada sub bab ini akan dibahas tentang *wiring* dari *hardware* yang digunakan. Secara umum perancangan perangkat keras dari sistem navigasi dengan sensor *accelerometer* dan kompas *digital* ini meliputi *wiring* sensor MPU6050 dan GY271 dengan Arduino Mega 2560 .

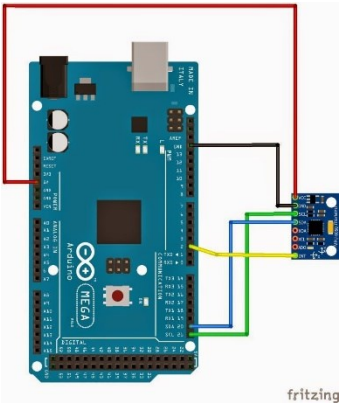
3.2.1 *Wiring* pada Sensor MPU6050 dan Arduino

Sensor MPU6050 berfungsi sebagai pemberi data berupa data percepatan. Sensor ini memiliki 8 pin, berikut adalah pin yang terdapat pada sensor MPU6050:

1. Pin GND Arduino dihubungkan dengan pin GND pada sensor.
2. Pin 5V dihubungkan dengan pin VCC pada sensor.
3. Pin SCA dihubungkan dengan pin SCA pada sensor.
4. Pin SCL Arduino dihubungkan dengan pin SCL pada sensor.
5. Pin 12 pada Arduino dihubungkan dengan pin INT pada sensor.
6. Pin XDA, XCL dan AD0 dibiarkan tidak terhubung.

Bentuk fisik dan *wiring* MPU6050 dengan Arduino ditunjukkan pada Gambar 3.2 dan Gambar 3.3.

Gambar 3. 2 Pin pada sensor MPU6050



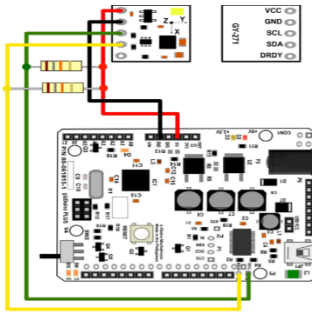
Gambar 3. 3 *Wiring* pada sensor MPU6050

3.2.2 *Wiring* pada Sensor GY271 dan Arduino

Sensor GY271 berfungsi untuk memberikan data berupa arah mata angin untuk menunjukkan arah heading ROV. Berikut ini adalah *wiring* sensor GY271 dengan Arduino Mega 2560 :

1. Pin VCC dihubungkan dengan pin VCC pada sensor.
2. Pin GND dihubungkan dengan pin *ground.*
3. Pin SCA dihubungkan dengan pin SCA pada sensor.
4. Pin SCL dihubungkan dengan pin SCL pada sensor.
5. Pin DRDY dibiarkan tidak terhubung

Gambar 3.4 merupakan Gambar fisik *wiring* Arduino dan sensor GY-271.



Gambar 3. 4 *Wiring* pada sensor GY271

3.3 Perancangan Program

Perancangan program pada Bab 3 ini dibagi menjadi 2 bagian yaitu pemrograman sensor *accelerometer* dan sensor kompas *digital*. Perancangan program dibuat terpisah untuk memudahkan penulis dalam merancang kedua buah sensor.

Secara garis besar proses pertama yang dilakukan adalah inisialisasi dan pembacaan data dari kedua sensor. Perhitungan data dimulai dengan mendapat nilai *raw* tiga sumbu yang ada pada sensor. Selanjutnya data difilter dengan *digital low pass filter* dan Kalman *filter* untuk sensor *accelerometer* dan penambahan *offset* sudut deklinasi pada sensor kompas *digital*. Berikut adalah jabaran dari perancangan kedua sensor tadi.

3.3.1 Perancangan Program Sensor *Accelerometer*

Sensor pertama yang digunakan ialah sensor *accelerometer*. Untuk mencapai tujuannya yaitu menampilkan data jarak, terdapat beberapa tahapan yang harus dilakukan. Berikut adalah tahapan – tahapan tersebut.

3.3.2 Menampilkan *Raw* Data *Accelerometer*

Untuk mendapatkan *raw* data pada sensor *accelerometer*, pertama dilakukan *setup* yang difungsikan agar sensor dapat berkomunikasi dengan arduino. Beberapa register pada sensor harus terlebih dahulu diaktifkan guna dapat melakukan komunikasi data tersebut. Salah satu register yang digunakan adalah register 1C yang berguna untuk memilih skala pembacaan yang dipilih pada sensor.

Register dari sensor dapat ditulis menggunakan heksa atau biner. Terdapat 4 skala pembacaan yang dapat dipilih pada register 1C, 4 skala tadi terdiri dari +/- 2g, +/-4g, +/-8g, +/-16g. Pemilihan skala sensor nantinya mempengaruhi nilai sentivitas pada data *accelerometer*. Semakin besar skala yang digunakan semakin presisi data yang didapatkan. Berikut adalah contoh *coding* pemilihan skala *range* +/- 2g pada sensor.

```
Wire.beginTransmission(0b1101000); //komunikasi I2C
Wire.write(0x1C); //Register Accelerometer
Wire.write(0b00000000); // pemilihan skala +/- 2g
```

Tabel 3.1 menampilkan nilai skala beserta sensitivitas yang dimiliki sensor *accelerometer* MPU6050.

Tabel 3. 1 Skala dan sensitivitas pada sensor *accelerometer*

AFS_SEL	*Full Scale Range*	LSB *Sensitivity*
0	$\pm 2g$	16384 LSB/g
1	$\pm 4g$	8192 LSB/g
2	$\pm 8g$	4096 LSB/g
3	$\pm 16g$	2048 LSB/g

Sebagai upaya pengurangan *noise*, MPU6050 menyediakan *digital low pass filter* pada chip didalamnya. Seperti sensivitas pada sensor, range dari *digital low pass filter* juga sudah disediakan oleh sensor. Untuk dapat mengaktifkannya maka address dari *digital low pass filter* ini harus terlebih dahulu dipanggil. [8]

Untuk mendapat data percepatan dalam satuan G, maka nilai pembacaan terlebih dahulu dibagi dengan sensititas dan kemudian dikali dengan gravitasi. Persamaan (3.1) merupakan rumus yang digunakan untuk mendapat nilai percepatan dari *accelerometer*.

$$A = \frac{Lsb}{Sensitivitas\ skala\ yang\ digunakan} * g \qquad (3.1)$$

A = Nilai akselerasi

Lsb= data berbentuk lsb

g = gravitasi

Sebagai contoh, semisal nilai pembacaan pada serial monitor Arduino IDE menampilkan nilai 16384 dengan skala pembacaan +/-2g pada sumbu x. Maka untuk mendapat nilai percepatan pada sumbu X tadi nilai percepatan sama dengan 16382/16284x9.8.

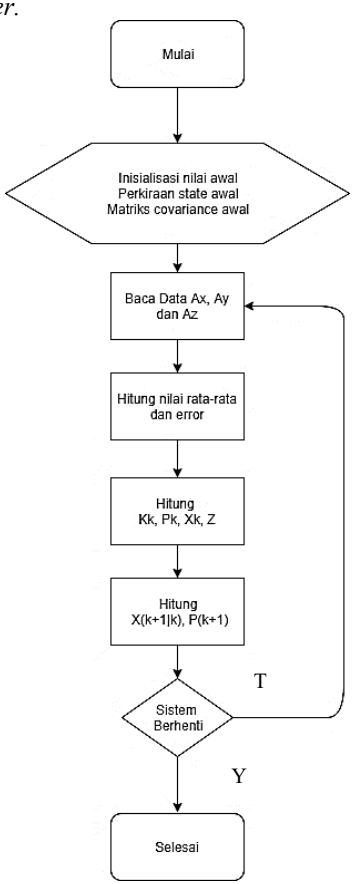
Setelah mendapat nilai dari pembacaan sensor penulis, melakukan pengambilan 5 data dari tiap pembacaannya untuk kemudian diambil rata-ratanya. Data disini diberi istilah x1, x2, x3, x4 dan x5 untuk mempermudah dalam pemrograman sensor. Kelima data tadi lalu dibagi dengan 5. Dibawah ini adalah persamaan guna mendapat nilai rata-rata dari data yang diinginkan :

$$\text{Axr} = \frac{jumlah\ nilai\ x}{banyak\ data\ x} \quad (3.2)$$

Axr = Rata-Rata

3.3.3 Parameter Kalman *Filter*

Parameter Kalman *filter* dibutuhkan untuk mengestimasi output sensor yang berupa *noise*. Gambar 3.4 adalah *flowchart* untuk mendapat parameter Kalman *filter.*



Gambar 3. 4 *Flowchart* parameter Kalman *filter*

Karena *filter* yang digunakan salah satunya adalah Kalman *filter* berbasis *variable state*, maka dibutuhkan pencarian model sistem dan model pengukurannya. Seperti yang tertera pada persamaan (2.5) dan (2.6) kita dapat menggunakan persamaan tersebut dimana persamaan tadi tertulis:

a. Model Sistem
$$X(k+1) = A_k X(k) + B_u(k) + W(k)$$
b. Model Pengukuran
$$Z(k+1) = H_{k+1} X(k+1) + V(k+1)$$

Dibawah ini adalah penjabaran nilai $X(k+1)$ dan $Z(k+1)$ yang didapat dengan menggunakan persamaan diatas :

$$\begin{bmatrix} \dot{x} \\ \dot{u} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a_x(k) \tag{3.3}$$

Karena pada tugas akhir ini menggunakan dua kali pengintegralan terhadap data akselerasi, sehingga terdapat dua derajat kebebasan pada kasus kali ini yaitu x dan v.

$$\dot{x} = u$$
$$\dot{u} = \overline{a_x}$$

$\dot{x}$ = nilai *derivative* dari posisi
$\dot{u}$ = nilai *derivative* dari kecepatan
($\overline{a_x}$) = percepatan pada sumbu X

Nilai *derivative* dari posisi ($\dot{x}$) adalah kecepatan ($\dot{u}$) dan nilai *derivative* dari kecepatan ($\dot{u}$) adalah percepatan ($\overline{a_x}$). Dibawah ini adalah persamaan integral untuk kedua data diatas

$$\dot{x} = \frac{x(k+1) - x(k)}{Ts} \tag{3.4}$$
$$\dot{u} = \frac{u(k+1) - u(k)}{Ts} \tag{3.5}$$

Dan apabila nilai $\dot{x}$ dan $\dot{u}$ dimasukkan pada persamaan (3.3) maka didapatkan persamaan x(k+1) seperti dibawah ini :

$$\begin{bmatrix} \frac{x(k+1)-x(k)}{Ts} \\ \frac{u(k+1)-u(k)}{Ts} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \overline{ax}(k) \quad (3.6)$$

$$\begin{bmatrix} x(k+1)-x(k) \\ u(k+1)-u(k) \end{bmatrix} = \begin{bmatrix} 0 & Ts \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + \begin{bmatrix} 0 \\ Ts \end{bmatrix} \overline{ax}(k) \quad (3.7)$$

$$\begin{bmatrix} x(k+1) \\ u(k+1) \end{bmatrix} = \begin{bmatrix} 0 & Ts \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + \begin{bmatrix} 0 \\ Ts \end{bmatrix} \overline{ax}(k) + \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \quad (3.8)$$

$$\begin{bmatrix} x(k+1) \\ u(k+1) \end{bmatrix} = \begin{bmatrix} 1 & Ts \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + \begin{bmatrix} 0 \\ Ts \end{bmatrix} \overline{ax}(k) \quad (3.9)$$

Dari persamaan (3.9) didapatkan nilai A=$\begin{bmatrix} 1 & Ts \\ 0 & 1 \end{bmatrix}$ dan nilai B=$\begin{bmatrix} 0 \\ Ts \end{bmatrix}$. Ts disini merupakan *time sampling* dari tiap pengambilan data. Ts yang dipakai pada sistem navigasi ini sendiri adalah 0.02s dengan frekuensi 50 Hz.

