REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

SISTEM KENDALI NAVIGASI ROBOT BERODA DENGAN GESTUR TANGAN MENGGUNAKAN METODE *KALMAN FILTER*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Amelio Eric Fransisco
NIM: 145150300111010

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

KENDALI NAVIGASI ROBOT BERODA DENGAN GESTUR TANGAN MENGGUNAKAN METODE *KALMAN FILTER*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Amelio Eric Fransisco
NIM: 145150300111010

Skripsi ini telah diuji dan dinyatakan lulus pada
19 Juli 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	*Dosen Pembimbing II*
Gembong Edhi Setyawan, S.T., M.T.	Rizal Maulana , S.T., M.T., M.Sc.
NIK: 201208 761201 1 001	NIK: 2016078 910091 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan,.S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

Repository Universitas Brawijaya Repository Universitas Brawijaya Repository

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 5 Juli 2019

Amelio Eric Fransisco

NIM: 145150300111010

KATA PENGANTAR

Puji Syukur kehadirat Tuhan yang Maha Esa yang telah melimpahkan rahmat, taufik dan hidayah-Nya, sehingga penulis dapat menyelesaikan laporan skripsi yang berjudul "Sistem Kendali Navigasi Robot Beroda Dengan Gestur Tangan Mengunakan Metode *Kalman filter*" dengan baik.

Penulis menyadari bahwa dalam penyusunan dan penelitian skripsi ini tidak lepas dari bantuan moral dan materil yang diberikan dari berbagai pihak, dengan itu penulis ingin menyampaikan rasa hormat dan terima kasih kepada :

1. Bapak Gembong Edhi Setyawan, S.T., M.T, dan Bapak Rizal Maulana, S.T., M.T., M.Sc selaku Pembimbing yang telah memberikan ilmu, saran, dan motivasi, serta dengan sabar mengarahkan penulis hingga terselesaikan skripsi ini,
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang,
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D, selaku Ketua Juruan Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang,
4. Dahnial Syauqy, S.T., M.T, selaku ketua Program Studi Teknik Komputer Universitas Brawijaya Malang,
5. Kedua orang tua dan seluruh keluarga besar atas segala nasihat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta senantiasa memberikan doa dan semangat,
6. Keluarga besar Laboratorium Drone di bawah naungan Laboratorium Sistem Komputer dan Robotika yang sudah banyak membantu memberi saran, motivasi dan dukungan dalam pengerjaan dan penulisan skripsi,
7. Seluruh Civitas Akademika Fakultas Ilmu Komputer Universitas Brawijaya yang telah turut memberi bantuan, semangat, dukungan dan do'a selama penulis menempuh studi di Teknik Komputer Universitas Brawijaya.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih belum sempurna, sehingga adanya kritik dan saran yang membangun sangat penulis harapkan. Dengan demikian semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi.

Malang, 5 Juli 2019

Penulis

ericshadow@student.ub.ac.id

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

ABSTRAK

Seiring dengan pemanfaatan robot beroda yang semakin meningkat, pengunanya dihadapkan dengan masalah terkait sistem kendali navigasi. Umumnya sistem kendali navigasi robot dikendalikan meggunakan remot kontrol atau *joystick* dengan media kabel. Oleh karena itu, dalam penelitian ini kendali navigasi robot beroda telah dikembangkan dengan menggunakan perintah manusia secara langsung, salah satunya dengan menggunakan gestur tangan. Perangkat gestur tangan dikembangkan dengan memanfaatkan sebuah sensor *inertial measurement unit* (IMU) sebagai pengukur kecepatan,orientasi dan gaya gravitasi menggunakan kombinasi *accelerometer* dan *gyroscope.* Sensor IMU mampu mendeteksi segala bentuk gerakan yang dihasilkan. Namun, hasil pengukurannya seringkali tidak akurat. Algoritma penyaringan *Kalman filter* merupakan pendekatan ideal untuk mengoptimalkan estimasi ketidakakuratan sinyal yang dihasilkan oleh sensor. Hasil pengujian menunjukkan penentuan variabel *Q_angle* = 0.001, *Q_bias* = 0.003 dan Nilai *R_measure* = 0.03 dengan metode *try and error* sangat efektif dalam meminimalisir akibat yang ditimbulkan dari *noise*. Hasil pengujian *complementary filter* juga menunjukkan bahwa nilai konstanta yang tepat untuk digunakan pada sistem adalah 0.93. Selanjutnya nilai yang telah akurat tersebut dikirimkan melalui modul komunikasi menuju perangkat robot beroda untuk mengendalikan navigasi. Berdasarkan pengujian gestur tangan dan navigasi robot beroda mampu bergerak sesuai dengan gestur tangan manusia. Kendali kecepatan robot beroda juga dapat diatur berdasarkan sudut *pitch* dan *roll* gestur tangan sebesar 90 derajat. Pengujian *delay* sistem dan *response time* kendali juga diuji untuk mengetahui tingkat ketanggapan sistem. Hasil pengujian menunjukkan bahwa *delay* saat sistem dinyalakan sebesar 1.91 detik dan *response time* yang dibutuhkan perangkat kendali untuk mengendalikan robot beroda adalah sebesar 0.044 detik.

Kata Kunci : *navigasi, robot beroda, gestur tangan, accelerometer, gyroscope, kalman filter*

REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

ABSTRACT

Along with the increasing use of wheeled robots, users are faced with problems related to the navigation control system. Generally the robot navigation control system is controlled using remote control or joystick with cable media. Therefore, in this study wheeled robot navigation controls have been developed using direct human commands, one of which is by using hand gestures. Hand gesture devices were developed using an inertial measurement unit (IMU) sensor as a measure of speed, orientation and gravitational force using a combination of accelerometer and gyroscope. The IMU sensor is capable of detecting all forms of movement produced. However, the measurement results are often inaccurate. The Kalman filter filtering algorithm is the ideal approach to optimize the estimation of the inaccuracy of signals produced by sensors. The test results show the determination of the variable Q_angle = 0.001, Q_bias = 0.003 and the value of R_measure = 0.03 with the try and error method which is very effective in minimizing the effects of noise. The results of the complementary filter test also indicate that the right constant value for use on the system is 0.93. Furthermore, the accurate value is sent through the communication module to the wheeled robot device to control navigation. Based on hand gesture testing and navigation the wheeled robot is able to move according to the gesture of the human hand. Control of wheeled robot speed can also be adjusted based on the pitch angle and hand gesture roll of 90 degrees. System delay testing and response time control were also tested to determine the level of system responsiveness. The test results show that the delay when the system is turned on is 1,91 seconds and the response time needed by the control device to control the wheeled robot is 0.044 seconds.

***Keywords**: navigation, wheeled robot, hand gesture, accelerometer, gyroscope, kalman filter*

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL x
DAFTAR GAMBAR xiii
DAFTAR LAMPIRAN xvi
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 3
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan masalah 4
1.6 Sistematika pembahasan 4
BAB 2 LANDASAN KEPUSTAKAAN 6
2.1 Tinjauan Pustaka 6
2.2 Landasan Teori 13
2.2.1 Robot beroda 13
2.2.2 *Wereable Device* 14
2.2.3 *Kalman filter* 15
2.2.4 *Inertial Measurement Unit* 16
2.2.5 Arduino 18
2.2.6 *Motor driver* L298D 19
2.2.7 Motor DC *(Battery Operated)* 20
2.2.8 NRF24L01 Module 20
BAB 3 METODOLOGI PENELITIAN 22
3.1 Studi Literatur 22
3.2 Rekayasa Kebutuhan Sistem 23