Setelah didapatkan nilai A dan B maka selanjutnya adalah menentukan matriks kovarian *noise* yang akan dipakai. Matriks kovarian *noise* terdiri dari nilai Q (kovarian *noise* sistem) dan R (kovarian *noise* pengukuran). Disini penulis menentukan nilai Q= $\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ dan R=0.5. Nilai ini dapat diganti sewaktu – waktu untuk mendapatkan nilai *error* yang lebih kecil pada sistem. Tahap selanjutnya adalah menentukan parameter yang terdapat pada filter Kalman. Untuk menentukan parameter yang terdapat pada filter Kalman maka dilakukan beberapa langkah seperti dibawah ini :

1. Menentukan nilai penguat Kalman

   Untuk mendapatkan nilai dari penguat Kalman maka nilai K_k, P_k- dan P_k harus didapatkan terlebih dahulu, untuk mencari ketiga variable tadi dapat dilakukan penghitungan dengan Persamaan (2.8), (2.9) dan (2.10).

   K_k disini merupakan bobot optimal filter, P_k- disebut juga sebagai kovarian pra estimasi *error* Kalman pada saat k dan P_k merupakan kovarian paska estimasi *error*. Nilai K_k, P_k- dan P_k dapat dihitung tanpa data.

2. Menentukan estimasi nilai X

   Untuk dapat menentukan estimasi nilai X terlebih dahulu dibutuhkan nilai Z dan barulah setelah itu digunakan persamaan (5) untuk mencari estimasi nilai X pada saat k+1 dan estimasi selanjutnya dapat dicari menggunakan persamaan dibawah ini :

$$\hat{X}(k+1|k+1) = \hat{X}(k+1|k) + \ldots$$
$$\ldots + K_{k+1}(Z(k+1) - H_k\hat{X}(k+1|k)) \qquad (3.10)$$

3. Tahap terakhir adalah menginisialisasi kembali keadaan k menjadi k=k+1, lalu kembali ke tahap awal.

Tahapan pertama merupakan penentuan nilai K_k, P_k^- dan P_k hingga didapatkan nilai K yang konstan. Berikut adalah hasil penghitungan menggunakan Program Matlab berdasarkan persamaan pada tahap 1 :

```
>> K=P0*H'*(inv(H*P0*H'+R))

K =

    0
    0
```

>> P_1^- =(1-(K*H))*P0*(1-K*H)'+K*R*K'

P_1^- =

```
    0    0
    0    0
```

```
>> P1=A* P1^-*A'+Q

P1 =

    0    0
    0    1

>> K=P1*H'*(inv(H*P1*H'+R))
```

K =

0
0

>> P_2^- =(1-(K*H))*P1*(1-K*H)'+K*R*K'
P_2^- =

1 1
1 1

>> P2=A* P_2^- *A'+Q

P2 =

1,0404 1,0200
1,0200 2,0000

>> K=P2*H'*(inv(H*P2*H'+R))

K =

0,6754
0,6622

.
.
.
.

>> P_{99}^- =(1-(K*H))*P98a*(1-K*H)'+K*R*K'

P_{99}^- =

1,0e+03 *

2,9165 2,9458
2,9458 2,9754

>> P49a=A*P_{99}^{-} *A'+Q

P99 =

1,0e+03 *

3,0355 3,0053
3,0053 2,9764

>> K=P99*H'*(inv(H*P99*H'+R))

K =

0,9998
0,9899

>> P_{100}^{-} =(1-(K*H))*P99*(1-K*H)'+K*R*K'

P_{100}^{-} =

1,0e+03 *

2,9165 2,9458
2,9458 2,9754

>>P100=A* P_{100}^{-} *A'+Q

P100 =

1,0e+03 *

3,0355 3,0053
3,0053 2,9764

>> K=P100*H'*(inv(H*P100*H'+R))

K =

0,9998
0,9899

Dari perhitungan diatas didapatkan nilai K = $\begin{bmatrix}0{,}9998\\0{,}9899\end{bmatrix}$. Setelah tahap pertama telah dilakukan dilanjutkan pada tahap kedua yaitu penentuan nilai X. Dalam proses perhitungan parameter – parameternya, tahap kedua ini membutuhkan data. Oleh karena itu prosesnya dapat dilakukan langsung dengan memasukkan data pada persamaan yang sudah ditulis sebelumnya pada tahap kedua tadi. Dibawah ini adalah cuplikan dari program yang digunakan untuk mengolah data akselerasi.

```
for i=1:jm10sek
    %//////////Rata-Rata/////////
    ax=AccelX(i,1)+aofs;

ax5=ax4;ax4=ax3;ax3=ax2;ax2=ax1;ax1=ax;
    axr=(ax1+ax2+ax3+ax4+ax5)/5;
    eax=axr-ax;
    xkp1=A*xk+B*axr;
    Z=H*xk;
    %////////Filter Kalman////////
    exk=exk+K*(Z-(H*exk));
    exk1=A*exk+B*eax;
    XES=xkp1+exk1;

    exk=exk1;
    xk=xkp1;
    out(i,:)=[XES];

end
XES
```

3.3.4 Parameter *Offset* Pembacaan Akselerometer

Setelah dilakukan pembacaan data berkali-kali maka dapat dicari *offset* pada pembacaan datanya. Pencarian *offset* dilakukan secara manual, yaitu dengan menambahkan nilai tertentu pada data masukan

(akselerasi) dan membandingkan hasil keluaran (posisi) pada program MATLAB.

3.4 Kompas *Digital*

Untuk mengaktifkan register dari sensor penulis memanfaatkan *library* QMC5883L. Data *raw* yang telah diambil mengalami beberapa koreksi seperti penambahan sudut deklinasi dan koreksi dalam keadaan reversed. Karena data *raw* pada sensor kompas masih dalam bentuk radian maka nilainya periu dikonversi terlebih dahulu kedalam satuan derajat. Pengkonversiannya dapat dilakukan dengan mengalikan nilai radian dengan 180/phi.

Setelah didapatkan derajat daripada sensor kompas tadi, maka hasilnya kemudian ditampilkan pada HMI ataupun serial monitor dari Arduino IDE. Disini penulis membuat *range* derajat untuk menunjukkan arah-arah yang terdapat pada kompas *digital* ataupun kompas analog. Pengambilan data dari range ini sendiri memanfaatkan kompas *digital* pada *handphone* karena dianggap lebih mudah dan *simple.* Tabel 3.2 merupakan tabel yang menunjukkan *list* dari *range* derajat yang dipakai berdasarkan kompas *digital* tadi.

Tabel 3. 2 *Range* arah mata angin pada kompas

No	Sudut (°)		Arah Mata Angin
	Minimal	Maksimal	
1	158	202	Selatan
2	113	157	Tenggara
3	76	112	Timur
4	23	75	Timur Laut
5	0	22	Utara
6	338	360	Utara
7	293	337	Barat Laut
8	248	292	Barat
9	203	247	Barat Daya

-----Halaman ini sengaja dikosongkan-----

BAB IV
PENGUJIAN DAN ANALISA

Pengujian dan analisis pada bab ini dilakukan untuk mengetahui performa sistem. Diawali dengan pengujian metode yang digunakan mulai dari pengambilan data *raw* sampai penggunaan *Kalman filter.* Kemudian pengujian pada sensor kompas *digital*. Dari metode yang digunakan diterapkan pada *real experiment* dan dibandingkan dengan hasil perhitungan pada MATLAB.

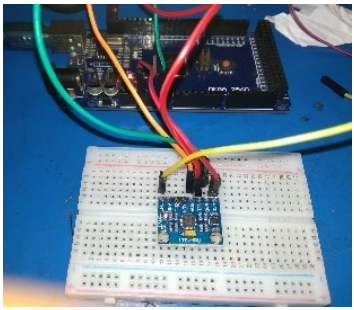
4.1 Hasil Pengujian Pengambilan Data

Pengambilan data pada sensor *accelerometer* ini dilakukan dengan 2 cara yaitu pengambilan dengan program MATLAB dan Arduino IDE. Pengambilan data tadi dilakukan dalam 2 keadaan yaitu pada saat kedua sensor dalam keadaan diam dan pada saat bergerak atau digeser. Untuk sensor kompas *digital*, dilakukan pengambilan data dengan menggerakkan sensor ke berbagai arah untuk kemudian dibandingkan dengan kompas *digital* yang terdapat pada *handphone* sebagai acuannya.

4.1.1 Pengambilan Data dalam Keadaan Diam pada Sensor *Accelerometer*

Keadaan diam disini diartikan sensor berada dalam keadaan tidak bergerak dan sensor diletakkan pada suatu permukaan yang datar. Data *raw* atau data kasar yang diambil pada sensor mula - mula berupa data bit. Untuk mendapat data percepatan maka data tersebut diubah terlebih dahulu kedalam satuan G. Setelah dirubah barulah data kemudian dapat diolah pada proses selanjutnya.

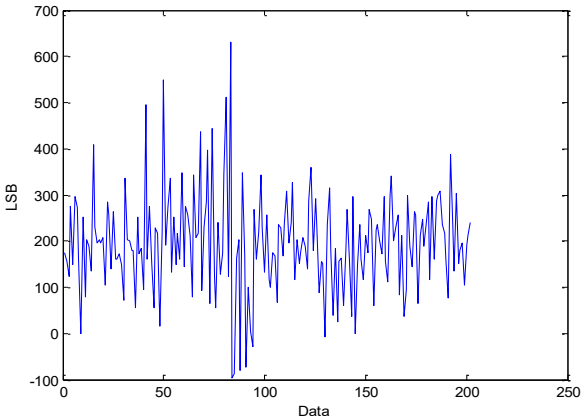
Disini terdapat 2 jenis data percepatan, dimana tiap data merupakan data dari kedua sumbu yang dimiliki sensor. Sumbu tersebut ialah sumbu X yang diperuntukkan untuk gerak maju dan mundur pada ROV, sedangkan sumbu kedua ialah sumbu Y guna menghitung jarak untuk gerak kanan atau kiri pada ROV. Gambar 4.1 merupakan cara pengambilan data *raw* pada sumbu X dan Y *accelerometer*.