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

3.3 Perancangan Sistem 24
3.4 Implementasi Sistem 24
3.5 Pengujian dan Analisis 25
3.6 Kesimpulan 26
BAB 4 REKAYASA KEBUTUHAN 27
4.1 Gambaran Umum Sistem 28
4.2 Rekayasa Kebutuhan 29
4.2.1 Kebutuhan Pengguna 29
4.2.2 Kebutuhan Sistem 30
4.2.3 Kebutuhan Fungsional 38
4.2.4 Kebutuhan Non-Fungsional 43
BAB 5 PERANCANGAN DAN IMPLEMENTASI 47
5.1 Perancangan Sistem 47
5.1.1 Perancangan Desain Perangkat Keras 48
5.1.2 Perancangan Akuisisi Sensor 51
5.1.3 Perancangan Pemrosesan 53
5.1.4 Perancangan Komunikasi 63
5.1.5 Perancangan Indikator 64
5.1.6 Perancangan Kendali 65
5.2 Implementasi Sistem 69
5.2.1 Implementasi Desain Perangkat Keras 70
5.2.2 Implementasi Akuisisi Sensor 72
5.2.3 Implementasi Pemrosesan 77
5.2.4 Implementasi Komunikasi 86
5.2.5 Implementasi Indikator 89
5.2.6 Implementasi Kendali 92
BAB 6 PENGUJIAN DAN ANALISIS 97
6.1 Pengujian Filter 98
6.1.1 Pengujian Konstanta *Complementary Filter* 98
6.1.2 Pengujian Kalman Filter 115
6.2 Pengujian Gestur Tangan 126
6.2.1 Tujuan Pengujian 126

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

6.2.2 Prosedur Pengujian 126
6.2.3 Pelaksanaan Pengujian 126
6.2.4 -Hasil Pengujian 126
6.2.5 Analisis Hasil Pengujian 130
6.3 Pengujian Navigasi 131
6.3.1 Tujuan Pengujian 131
6.3.2 Prosedur Pengujian 131
6.3.3 Pelaksanaan Pengujian 132
6.3.4 Hasil Pengujian 133
6.3.5 Analisis Hasil Pengujian 135
6.4 Pegujian *Delay* dan *Response time* 137
6.4.1 Tujuan Pengujian 137
6.4.2 Prosedur Pengujian 137
6.4.3 Pelaksanaan Pengujian 137
6.4.4 Hasil Pengujian 138
6.4.5 Analisis Hasil Pengujian 140
BAB 7 Penutup 142
7.1 Kesimpulan 142
7.2 Saran 143
DAFTAR PUSTAKA 144
LAMPIRAN A KODE PROGRAM 147
LAMPIRAN B AKUISISI DATA UJI *KALMAN FILTER* 154
LAMPIRAN C HASIL UJI GESTUR TANGAN 161

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

DAFTAR TABEL

Tabel 2.1 Tabel perbandingan filter 7
Tabel 2.2 Perbandingan mikrokontroler arduino 19
Tabel 4.1 Format penulisan rekayasa kebutuhan sistem 28
Tabel 4.2 Mengetahui variasi gestur tangan 29
Tabel 4.3 Indikator antarmuka pergerakan robot beroda 30
Tabel 4.4 Fungsi Arduino IDE 31
Tabel 4.5 Spesifikasi kebutuhan Arduino Nano V3 31
Tabel 4.6 Fungsi Arduino Nano V3 (Terletak pada TX) 32
Tabel 4.7 Fungsi Arduino Nano V3 (Terletak pada RX) 32
Tabel 4.8 Spesifikasi kebutuhan *Motor driver* L298N 33
Tabel 4.9 Spesifikasi kebutuhan NRF24L01 33
Tabel 4.10 Spesifikasi kebutuhan Motor DC 34
Tabel 4.11 Spesifikasi kebutuhan Baterai Li-Po 35
Tabel 4.12 Spesifikasi kebutuhan Baterai 9v 35
Tabel 4.13 Spesifikasi kebutuhan PCB *Matrix* 36
Tabel 4.14 Spesifikasi kebutuhan sakelar 36
Tabel 4.15 Spesifikasi kebutuhan kabel *jumper* 36
Tabel 4.16 Spesifikasi dan fungsi akrilik 37
Tabel 4.17 Spesifikasi kebutuhan dan fitur MPU-6050 37
Tabel 4.18 Menjelaskan kebutuhan menghubungkan perangkat TX dan RX 38
Tabel 4.19 Menjelaskan berbagai macam gerakan gestur tangan 39
Tabel 4.20 Menjelaskan akuisisi nilai sensor 39
Tabel 4.21 Menjelaskan proses filter sinyal sensor MPU-6050 40
Tabel 4.22 Menjelaskan konversi nilai sensor menjadi derajat 40
Tabel 4.23 Menjelaskan konversi nilai sensor menjadi PWM 41
Tabel 4.24 Menjelaskan proses pengiriman nilai PWM melalui NRF24L01 41
Tabel 4.25 Menjelaskan proses penerimaan nilai PWM melalui NRF24L01 42
Tabel 4.26 Menjelaskan kebutuhan navigasi robot 42
Tabel 4.27 Menjelaskan Indikator navigasi robot 43
Tabel 4.28 Menjelaskan kebutuhan karakteristik pengguna 43

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Tabel 4.29 Menjelaskan kebutuhan lingkungan operasi 44
Tabel 4.30 Menjelaskan ketergantungan serta asumsi sistem dan pengguna 45
Tabel 4.31 Menjelaskan batasan perancangan dan implementasi 45
Tabel 5.1 Skema pengkabelan perangkat akuisisi sensor 52
Tabel 5.2 Skema pengkabelan perangkat *transmitter* 54
Tabel 5.3 Skema pengkabelan perangkat *receiver* 58
Tabel 5.4 Indikator nyala lampu LED 65
Tabel 5.5 Klasifikasi gerak tangan 65
Tabel 5.6 Klasifikasi gerak Robot 67
Tabel 5.7 Kode program konfigurasi komunikasi dengan protokol I2c 73
Tabel 5.8 Kode program akuisisi nilai sensor 75
Tabel 5.9 Kode program proses konversi nilai 78
Tabel 5.10 Kode program Implementasi nilai pwm 80
Tabel 5.11 *Step 1 Time update* 82
Tabel 5.12 *Step 2 Update estimation error covariance* 82
Tabel 5.13 *Step 3 Calculation angle and bias* 82
Tabel 5.14 *Step 4 Measurement update estimation error* 83
Tabel 5.15 *Step 5 Calculate Kalman gain* 83
Tabel 5.16 *Step 6 Update Posteriori Estimate* 84
Tabel 5.17 *Step 7 Update posteriori error covariance matrix* 84
Tabel 5.18 Kode program proses *kalman filter* 85
Tabel 5.19 *Library* perangkat TX 86
Tabel 5.20 Fungsi *library* perangkat TX 87
Tabel 5.21 Kode program mengirim nilai dari TX menuju RX 87
Tabel 5.22 Kode program menerima nilai dari perangkat TX 89
Tabel 5.23 Indikator maju 90
Tabel 5.24 Indikator ke kanan 90
Tabel 5.25 Indikator ke kiri 91
Tabel 5.26 Indikator mundur 92
Tabel 5.27 Implementasi gerak tangan 93
Tabel 5.28 Implementasi gerak motor 94
Tabel 6.1 Perbandingan lima data sampel sudut *pitch* k = 0.93 100