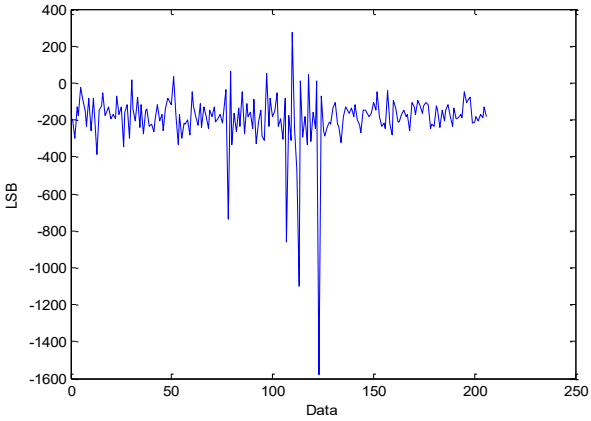
Gambar 4. 1 Pengujian sensor *accelerometer*

Dengan meletakkan sensor pada permukaan datar atau pada kasus ini adalah *breadboard*, maka getaran atau gerakan yang akan menambah *noise* pada sensor dapat dicegah. *Noise* ini sewaktu-waktu dapat terjadi ketika sensor digeser atau dimiringkan oleh pengguna.

Pada keadaan tersebut penulis mengambil dua data sekaligus yaitu data *raw* pada sumbu X dan sumbu Y. Kedua data ini nantinya akan dimasukkan kedalam *filter* Kalman agar *noise* yang pembacaan dapat diminimalisir. Gambar 4.2 menampilkan tampilan *raw* data dari sensor *accelerometer* pada sumbu X sedangkan Gambar 4.3 menampilkan grafik dari *raw* data dari sensor *accelerometer* pada sumbu Y.

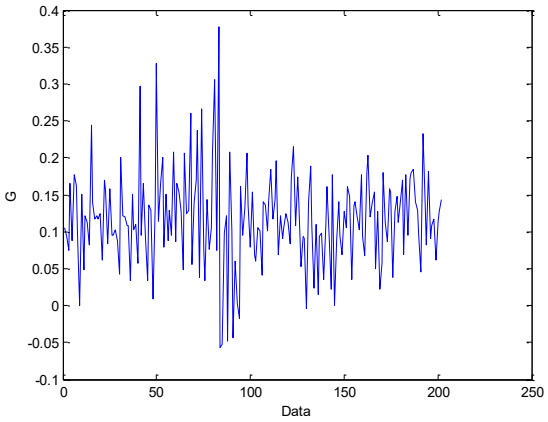


Gambar 4. 2 *Raw* data *accelerometer* sumbu X

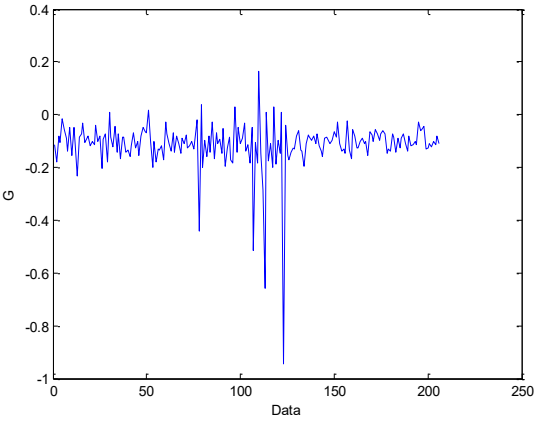


Gambar 4. 3 *Raw* data *accelerometer* sumbu Y

Gambar diatas merupakan bentuk gelombang *raw* data dari pembacaan sensor *accelerometer* yang dipakai. Seperti yang dijelaskan diatas, data masih berupa data LSB *(Least Significant Bit*). Untuk itu dapat digunakan persamaan (3.1) agar dapat menghitung data percepatan dalam satuan G. Dengan memasukkan persamaan tadi didapatkan grafik seperti pada grafik dibawah ini. Dimana pada Gambar 4.4 merupakan grafik pada sumbu X dan Gambar 4.5 pada sumbu Y.



Gambar 4. 4 Data konversi *accelerometer* sumbu X



Gambar 4. 5 Data konversi *accelerometer* sumbu Y

Gambar 4.2, Gambar 4.3, Gambar 4.4 dan Gambar 4.5 merupakan grafik pada salah satu pengujian dalam mendapatkan data *raw* dari sensor *accelerometer.* Data kedua dan selanjutnya pada pengujian yang sama dilampirkan pada Bab lampiran. Pada data *raw* didapatkan nilai yang sangat bervariasi, mulai dari nilai positif hingga negatif meskipun sensor berada dalam posisi diam. Hal ini diakibatkan oleh *noise* yang sangat besar yang terkandung didalam sensor. Nilai yang bervariasi ini dapat ditemukan pada data dari kedua sumbu, baik itu sumbu X ataupun sumbu Y.

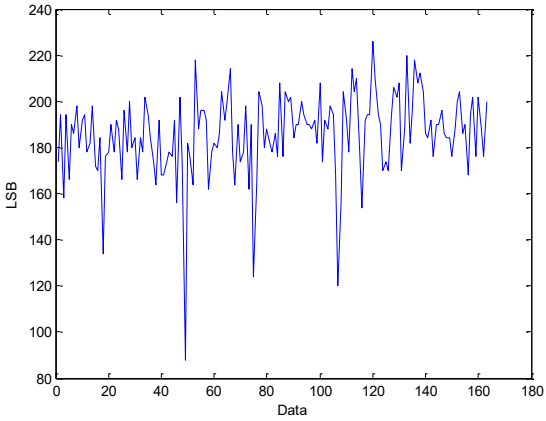
Nilai puncak yang terbaca dalam satuan bit pada sumbu X mencapai nilai 600 dengan nilai terendah -100. Sedangkan untuk sumbu Y nilai puncaknya mencapai 220 dengan nilai terendah -1600. Perbedaan nilai antara nilai tertinggi dan terendah pada sumbu X adalah 700 dan untuk sumbu Y adalah 1380. Apabila dilihat dari bentuk grafik, nilai yang harus dipenuhi pada sumbu X agar stabil adalah pada kisaran 180 sampai 200. Dan untuk sumbu Y sendiri berada pada nilai -200.

Pada saat dikonversi pada satuan G, grafik yang dihasilkan mempunyai bentuk yang serupa, akan tetapi nilai pembacaannya telah mendekati nilai percepatan yang sebenarnya yaitu 0. Nilai 0 disebabkan karena sensor pada pengujian ini berada pada keadaan diam dengan percepatan 0.

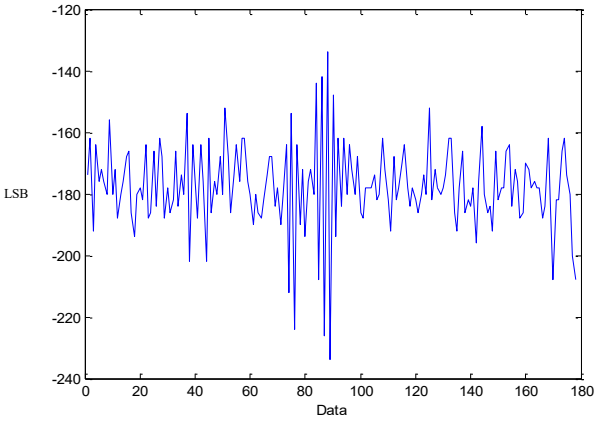
4.1.2 Pengambilan Data dengan *Digital Low Pass Filter*

Dikarenakan *noise* yang masih banyak pada *raw* data yang diambil, data disaring terlebih dahulu melewati *digital low pass filter* (DLPF) untuk kemudian masuk pada Kalman *filter*. *Low pass Filter* disini berguna untuk mengurangi *noise* yang sangat besar pada pembacaan tadi. Karena *digital low pass filter* ini sudah tersedia pada sensor *accelerometer,* maka tambahan program tidak lagi diperlukan.

Seperti yang tertera pada Bab 3, pada pengambilan data sensor *accelerometer* pengguna tinggal memilih bit terakhir pada *address digital low pass filter* yang diinginkan. Disini *low pass filter* menyebabkan *delay* pada setiap pembacaan data sensor. Penulis menggunakan *digital low pass filter* dengan *bandwidth* 50 Hz dalam waktu pengambilan data selama 4 detik. Untuk melihat perbedaan dari setiap pemilihan *digital low pass filter* yang digunakan, maka dapat melakukan pengambilan data yang berbeda sesuai dengan keinginan pengguna. Gambar 4.6 menunjukkan data pembacaan sensor setelah melewati *digital low pass filter* pada sumbu X. Sedangkan pada Gambar 4.7 adalah tampilan grafik pada sumbu Y dengan *digital low pass filter* dengan *setting* yang sama pada sumbu X.



Gambar 4. 6 Data *accelerometer* sumbu X dengan DLPF



Gambar 4. 7 Data *accelerometer* sumbu Y dengan DLPF

Pengambilan data diatas menggunakan satuan bit dibuat agar memudahkan dalam melihat *noise* pada pembacaan sensor. Untuk penggunaan DLPF sendiri nilai yang dihasilkan pada keluaran sensor sudah menjadi lebih baik. Akan tetapi masih terlihat *noise* dari data pada pembacaan sensor.

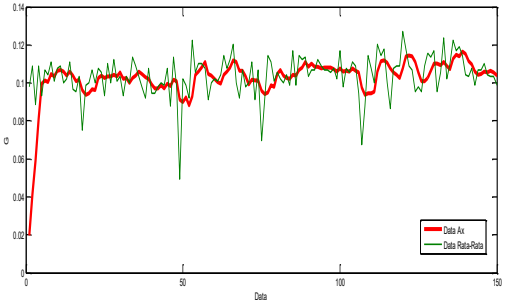
Namun apabila dibandingkan dengan data *raw* sebelumnya, maka dapat terlihat bahwa data ini lebih memiliki sedikit *noise* dibandingkan dengan data *raw.* Pada data *raw* nilai tertinggi pada pembacaan sumbu X dan Y masing – masing adalah 600 dan 300 dengan nilai pembacaan paling rendah pada masing – masing sumbunya adalah -100 dan -1600.

Sedangkan pada penggunaan *digital low pass filter* sendiri, nilai tertinggi dari pembacaan sensor pada sumbu X dan Y masing – masing adalah 220 dan -130 dengan pembacaan paling rendah adalah 70 dan -230. Tentunya nilai ini lebih kecil dibandingkan dengan nilai data *raw.* Jadi dapat ditarik kesimpulan bahwa *noise* pada data mengalami pengurangan ketika melewati *digital low pass filter*. Variasi daripada nilai dari data yang terbaca menjadi lebih kecil terutama pada sumbu Y yang berkurang drastis dari -1300 menjadi 100.