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Tabel 6.2 Perbandingan lima data sampel sudut *roll* k = 0.93 101
Tabel 6.3 Perbandingan lima data sampel sudut *pitch* k = 0.94 103
Tabel 6.4 Perbandingan lima data sampel sudut *roll* k = 0.94 104
Tabel 6.5 Perbandingan lima data sampel sudut *pitch* k = 0.95 106
Tabel 6.6 Perbandingan lima data sampel sudut *roll* k = 0.95 107
Tabel 6.7 Perbandingan lima data sampel sudut *pitch* k = 0.96 109
Tabel 6.8 Perbandingan lima data sampel sudut *roll* k = 0.96 110
Tabel 6.9 Perbandingan lima data sampel sudut *pitch* k = 0.97 112
Tabel 6.10 Perbandingan lima data sampel sudut *roll* k = 0.97 113
Tabel 6.11 Perbandingan selisih sudut acuan kelima konstanta 114
Tabel 6.12 Perbandingan 10 data sampel sudut *pitch* skenario 1 116
Tabel 6.13 Perbandingan 10 data sampel sudut *roll* skenario 1 118
Tabel 6.14 Perbandingan 10 data sampel sudut *pitch* skenario 2 119
Tabel 6.15 Perbandingan 10 data sampel sudut *roll* skenario 2 121
Tabel 6.16 Perbandingan 10 data sampel sudut *pitch* skenario 3 122
Tabel 6.17 Perbandingan 10 data sampel sudut *roll* skenario 3 124
Tabel 6.18 Perbandingan standar deviasi 125
Tabel 6.19 Keberhasilan mendeteksi gestur 131
Tabel 6.20 Perbandingan kecepatan navigasi robot maju 135
Tabel 6.21 Perbandingan kecepatan navigasi robot mundur 135
Tabel 6.22 Perbandingan kecepatan manuver ke kiri 136
Tabel 6.23 Perbandingan kecepatan manuver ke kanan 136
Tabel 6.24 Hasil uji *delay* hubungan antar perangkat 140
Tabel 6.25 Hasil uji *response time* pada masing-masing gestur 141

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

DAFTAR GAMBAR

Gambar 2.1 Gambaran platform penstabil 6
Gambar 2.2 Filter *Covariance* pengukuran dalam keadaan optimal 9
Gambar 2.3 Pengujian gestur tangan 11
Gambar 2.4 Platform robot beroda 4WD 13
Gambar 2.5 *Robotic Symbionts* 15
Gambar 2.6 *Kalman filter* pada Radar 16
Gambar 2.7 Piezo Electric *Accelerometer* 17
Gambar 2.8 Sensor IMU MPU-6050 17
Gambar 2.9 Arduino IDE dan Serial Monitor 18
Gambar 2.10 *Motor driver* L298N 19
Gambar 2.11 Putaran Motor DC 20
Gambar 2.12 Frekuensi dan *adressing* pada NRF24L01 21
Gambar 3.1 Diagram alir metodologi penelitian 22
Gambar 4.1 Diagram rekayasa kebutuhan 27
Gambar 4.2 Diagram blok sistem 29
Gambar 5.1 Diagram alir perancangan sistem 47
Gambar 5.2 Desain *gloves* 48
Gambar 5.3 Desain penampang Robot beroda 49
Gambar 5.4 Desain Robot beroda (tampak atas) 50
Gambar 5.5 Desain Robot beroda (tampak bawah) 51
Gambar 5.6 Skematik akuisisi sensor dan desain *transmitter* 52
Gambar 5.7 Desain dari perangkat *Transmitter* 53
Gambar 5.8 Skematik perangkat *Transmitter* 54
Gambar 5.9 Diagram alir perangkat *transmitter* 56
Gambar 5.10 Skematik dan desain perangkat *receiver* 57
Gambar 5.11 Diagram alir perangkat *receiver* 59
Gambar 5.12 Komunikasi searah TX menuju RX modul NRF24L01 64
Gambar 5.13 Indikator pada perangkat TX 64
Gambar 5.14 Alur implementasi pada sistem 69
Gambar 5.15 Implementasi desain *Gloves* 70

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Gambar 5.16 Implementasi Robot beroda (tampak atas) 71
Gambar 5.17 Implementasi desain robot beroda (tampak bawah) 72
Gambar 5.18 Implementasi sensor MPU-6050 dan Arduino uno V3 73
Gambar 5.19 Hasil akuisisi data sensor MPU-6050 77
Gambar 5.20 Implementasi perangkat RX 79
Gambar 5.21 Implementasi modul komunikasi NRF24L01..... 86
Gambar 6.1 Diagram alir pengujian dan analisis 97
Gambar 6.2 Tampilan program PLX-DAQ dan microsoft excel..... 98
Gambar 6.3 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.93..... 99
Gambar 6.4Grafik sudut *pitch* dengan 5 data sampel pada k = 0.93..... 100
Gambar 6.5 Grafik sudut *roll* dengan 100 data sampel pada k = 0.93 101
Gambar 6.6 Grafik sudut *roll* dengan 5 data sampel pada k = 0.93 102
Gambar 6.7 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.94..... 102
Gambar 6.8 Grafik sudut *pitch* dengan 5 data sampel pada k = 0.94..... 103
Gambar 6.9 Grafik sudut *roll* dengan 100 data sampel pada k = 0.94 104
Gambar 6.10 Grafik sudut *roll* dengan 5 data sampel pada k = 0.94 105
Gambar 6.11 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.95..... 105
Gambar 6.12 Grafik sudut *pitch* dengan 5 data sampel pada k = 0.95..... 106
Gambar 6.13 Grafik sudut *roll* dengan 100 data sampel pada k = 0.95 107
Gambar 6.14 Grafik sudut *roll* dengan 5 data sampel pada k = 0.95 108
Gambar 6.15 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.96..... 108
Gambar 6.16Grafik sudut *pitch* dengan 5 data sampel pada k = 0.96..... 109
Gambar 6.17 Grafik sudut *roll* dengan 100 data sampel pada k = 0.96 110
Gambar 6.18 Grafik sudut *roll* dengan 5 data sampel pada k = 0.96 111
Gambar 6.19 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.97..... 111
Gambar 6.20 Grafik sudut *pitch* dengan 5 data sampel pada k = 0.97..... 112
Gambar 6.21 Grafik sudut *roll* dengan 100 data sampel pada k = 0.97 113
Gambar 6.22Grafik sudut *roll* dengan 5 data sampel pada k = 0.97 114
Gambar 6.23 Grafik sudut *pitch* dengan 100 data sampel skenario 1..... 116
Gambar 6.24 Grafik sudut *pitch* dengan 10 data sampel skenario 1..... 117
Gambar 6.25 Grafik sudut *roll* dengan 100 data sampel skenario 1 117
Gambar 6.26 Grafik sudut *roll* dengan 10 data sampel skenario 1 118

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Gambar 6.27 Grafik sudut *pitch* dengan 100 data sampel skenario 2..... 119
Gambar 6.28 Grafik sudut *pitch* dengan 10 data sampel skenario 2..... 120
Gambar 6.29 Grafik sudut *roll* dengan 100 data sampel skenario 2 120
Gambar 6.30 Grafik sudut *roll* dengan 10 data sampel skenario 2 121
Gambar 6.31 Grafik sudut *pitch* dengan 100 data sampel skenario 3..... 122
Gambar 6.32 Grafik sudut *pitch* dengan 10 data sampel skenario 3..... 123
Gambar 6.33 Grafik sudut *roll* dengan 100 data sampel skenario 3 123
Gambar 6.34 Grafik sudut *roll* dengan 10 data sampel skenario 3 124
Gambar 6.35 Gestur maju..... 127
Gambar 6.36 Grafik gestur maju..... 127
Gambar 6.37 Gestur mundur..... 127
Gambar 6.38 Grafik gestur mundur..... 128
Gambar 6.39 Gestur kanan..... 128
Gambar 6.40 Grafik Gestur kanan 129
Gambar 6.41 Gestur kiri..... 129
Gambar 6.42 Grafik gestur kiri..... 129
Gambar 6.43 Grafik gestur maju,mundur,kiri dan kanan..... 130
Gambar 6.44 Pengujian kecepatan navigasi robot maju 133
Gambar 6.45 Pengujian kecepatan navigasi robot mundur 133
Gambar 6.46 Pengujian kecepatan manuver ke kiri..... 134
Gambar 6.47 Pengujian kecepatan manuver ke kanan 134
Gambar 6.48 *Delay* sistem 138
Gambar 6.49 *Response time* gestur maju 138
Gambar 6.50 *Response time* gestur mundur 139
Gambar 6.51 *Response time* gestur kiri 139
Gambar 6.52 *Response time* gestur kanan 140