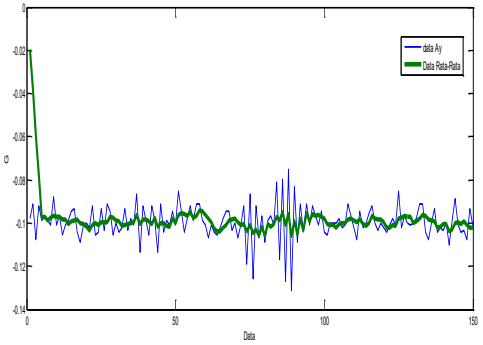
Setelah melakukan pengambilan data dengan penyariangan menggunakan *digital low pass filter* pada kedua sumbu sensor *accelerometer*, data selanjutnya yang diperlukan adalah rata-rata dari

kedua data sumbu tersebut. Pengambilan data rata-rata ditujukan untuk mengurangi nilai *error* lebih lanjut sebelum dilakukan penyaringan Kalman.

Data *raw* diambil sebanyak n kemudian dibagi sebanyak n pula agar didapatkan nilai data rata-rata. Pada kasus ini penulis menentukan nilai n = 5. Disini dapat digunakan persamaan (3.2) seperti yang tertulis pada bab 3 untuk mendapatkan nilai rata – rata tadi. Data Ax dan Ay merupakan data percepatan dari kedua jumlah nilai x dibagi dengan banyak data. Gambar 4.8 merupakan grafik perbandingan nilai sebelum dan setelah diambil rata – rata dari pembacaan nilai sensor pada sumbu X. Sedangkan Gambar 4.9 merupakan grafik perbandingan serupa namun pada sumbu Y.



Gambar 4. 8 Perbandingan data sebelum dan sesudah dirata-ratakan pada sumbu X



Gambar 4. 9 Perbandingan data sebelum dan sesudah dirata-ratakan pada sumbu Y

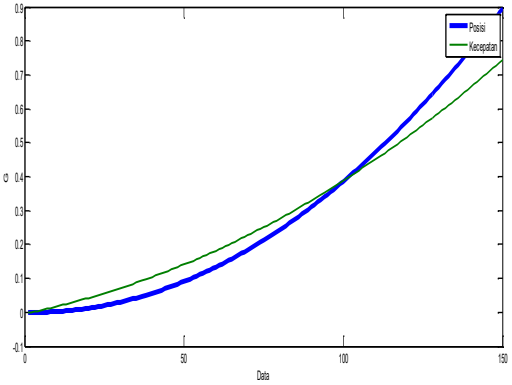
Apabila dilihat dari Gambar 4.8 dan Gambar 4.9 data yang hanya menggunakan *digital low pass filter* terlihat memiliki lebih banyak *noise* sedangkan data yang telah dirata-ratakan lebih mendekati stabil dengan tetap mengikuti grafik dari data yang menggunakan *digital low pass filter*.

Pada Gambar 4.8 data rata – rata sumbu X ditampilkan dengan garis berwarna merah dan data yang hanya disaring menggunakan *digital low pass filter* pada sumbu X ditampilkan dengan garis berwarna hijau. Sedangkan data rata – rata sumbu Y ditampilkan dengan garis berwarna hijau dan data yang hanya disaring menggunakan *digital low pass filter* pada sumbu Y ditampilkan dengan garis berwarna biru.

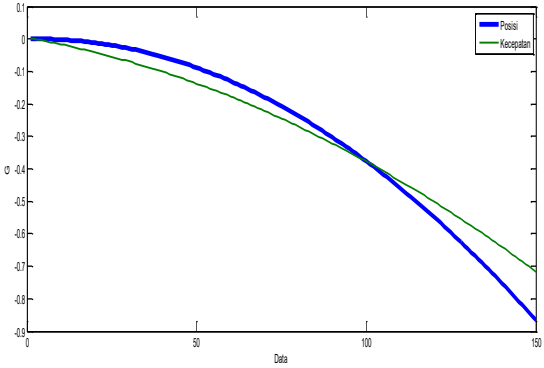
4.1.3 Pengolahan Data pada Kalman *Filter*

Untuk melakukan estimasi terhadap keadaan atau posisi selanjutnya, maka dilakukan penghitungan estimasi *error* menggunakan Kalman *filter*. Karena pengujiannya yang berada pada keadaan diam, nilai dari estimasi *error* ini dibuat agar mendekati angka 0. Untuk proses pengolahan data pada Kalman *filter* ini dapat melakukan tahapan – tahapan yang telah dijelaskan pada sub bab Parameter Kalman *Filter*.

Gambar 4.10 dan 4.11 menampilkan hasil dari pengolahan data menggunakan persamaan Kalman *filter* dengan keluaran berupa posisi dan kecepatan.



Gambar 4. 10 Nilai kecepatan dan posisi Kalman f*ilter* pada sumbu X

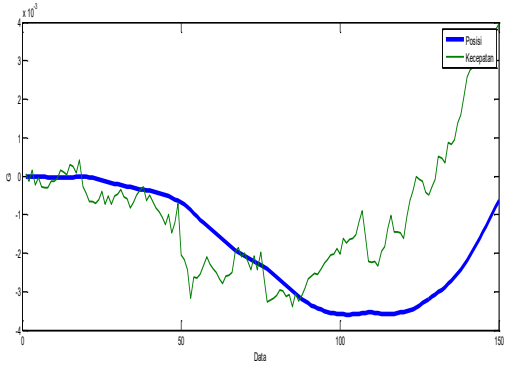


Gambar 4. 11 Nila**i** kecepatan dan posisi Kalman *filter* pada sumbu Y

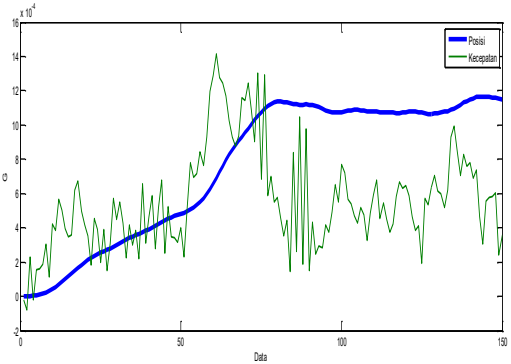
Berdasarkan Gambar 4.10 data posisi ditampilkan dengan garis berwarna biru dan data kecepatan ditampilkan dengan garis berwarna hijau. Begitupula pada Gambar 4.11 data posisi ditampilkan dengan garis berwarna biru dan data kecepatan ditampilkan dengan garis berwarna hijau. Dengan mengamati kedua gambar grafik tersebut maka dapat ditarik kesimpulan bahwa data sebelumnya yang memiliki *noise* ataupun error kini menjadi lebih mendekati keadaan yang sebenarnya. Nilai keluaran pun telah mendekati nilai *real* yaitu dengan *error* sebesar 0 m. Hal ini terjadi pada kedua sumbu, baik itu sumbu X maupun pada sumbu Y.

4.1.4 Pengambilan Data *Accelerometer* dengan Penambahan Nilai *Offset*

Untuk dapat mengurangi *error* pada sensor *accelerometer* itu sendiri maka diperlukan *alignment* terhadap *sample* data dari nilai masukan daripada sensor *Accelerometer*. *Sample* data yang dimaksud disini adalah mengambil data dalam kurun waktu tertentu untuk digunakan sebagai acuan untuk pembacaan data sensor selanjutnya. Seperti yang dijelaskan pada Bab 3, penentuan nilai *offset* dilakukan secara manual. Nilai *offset* yang didapat sebesar 0.08 dan 0.09994 pada masing – masing sumbu X dan Y. Dengan memasukkan nilai *offset* tadi maka didapat grafik seperti pada Gambar 4.12 pada sumbu X. Sedangkan untuk sumbu Y didapatkan grafik seperti pada Gambar 4.13.



Gambar 4. 12 Nilai keluaran berupa posisi dan kecepatan dengan pemberian nilai *offset* pada sumbu X

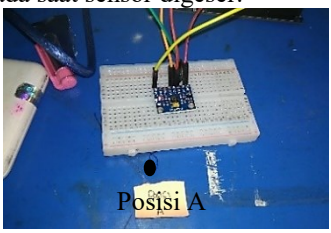


Gambar 4. 13 Nilai keluaran berupa posisi dan kecepatan dengan pemberian nilai *offset* pada sumbu Y

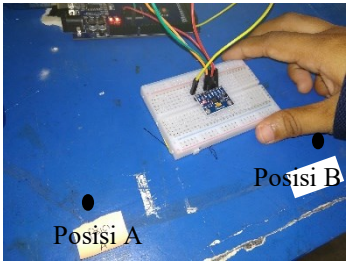
Kedua Gambar yaitu 4.12 dan 4.13 menampilkan hasil dari pengolahan data keluaran berupa nilai posisi dan kecepatan. Data Kecepatan merupakan garis berwarna hijau sedangkan data posisi merupakan garis berwarna biru. Disini pemberian nilai *offset* berguna untuk mengurangi nilai *error* akibat pengintegralan dua kali sebelumnya. Karena nilai *error* tersebut berpola kenaikannya, maka diharapkan penambahan nilai *offset* ini dapat memangkas nilai *error* pada pembacaan sensor. Dengan menambahkan nilai *offset* pada data masukan maka didapat nilai ouput sumbu X dengan 0,0003m pada data posisi dan -0,0008m pada data kecepatan. Sedangkan pada sumbu Y adalah 0,0012m pada data posisi dan 0,0002m pada data kecepatan.

4.2 Pengambilan Data *Accelerometer* pada Keadaan Digeser

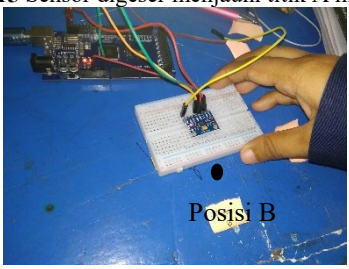
Pengambilan data sensor *accelerometer* yang terakhir adalah pengambilan data pada keadaan sensor digeser berdasarkan sumbu X dan sumbu Y nya. Sumbu X dan Y digeser linier menuju jarak tertentu. Gambar pengambilan dibawah ini menampilkan bentuk fisik pengambilan data pada saat sensor digeser.



Gambar 4. 14 Sensor berada di titik A



Gambar 4. 15 Sensor digeser menjauhi titik A menuju titik B



Gambar 4. 16 Sensor berada di titik B

Gambar 4.14 menampilkan apabila sensor berada pada posisi A, lalu pada Gambar 4.15 sensor tengah digeser menuju titik B. Dan pada Gambar 4.16 sensor telah mencapai titik B. Pengambilan data dilakukan

dengan memproses data yang telah diambil sebelumnya pada Arduino barulah kemudian diolah di Matlab untuk diambil *sample* data.

Tabel 4.1 Data posisi *accelerometer* pada sumbu X berdasarkan jarak

No	Waktu (s)	Jarak (m)		Persentase *Error* (%)
		Real	Sensor	
1	20	0	0	0
2	20	0,5	0,6	20
3	20	1	0,95	5
4	20	0,6	0,65	8,33
			Rata-rata	8.33

Tabel 4.2 Data posisi *accelerometer* pada sumbu X berdasarkan waktu.

No	Waktu (s)	Jarak (m)		Persentase *Error* (%)
		Real	Sensor	
1	20	0	0	0
2	70	0	0,1	0
3	20	1	0,95	5
4	70	1	2,1	110
5	30	0,8	1	20
6	100	0,8	2,3	187,5
			Rata-rata	99,16

Berdasarkan Tabel 4.1 merupakan pengujian data *accelerometer* pada sumbu X berdasar jarak**.** Sedangkan Tabel 4.2 merupakan pengujian *accelerometer* pada sumbu X berdasarkan waktu. Selain itu pengujian juga dilakukan pada sumbu Y dimana ditampilkan pada Tabel 4.3 yang merupakan pengujian data *accelerometer* pada sumbu Y berdasarkan jarak, dan Tabel 4.4 yaitu pengujian data *accelerometer* pada sumbu Y berdasarkan waktu**.**

Tabel 4.3 Data posisi *accelerometer* pada sumbu Y berdasarkan jarak

No	Waktu (s)	Jarak (m)		Persentase *Error* (%)
		Real	Sensor	
1	20	0	0	0
2	20	0,3	0,32	6,67
3	20	0.5	0,59	18
4	20	1	1.08	8
			Rata-rata	8.67

Tabel 4.4 Data posisi *accelerometer* pada sumbu Y berdasarkan waktu

No	Waktu (s)	Jarak (m)		Persentase *Error* (%)
		Real	Sensor	
1	20	0	0	0
2	100	0	0,06	0
3	20	0.5	0,59	5
4	100	0.5	0,98	96
5	20	1	1.08	20
6	100	1	2	100
			Rata-rata	51

Pada pengambilan data berdasarkan Tabel 4.1 yaitu pengujian data pengujian *accelerometer* pada sumbu X berdasar jarak, pengujian dimulai dari jarak 0m hingga 1m ditentukan secara acak untuk melihat perbandingan persentase *error* yang didapatkan dengan *time* yang sama. Sedangkan pada Tabel 4.2 pengujian *accelerometer* pada sumbu X berdasar waktu, pengujian dilakukan dengan membedakan waktu dari tiap data yang diambil. Dari pengujian tersebut ditemukan bahwa ternyata faktor yang paling menentukan adalah waktu yang ditempuh. Data yang didapat pada saat waktu yang sama *error* rata-rata yang didapatkan adalah 8,33%. Sedangkan pada saat waktu pengambilannya

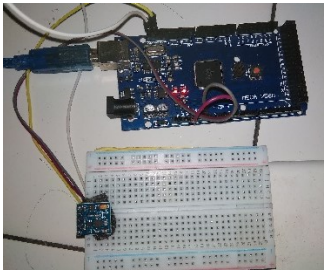
dibuat lebih lama yaitu hingga 100 detik, rata-rata *error* yang didapat mencapai 99,16%. Sedangkan pada Tabel 4.3 yaitu pengujian data pengujian *accelerometer* pada sumbu Y berdasar jarak, nilai rata-rata *error* yang didapat ketika pengambilan data berlangsung selama 20 detik adalah 8,67%. Dan pada Tabel 4.4 yaitu pengujian data pengujian *accelerometer* pada sumbu Y berdasar waktu, saat pengambilan data diperlama hingga 100 detik *error* rata-rata yang didapat mencapai 51%.

Error dari pembacaan sensor dapat mencapai hingga puluhan cm. Semakin lama waktu yang diperlukan sensor untuk menuju titik tertentu maka menyebabkan data yang diintegralkan pun menjadi semakin banyak. Pengintegralan yang berkali – kali tersebut menyebabkan nilai keluaran pada sensor akan tetap naik dan menambah nilai *error* pembacaan.

Untuk pengujian secara *real time* sendiri sensor tidak mampu menampilkan data yang mendekati data *real*. Data mengalami *error* sebesar 10m hanya dalam rentan waktu 5 detik dan terus mengalami kenaikan. Untuk itu penulis tidak menampilkannya pada Bab 4 ini.

4.3 Pengujian pada Data Sensor Kompas *Digital*

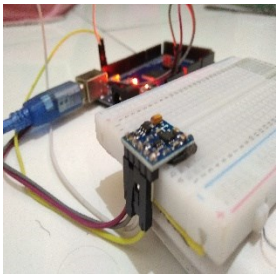
Kompas *digital* bertujuan untuk menunjukkan arah *heading* dari ROV berdasarkan arah mata angin. Pengujian dilakukan pada saat sensor dalam keadaan diam dan diputar hingga 360°. Gambar 4.17 menampilkan bentuk fisik cara pengujian pada sensor kompas.



Gambar 4. 17 Pengujian sensor GY271 pada saat diam

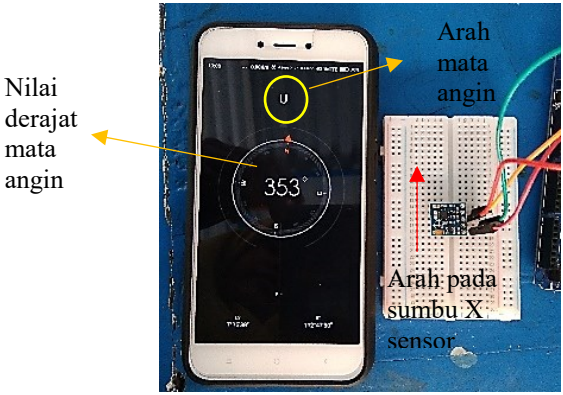
Untuk mencegah pergerakan yang dapat mempengaruhi keluaran sensor, maka sensor ditempelkan pada *breadboard* agar sensor tetap dapat tegak berdiri. Jadinya penggeseran sensor terhadap arah yang dikehendaki menjadi lebih mudah. Gambar 4.18 merupakan bentuk fisik

daripada peletakan sensor pada *breadboard* untuk kemuadian diambila datanya.



Gambar 4. 18 Pengujian sensor GY271.

Dari pengujian seperti pada Gambar 4.18 penulis mengambil data berupa keluaran sudut mengikuti pergeseran sensor pada tiap mata angin yang ada. Disini data yang dibandingkan adalah data pada pembacaan sensor kompas *digital* dengan kompas pada *handphone*. Dari data tadi kemudian didapat persentase *error* dari pembacaan data pada sensor. Gambar 4.19 merupakan gambar pada saat pengujian menggunakan kompas pada *handphone.* Sensor diletakkan bersebelahan dengan kompas pada *handphone* agar arah yang dituju pada saat dilakukan pergeseran pada kedua benda sama. Pengujian dapat dilihat pada Gambar 4.19.



Gambar 4.19 Pengujian sensor GY271 dengan menggunakan kompas pada *handphone*

Dari pengujian seperti diatas didapatkan data perbandingan sensor kompas dengan kompas *digital* yang terdapat pada *handphone* seperti pada Tabel 4.5.

Tabel 4.5 Hasil perbandingan data pada sensor kompas dan kompas *digital handphone*

No	Sudut (°)		Arah		Persentase *Error* (%)
	Kompas *Handphone*	Sensor Kompas	Kompas *Handphone*	Sensor Kompas	
1	345	345	Utara	Utara	0
2	307	310	Barat Laut	Barat Laut	0,97
3	362	367	Barat	Barat	1,38
4	174	175	Selatan	Selatan	0,57
5	146	145	Tenggara	Tenggara	0,6
6	85	83	Timur	Timur	2,3
7	62	62	Timur Laut	Timur Laut	0
8	207	210	Barat Daya	Barat Daya	1,44

Berdasarkan Tabel 4.5 diatas terlihat bahwa nilai *error* paling rendah terjadi pada saat sensor diarahkan menuju arah Utara dan Timur Laut, dengan persentase nilai *error* sebesar 0%. Pembacaan *error* terendah selanjutnya merupakan arah Selatan dengan persentase nilai *error* sebesar 0,57%.

Pembacaan *error* terbesar terjadi ketika sensor diarahkan menuju arah Timur Laut. Kompas *digital* pada *handphone* menunjukkan arah Timur Laut dengan persentase *error* 2,352%.

Variasi dengan nilai *error* ini disebabkan karena ketidakmampuan sensor dalam melakukan pembacaan pada sudut tertentu. Nilai *error* akan semakin besar ketika sensor diarahkan mendekati nilai maksimal dan minimum pada *range* arah mata angin di tabel 3.2. Dan nilai *error*nya akan semakin kecil ketika sensor berada pada posisi diantara dari nilai maksimum dan minimum pada *range* tadi.

BAB V
PENUTUP

Setelah melakukan perencanaan, perancangan, dan pengujian alat maka dapat mengambil kesimpulan dan memberikan saran demi penyempurnaan Tugas Akhir ini.

5.1 Kesimpulan

Hasil dari perancangan alat serta pengukuran dari Perancangan Sistem Navigasi ROV Menggunakan Modul Kompas *Digital* Dan *Accelerometer* dapat diambil kesimpulan sebagai berikut:

1. Nilai *error* pada sensor *accelerometer* semakin besar seiring lamanya waktu pengambilan data. Rata-rata nilai *error* mencapai 99,16% dalam waktu 100 detik pada sumbu X dan 51% pada sumbu Y. Sehingga pengaplikasian pada ROV belum dapat dilakukan, dikarenakan nilai *error* yang besar tadi.
2. Sensor kompas *digital* mampu menampilkan kedelapan arah mata angin yang terdapat pada kompas *handphone*. Rata-rata *error* paling besar didapat pada arah Barat Daya dengan 3,81% dan paling kecil pada arah Barat Laut dengan 0,23%.

5.2 Saran

Untuk pengembangan dan penyempurnaan pembuatan Sistem Perancangan Sistem Navigasi ROV Menggunakan Modul Kompas *Digital* Dan *Accelerometer*, maka diberikan beberapa saran sebagai berikut:

1. Menggunakan sensor *accelerometer* yang telah terintegrasi dan memiliki *noise* pembacaan yang kecil. Dikarenakan kemampuan pengambilan data sensor merupakan faktor yang paling berpengaruh dalam pemrosesan sistem.
2. Dalam peletakan sensor disarankan menggunakan *breadboard* untuk menghindari kemiringan yang tidak diinginkan sehingga nilai pembacaan tidak mengalami perubahan nilai yang drastis.

3. Untuk mengetahui nilai *error* secara lebih teliti dapat digunakan durasi lebih panjang dengan *time sampling* yang lebih kecil dan melakukan penggantian skala sensitivitas.
4. Memperbanyak *literature* untuk dapat mempermudah dalam pendesainan dan pemrograman sistem.
5. Dalam peletakan sensor kompas dan *accelerometer* sebaiknya diletakkan pada bagian tengah ROV dan menjauhi motor guna mengurangi *error* dari luar.

DAFTAR PUSTAKA

[1] Riyadi M., dkk, “Pendeteksi Posisi Menggunakan Sensor Accelerometer Mma7260q Berbasis Mikrokontroler Atmega 32”, Jurnal Penelitian Jurusan Teknik Elektronika – Universitas Diponegoro, Semarang, 2011.