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

LAMPIRAN A KODE PROGRAM 147
A.1 Kode program I2c 147
A.2 Kode program *gloves* (*Transmitter*) 148
A.3 Kode program robot beroda (*Receiver*) 152
LAMPIRAN B AKUISISI DATA UJI *KALMAN FILTER* 154
B.1 Akuisisi data skenario 1 154
B.2 Akuisisi data skenario 2 156
B.3 Akuisisi data skenario 3 158
LAMPIRAN C HASIL UJI GESTUR TANGAN 161
C.1 Gestur Maju 161
C.2 Gestur Mundur 161
C.3 Gestur Kiri 161
C.4 Gestur Kanan 162

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

BAB 1 PENDAHULUAN

1.1 Latar belakang

Pemanfaatan robot saat ini telah banyak memasuki setiap aspek kehidupan manusia. Robot dapat membantu manusia dalam kehidupan sehari-hari untuk bekerja serta memberikan hiburan. Robot dapat dirancang menggunakan algoritma yang memungkinkan robot bergerak secara otomatis, dengan mempertimbangkan informasi sekitar yang dibaca dari sensor yang ada (Elektro, 2014). Berdasarkan kemampuan bergeraknya robot dibagi menjadi beberapa jenis antara lain robot humanoid, robot beroda, robot berkaki, robot manipulator, *flying* robot dan robot *cyborg*.

Robot banyak digunakan untuk mempermudah pekerjaan manusia. Bahkan untuk tugas berat peran robot sangat dibutuhkan untuk menggantikan peran manusia. Salah satu jenis robot yang banyak mengalami perkembangan adalah robot beroda. Robot ini memiliki kemampuan mobilitas yang tinggi sehingga dapat berpindah dari satu tempat ke tempat yang lain. Dalam dunia industri, robot beroda dapat digunakan untuk memindahkan barang serta dapat digunakan sebagai angkutan para pekerja. Robot beroda juga dapat digunakan dalam situasi berbahaya , seperti untuk meneliti apabila terjadi bencana alam dan peperangan. Tentunya hal ini tidak lepas dari kontrol pengguna dalam mengendalikan navigasi robot beroda tersebut.

Seiring dengan pemanfaatan robot beroda yang semakin meningkat, pengunanya dihadapkan dengan masalah terkait sistem kendali navigasi. Pengontrolan navigasi robot beroda bertujuan untuk dapat mengendalikan pergerakan robot agar dapat berjalan maju, mundur, berbelok ke kiri dan ke kanan. Navigasi merupakan hal terpenting dalam penentuan posisi dan orientasi dari robot. Umumnya sistem kendali navigasi robot dikendalikan meggunakan remot kontrol atau *joystick* dengan media kabel. Oleh karena itu, Saat ini kendali navigasi robot beroda telah dikembangkan dengan menggunakan perintah manusia secara langsung, salah satunya dengan menggunakan gestur tangan.