[2] Zul Hafizhuddin Fahmi1., dkk, “Implementasi Complementary Filter Menggunakan Sensor Accelerometer dan Gyroscope pada Keseimbangan Gerak Robot Humanoid”, Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer Fakultas Ilmu Komputer – Universitas Brawijaya, Malang, 2017.

[3] Komitmen, "Remotely Operated Vehicle (ROV)" [Online]. Available: http://www.komitmen.org/2016/11/Remotely OperatedVehicles_10.html. [Accessed 29 April 2018]

[4] Alma’i, Vidi Rahman, Aplikasi Sensor Accelerometer pada Pendeteksi Posisi, Penelitian Teknik Elektro Universitas Diponegoro. Semarang, 2009.

[5] Riyadi M., dkk, “Pendeteksi Posisi Menggunakan Sensor Accelerometer Mma7260q Berbasis Mikrokontroler Atmega 32”, Jurnal Penelitian Jurusan Teknik Elektronika – Universitas Diponegoro, Semarang, 2011.

[6] Ribeiro Maria Isabel, Kalman and Extended Kalman Filters: Concept, Derivation and Properties, Institute for System and Robotics. Portugal, 2004

[7] M. Banzi, Getting Started with Arduino, California: O'Reilly, 2008.

[8] Invesense, “MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2”, InvenSense Inc. USA, 2013.

[9] QST, “3-Axis Magnetic Sensor QMC5883L”,QST Cooporation, Shanghai, 2016.

[10] D. H. Titterton, J. L. Weston, Strapdown Inertial Navigation Technology, Peregrinus, Ltd. London, 1997

[11] M. Bao, Micro Mechanical Transducers: Pressure Sensors, Accelerometers, And Gyroscope, Elsevier, 2000.

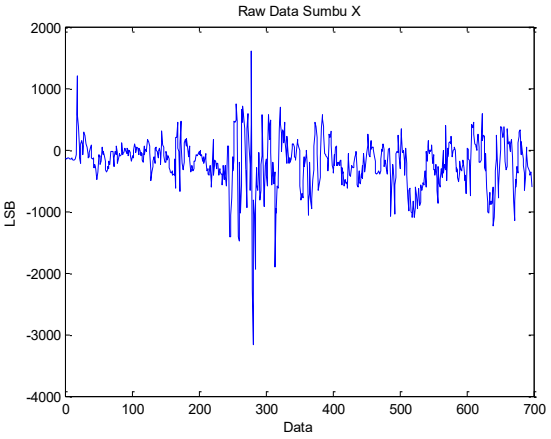
[12] Nawrat Aleksander, dkk, Inertial Navigation Systems and Its Practical Applications, 2010.

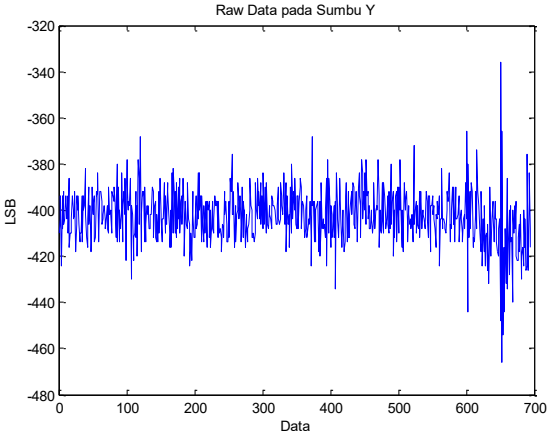
[13] Wahyudi, Adhi Susanto, Sasongko Pramono Hadi, Wahyu Widada, Simulasi Filter Kalman untuk Estimasi Posisi dengan Meggunakan Sensor Accelerometer, Jurnal Techno Science. Semarang, 2009.
[14] Welch Greg dan Bishop Gary, An Introduction to the Kalman Filter. Department of Computer Science University of North Carolina. Chapel Hill, 2006.

LAMPIRAN

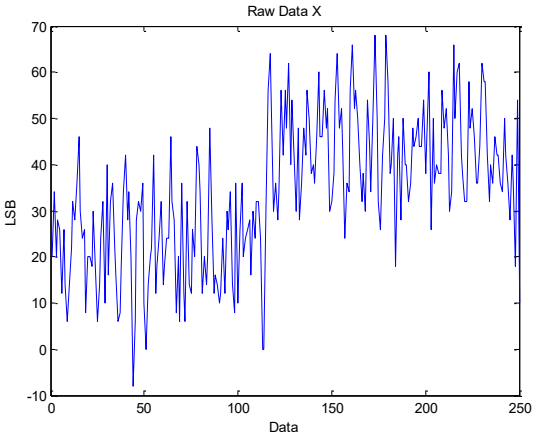
A. Grafik Raw Data pada Sumbu X dan Y

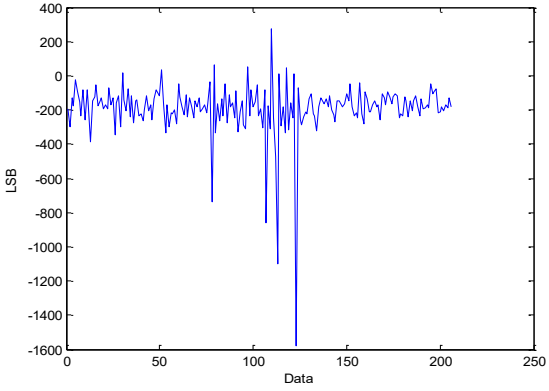
A.1 Pengujian 1



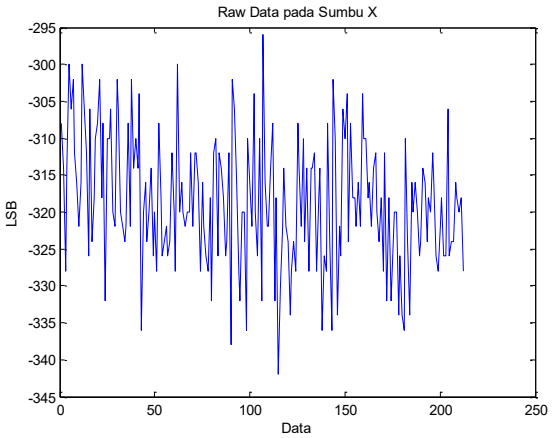


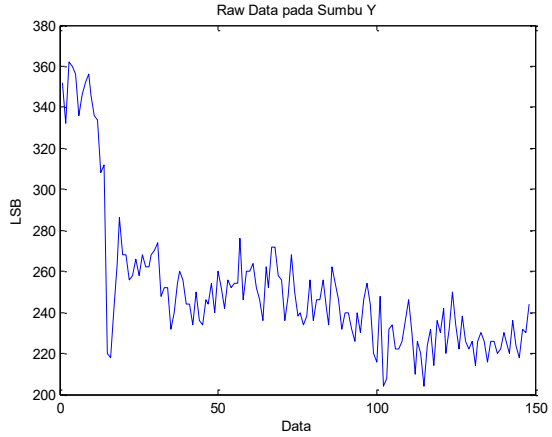
A.2 Pengujian 2



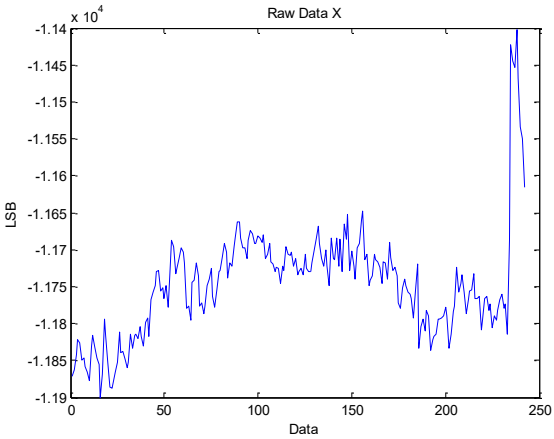


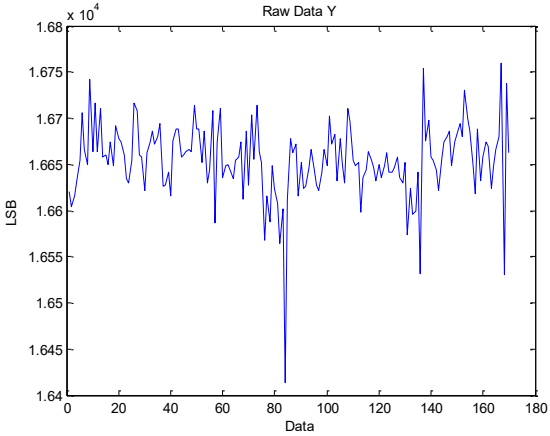
A.3 Pengujian 3





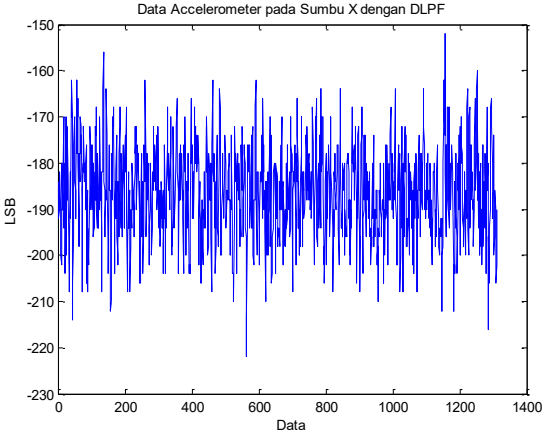
A.4 Pengujian 4

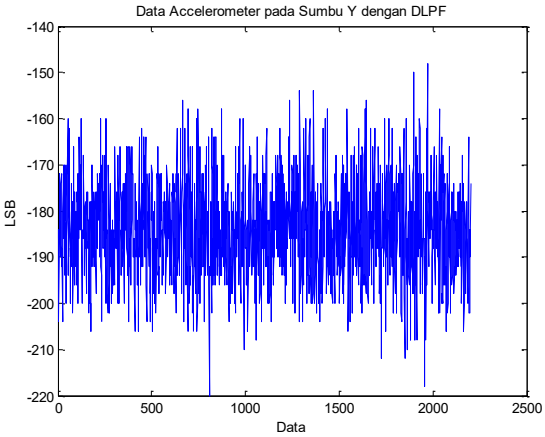




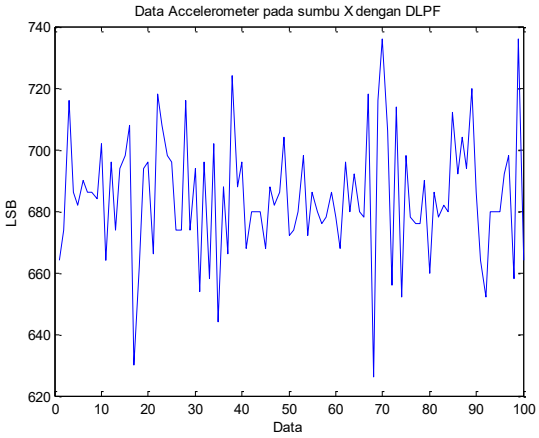
B. Grafik Data dengan DLPF

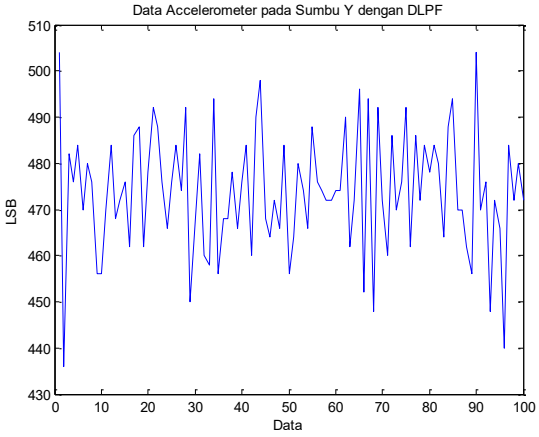
B.1 Pengujian 1



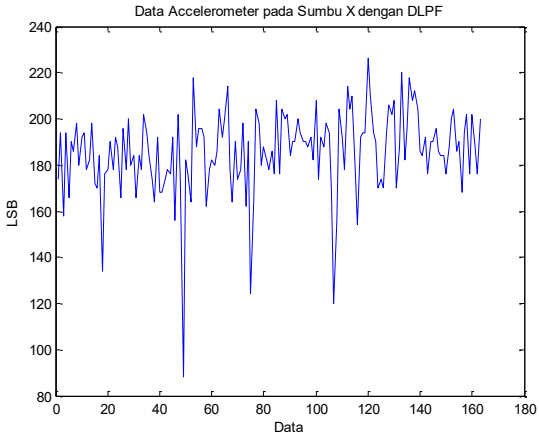


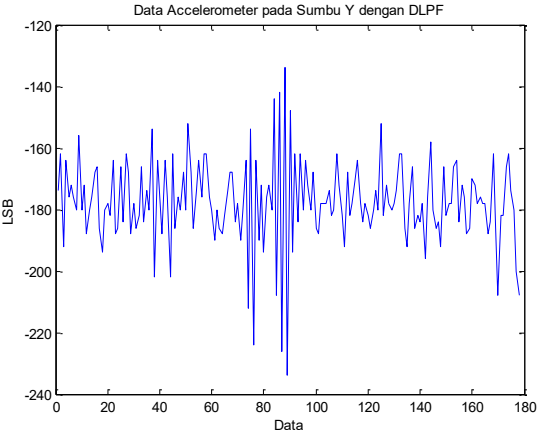
B.2 Pengujian 2



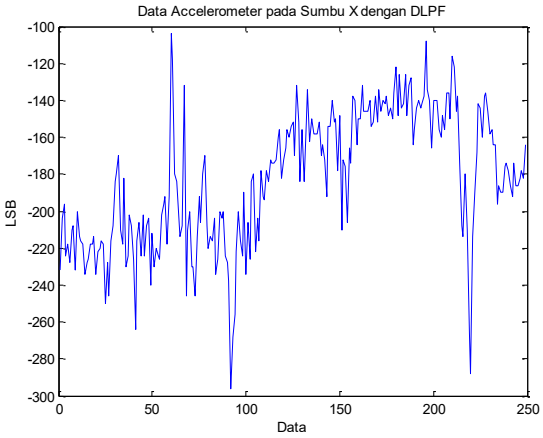


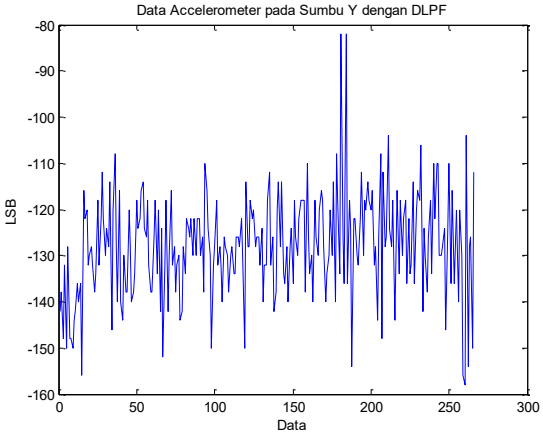
B.3 Pengujian 3





B.4 Pengujian 4





C. Tabel Pengujian Kompas terhadap 8 Arah Mata Angin

No	Sudut Kompas (°)		Arah Mata Angin		Persentase *Error* (%)
	Sensor Kompas	Kompas *Handphone*	Sensor Kompas	Kompas *Handphone*	
1	145	146	Tenggara	Tenggara	0,68
2	147	163	Tenggara	Tenggara	6,36
3	145	148	Tenggara	Tenggara	2,02
4	140	142	Tenggara	Tenggara	1,40
5	145	149	Tenggara	Tenggara	2,68
				Rata-Rata	3,32

No	Sudut Kompas (°)		Arah Mata Angin		Persentase *Error* (%)
	Sensor Kompas	Kompas *Handphone*	Sensor Kompas	Kompas *Handphone*	
1	62	62	TimuLaut	Timur Laut	0
2	70	71	Timur Laut	Timur Laut	1,40
3	53	53	Timur Laut	Timur Laut	0
4	45	46	Timur Laut	Timur Laut	2,17
5	37	37	Timur Laut	Timur Laut	0
				Rata-Rata	0,71

No	Sudut Kompas (°)		Arah Mata Angin		Persentase *Error* (%)
	Sensor Kompas	Kompas *Handphone*	Sensor Kompas	Kompas *Handphone*	
1	166	165	Selatan	Selatan	0,60
2	174	171	Selatan	Selatan	1,75
3	189	188	Selatan	Selatan	0,53
4	161	162	Selatan	Selatan	0,61
5	175	173	Selatan	Tenggara	1,15
				Rata-Rata	0,68

No	Sudut Kompas (°)		Arah Mata Angin		Persentase *Error* (%)
	Sensor Kompas	Kompas *Handphone*	Sensor Kompas	Kompas *Handphone*	
1	247	250	Barat	Barat	1,2
2	257	261	Barat	Barat	1,53
3	259	263	Barat	Barat	1,52
4	257	257	Barat	Barat	0
5	283	288	Barat	Barat	1,73
				Rata-Rata	1,97

No	Sudut Kompas (°)		Arah Mata Angin		Persentase *Error* (%)
	Sensor Kompas	Kompas *Handphone*	Sensor Kompas	Kompas *Handphone*	
1	19	20	Utara	Utara	5
2	342	340	Utara	Utara	0,58
3	355	356	Utara	Utara	0,28
4	356	360	Utara	Utara	1,11
5	17	17	Utara	Utara	0
				Rata-Rata	1,16

No	Sudut Kompas (°)		Arah Mata Angin		Persentase *Error* (%)
	Sensor Kompas	Kompas *Handphone*	Sensor Kompas	Kompas *Handphone*	
1	241	251	Barat Daya	Barat Daya	3,98
2	238	250	Barat Daya	Barat Daya	4,8
3	230	240	Barat Daya	Barat Daya	4,16
4	224	234	Barat Daya	Barat Daya	4,27
5	203	207	Barat Daya	Barat Daya	1,93
				Rata-Rata	3,81

No	Sudut Kompas (°)		Arah Mata Angin		Persentase *Error* (%)
	Sensor Kompas	Kompas *Handphone*	Sensor Kompas	Kompas *Handphone*	
1	328	327	Barat Laut	Barat Laut	0,3
2	334	333	Barat Laut	Barat Laut	0,3
3	320	315	Barat Laut	Barat Laut	1,58
4	305	302	Barat Laut	Barat Laut	0,99
5	295	301	Barat Laut	Barat Laut	1,99
				Rata-Rata	0,23

D. Listing Program

D.1 Program Matlab

```
clear all
close all
clc
%//////////////Sumbu X///////////////
load AccelX.txt
AccelX=(9.2/16384)*AccelX;
ax1=0;
ax2=0;
ax3=0;
ax4=0;
ax5=0;
%//////////////Sumbu Y///////////////
load AccelY.txt
AccelY=(9.2/16384)*AccelY;
jmd=length(AccelY);
ay1=0;
ay2=0;
ay3=0;
ay4=0;
ay5=0;
%//////////////Inisialisasi///////////////
Aa=[1 0.02;0 1];
Ba=[0 0.02]';
Ca=[1 0];
```

```
Ka=[0.9989 0.8889]';
%//////////////Sumbu X///////////////
axk=[0 0]';
aexk=[0 0]';
jm10sek=500
aofsx=0.278054
%//////////////Sumbu Y///////////////
ayk=[0 0]';
aeyk=[0 0]';
aofsy=0.103003
%/////////////////Gyro///////////////
load GyroX.txt
GyroX=GyroX/131;
gx1=0;
gx2=0;
gx3=0;
gx4=0;
gx5=0;
load GyroY.txt
GyroY=GyroY/131;
gy1=0;
gy2=0;
gy3=0;
gy4=0;
gy5=0;
Ay=0;
By=1;
Cy=1;
Ky=0.6667;
gxk=0;
egxk=0;
gyk=0;
egyk=0
jm10sek=500
gofsx=0.4759;
gofsy=1.3817;
for i=1:jm10sek
    %//////////////Gyro Sumbu X///////////////
    gx=GyroX(i,1)+gofsx;
    gx5=gx4;gx4=gx3;gx3=gx2;gx2=gx1;gx1=gx;
```

```
    grx=(gx1+gx2+gx3+gx4+gx5)/5;
    egx=grx-gx;
    gxkp1=Ay*gxk+By*grx;
    gxr=Cy*gxk;
   %///////////////Filter
Kalman//////////////////
    egxk=egxk+Ky*(gxr-(Cy*egxk));
    egxkp1=Ay*egxk+By*egx;
    GXS=gxkp1+egxkp1;
    egxk=egxkp1;
    gxk=gxkp1;
    %///////////////Gyro Sumbu Y///////////////
    gy=GyroY(i,1)+gofsy;
    gy5=gy4;gy4=gy3;gy3=gy2;gy2=gy1;gy1=gy;
    gry=(gy1+gy2+gy3+gy4+gy5)/5;
    egy=gry-gy;
    gykp1=Ay*gyk+By*gry;
    gyr=Cy*gyk;
   %///////////////Filter
Kalman//////////////////
    egyk=egyk+Ky*(gyr-(Cy*egyk));
    egykp1=Ay*egyk+By*egy;
    GYS=gykp1+egykp1;
    egyk=egykp1;
    gyk=gykp1;
    axakhir=sin(GYS)*cos(GXS);
    ayakhir=axakhir*-1;
    %///////////////Accelero                Sumbu
X////////////////
    ax=AccelX(i,1)+aofsx;
    ax5=ax4;ax4=ax3;ax3=ax2;ax2=ax1;ax1=ax;
    arx=(ax1+ax2+ax3+ax4+ax5)/5;
    eax=arx-ax;
    axkp1=Aa*axk+Ba*arx;
    axr=Ca*axk;
    %///////////////Accelero                Sumbu
Y////////////////
    ay=AccelY(i,1)+aofsy;
    ay5=ay4;ay4=ay3;ay3=ay2;ay2=ay1;ay1=ay;
    ary=(ay1+ay2+ay3+ay4+ay5)/5;
```

```
    eay=ary-ay;
    aykp1=Aa*ayk+Ba*ary;
    ayr=Ca*ayk;
    %//////////////Filter
Kalman//////////////////
    aexk=aexk+Ka*(axr-(Ca*aexk));
    eaxkp1=Aa*aexk+Ba*eax;
    AXS=axkp1+eaxkp1-axakhir;
    aexk=eaxkp1;
    axk=axkp1;
    %//////////////Filter
Kalman//////////////////
    aeyk=aeyk+Ka*(ayr-(Ca*aeyk));
    eaykp1=Aa*aeyk+Ba*eay;
    AYS=aykp1+eaykp1-ayakhir;
    aeyk=eaykp1;
    ayk=aykp1;
%    out(i,:)=[AXS AYS];


end
GXS
GYS
AXS
AYS
```

D.2 Program Arduino (*Accelerometer*)

```
#include <Wire.h>
////////////Accelero & Gyro//////////////
long accelX, accelY, accelZ;
float gForceX, gForceY, gForceZ;
long gyroX, gyroY, gyroZ;
float rotX, rotY, rotZ;
int i;
//////Gyro///////
int gx,gy,gz,gxx,gyy,gzz;
int gx1=0,gx2=0,gx3=0,gx4=0,gx5=0;
int gy1=0,gy2=0,gy3=0,gy4=0,gy5=0;
```

```
int gz1=0,gz2=0,gz3=0,gz4=0,gz5=0;
float gxr,gyr,gzr,grx,gry,grz;
//////Accelero//////
int ax,ay,az,axx,ayy,azz;
int ax1=0,ax2=0,ax3=0,ax4=0,ax5=0;
int ay1=0,ay2=0,ay3=0,ay4=0,ay5=0;
int az1=0,az2=0,az3=0,az4=0,az5=0;
float axr,ayr,azr,arx,ary,eax,eay;
float  aofsx=0.278054,aofsy=0.103003;

//////////Matriks Accelerometer/////////////
float ab=1, ac=0.02, ad=0, ae=1;
float bb=0, bc=0.02;
float cb=1, cc=0;
float kb=0.9998, kc=0.9899;
/////////Sumbu X//////////////
float axka=0, axkb=0;
float aexka=0, aexkb=0;
float axkp2=0, axkp3=0;
//float eaxkpa=0, eaxkpb=0;
float eaxkpa1=0, eaxkpb1=0;
float AXSa=0, AXSb=0;
///////Sumbu Y//////////
float ayka=0, aykb=0;
float aeyka=0, aeykb=0;
float aykp2=0, aykp3=0;
//float eaykpa=0, eaykpb=0;
float eaykpa1=0, eaykpb1=0;
float AYSa=0, AYSb=0;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  setupMPU();
compass.init();
}
void loop() {
  recordAccelRegisters();
  delay(20);
```

```
}

void setupMPU(){
  Wire.beginTransmission(0b1101000); //This is the I2C address of the
MPU (b1101000/b1101001 for AC0 low/high datasheet sec. 9.2)
  Wire.write(0x6B); //Accessing the register 6B - Power Management
(Sec. 4.28)
  Wire.write(0b00000000); //Setting SLEEP register to 0. (Required; see
Note on p. 9)
  Wire.endTransmission();
  Wire.beginTransmission(0b1101000);
  Wire.write(0x1A);
  Wire.write(0b00000110);
  Wire.endTransmission();
  Wire.beginTransmission(0b1101000); //I2C address of the MPU
  Wire.write(0x1B); //Accessing the register 1B - Gyroscope
Configuration (Sec. 4.4)
  Wire.write(0x00000000); //Setting the gyro to full scale +/- 250deg./s
  Wire.endTransmission();
  Wire.beginTransmission(0b1101000); //I2C address of the MPU
  Wire.write(0x1C); //Accessing the register 1C - Acccelerometer
Configuration (Sec. 4.5)
  Wire.write(0b00000000); //Setting the accel to +/- 2g
  Wire.endTransmission();
}

void recordAccelRegisters() {
  Wire.beginTransmission(0b1101000); //I2C address of the MPU
  Wire.write(0x3B); //Starting register for Accel Readings
  Wire.endTransmission();
  Wire.requestFrom(0b1101000,6); //Request Accel Registers (3B - 40)
  while(Wire.available() < 6);
  accelX = Wire.read()<<8|Wire.read(); //Store first two bytes into
accelX
  accelY = Wire.read()<<8|Wire.read(); //Store middle two bytes into
accelY
  accelZ = Wire.read()<<8|Wire.read(); //Store last two bytes into accelZ
  processAccelData();
```

```
}
void FilterAccelero(){
//////////////Accelero Sumbu X///////////////
   eax=arx-axx;
   ///axkp1
   axkp2=((ab*axka+ac*axkb)+(bb*arx));
   axkp3=((ad*axka+ae*axkb)+(bc*arx));
   axr=cb*axka+cc*axkb;
//////////////Filter Kalman///////////////
   ////aexk
   aexka=aexka+(kb*(axr-(cb*aexka+cc*aexkb)));
   aexkb=aexkb+(kc*(axr-(cb*aexka+cc*aexkb)));
   ////eaxkp1
   eaxkpa1=((ab*aexka+ac*aexkb)-bb*eax);
   eaxkpb1=((ad*aexka+ae*aexkb)-bc*eax);
   ////AXS
   AXSa=axkp2+eaxkpa1;
   AXSb=axkp3+eaxkpb1;//-axakhir;
   Serial.print("AXS=");
   Serial.println(AXSa);
   Serial.print("AXS 2=");
   Serial.println(AXSb);

   aexka=eaxkpa1;
   aexkb=eaxkpb1;
   axka=axkp2;axkb=axkp3;
   //////////////Sumbu Y//////////////
   //////////////Accelero Sumbu Y///////////////
   eay=ary-ay;
   ///aykp1
   aykp2=((ab*ayka+ac*aykb)+(bb*ary));
   aykp3=((ad*ayka+ae*aykb)+(bc*ary));
   ayr=cb*ayka+cc*aykb;
   //////////////Filter Kalman///////////////
   ////eaykp
   aeyka=aeyka+(kb*(ayr-(cb*aeyka+cc*aeykb)));
   aeykb=aeykb+(kc*(ayr-(cb*aeyka+cc*aeykb)));
   ////eaykp1
```

```
    eaykpa1=((ab*aeyka+ac*aeykb)-bb*eay);
    eaykpb1=((ad*aeyka+ae*aeykb)-bc*eay);
    ////AXS
    AYSa=aykp2+eaxkpa1;
    AYSb=aykp3+eaxkpb1;//-ayakhir;
    Serial.print("AYS=");
  ` Serial.print(AYSa);
    Serial.print("AYS 2=");
    Serial.println(AYSb);

    aeyka=eaykpa1;
    aeykb=eaykpb1;
    ayka=aykp2;aykb=aykp3;
}
void processAccelData(){
axx = accelX*16384/9.8;
ax=axx+aofsx;
ayy = accelY;//16384*9.8;
ay=ayy+aofsy;
azz = accelZ;//16384*9.8;

for(i=0;i<1;i++)
{
//////Sumbu X/////

ax3=ax2;
ax2=ax1;
ax1=ax;
arx=(ax1+ax2+ax3)/3;
Serial.print(arx);
//////Sumbu Y/////
ay5=ay4;
ay4=ay3;
ay3=ay2;
ay2=ay1;
ay1=ay;
ary=(ay1+ay2+ay3+ay4+ay5)/5;
//////Sumbu Z/////
az5=az4;
```

```
az4=az3;
az3=az2;
az2=az1;
az1=az;
azr=(az1+az2+az3+az4+az5)/5;
FilterAccelero();}}
```

D.2 Program Arduino (Kompas)

```
/*
  e-Gizmo QMC5883L GY-271 Compass

  Sample sketch for the GY-271 QMC5883L
  for getting the raw data of x, y, z and
  Radius in degrees.

  Codes by e-Gizmo Mechatronix Central
  http://www.e-gizmo.com
  July 10,2017

*/
#include <Wire.h>
#include <QMC5883L.h>

QMC5883L compass;

void setup() {
  Wire.begin();
  Serial.begin(9600);
  compass.init();

//qmc.setMode(Mode_Continuous,ODR_200Hz,RNG_2G,OSR_256);
}

void loop() {
  int x,y,z;
  compass.read(&x,&y,&z); // membaca data sumbu x, y dan z
```

```
  // Atan2() otomatis mengecek rumus yang benar untuk kuadran yang dipakai
  //menghitung heading ketika magnetometer berubah, kemudian disinkronkan dg sinyal axis
  float heading = atan2(y, x);

  float declinationAngle = -0.090; // menentukan sudut deklinasi (derajat}
  heading += declinationAngle;

  if(heading < 0)   //mengecek saat sinyal berbalik
    heading += 2*PI;

  if(heading > 2*PI) // //mengecek tambahan deklinasi
    heading -= 2*PI;
  //mengubah radian ke derajat
  float headingDegrees = heading * 180/M_PI;
  Serial.print("    Derajat ");
  Serial.print(headingDegrees);
  Serial.print("    Arah mata angin: ");
  if (headingDegrees >= 158 && headingDegrees <= 202) {
     Serial.println("SELATAN");
   } else if (headingDegrees >= 113 && headingDegrees <= 157) {
     Serial.println("TENGGARA");
   } else if (headingDegrees >= 76 && headingDegrees <= 112) {
     Serial.println("TIMUR");
   } else if (headingDegrees >= 23 && headingDegrees <= 75 ) {
     Serial.println("TIMUR LAUT");
   } else if (headingDegrees >= 0 && headingDegrees <= 22) {
     Serial.println("UTARA");
   } else if (headingDegrees >= 338 && headingDegrees <= 360) {
     Serial.println("UTARA");
   } else if (headingDegrees >= 293 && headingDegrees <= 337) {
     Serial.println("BARAT LAUT");
   } else if (headingDegrees >= 248 && headingDegrees <= 292) {
     Serial.println("BARAT");
   } else if (headingDegrees >= 203 && headingDegrees <= 247) {
     Serial.println("BARAT DAYA");}
  delay(500);}
```

-----Halaman ini sengaja dikosongkan-----

RIWAYAT HIDUP PENULIS

Nama	:	Gerdina Ika Wahyu
TTL	:	Sumenep, 12 Maret 1997
Jenis Kelamin	:	Perempuan
Agama	:	Islam
Alamat Rumah	:	Jl Nanas 1 Kolor, Sumenep
Nomor HP	:	087851581015
E-mail	:	gerdina21@gmail.com

RIWAYAT PENDIDIKAN

- 2003-2009 : SDN 1 Sumenep
- 2009-2012 : SMP Negeri 1 Sumenep
- 2012-2015 : SMA Negeri 1 Sumenep
- 2015 – sekarang : Bidang Studi Komputer Kontrol, Program D3Teknik Elektro, ITS

PENGALAMAN KERJA

- PT. TMMI (Toyota Motor Manufacturing Indonesia) (2017)

PENGALAMAN ORGANISASI

- DPM ITS (2017-2018)