Manusia pada dasarnya lebih terbiasa berinteraksi menggunakan suara, posisi tubuh dan gestur (gerakan) tangan, penggunaan gestur tangan ini lebih mudah dilakukan karena lebih alami dan interaktif serta proses pembelajaran yang relatif lebih singkat (Sutarno, et al., 2016). Gestur merupakan bentuk komunikasi non-verbal dengan aksi tubuh yang dapat mengkomunikasikan pesan-pesan tertentu, sebagai pengganti wicara atau ucapan. Sistem kendali robot menggunakan gestur tangan terus berkembangan, terutama untuk menggantikan peran perangkat masukan seperti *mouse, keyboard dan joystick* yang digunakan untuk interaksi manusia dengan komputer. Namun, perangkat ini memiliki kelemahan diantaranya tidak alami bagi manusia dan jarak pengontrolan masih cukup dekat (Christina, et al., 2015). Gestur tangan menerapkan cara kerja sistem penglihatan dan pengenalan oleh mata manusia sebagai sensor yang merekam objek berupa

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

gestur tangan dan kemudian dapat mengirim informasi ke otak untuk diproses sehingga menghasilkan sebuah keluaran (Takeshi, et al., 2018).

Perangkat gestur tangan dikembangkan dengan memanfaatkan sebuah sensor *inertial measurement unit* (IMU) sebagai pengukur kecepatan,orientasi dan gaya gravitasi menggunakan kombinasi *accelerometer* dan *gyroscope.* Sensor IMU diharapkan mampu mendeteksi segala bentuk gerakan yang dihasilkan. Namun, hasil pengukurannya seringkali tidak akurat karena beberapa faktor diantaranya *bias errors, scale factor errors* dan *time-correlated noise* (Martin, 2017). Ketidakakuratan tersebut tentunya dapat mempengaruhi kinerja pada perangkat gestur tangan. Untuk mengatasi hal tersebut perlu menerapkan sebuah algoritma penyaringan sinyal. Algoritma penyaringan *Kalman filter* merupakan pendeketan standar untuk mengoptimalkan estimasi ketidakakuratan sinyal yang dihasilkan oleh sensor (Qiang, et al., 2015).

Kalman filter merupakan algoritma yang menggunakan serangkaian data yang diamati dari waktu ke waktu, yang mengandung *noise* dan ketidakakuratan lainnya, untuk memperkirakan variabel yang tidak diketahui (Linier) dengan meningkatkan keakuratan (Qiang, et al., 2015). Filter Kalman dapat dikatakan sebagai filter terbaik yang dapat digunakan, bahkan dari sudut pandang teoretis, karena filter ini meminimalkan kesalahan dari nilai sinyal yang sebenarnya. Proses filter harus dapat menghitung koefisien *matrix*, kesalahan berbasis proses, kesalahan pengukuran serta dapat diterapkan secara *real-time*, cepat, efisien, dan kuat serta anti-interferensi. *Kalman filter* telah banyak digunakan di bidang perhitungan orbit, pelacakan target dan navigasi, seperti pelacakan manuver target dan posisi GPS. Lebih jauh lagi, *Kalman filter* juga memainkan peran penting dalam fusi data sensor, integrasi navigasi dan pemosisian dinamis. Sehingga data sensor yang dihasilkan untuk kendali navigasi robot beroda jauh lebih optimal dan akurat.

Penelitian ini mengimplementasikan sistem kendali gestur tangan dengan algoritma *kalman filter* untuk pengolahan sinyal data pada perangkat kontrol gerak robot beroda dengan media komunikasi *wireless*. Kontrol navigasi robot dilakukan berdasarkan gestur tangan manusia. Sebuah sarung tangan berbasis *wearable device* yang dilengkapi dengan sensor IMU dibangun untuk dapat mendeteksi setiap bentuk gerakan tangan. Selanjutnya, hasil pendeteksian gerakan tangan ini dipetakan dalam beberapa bagian pengontrolan, agar sesuai dengan pola pengontrolan navigasi robot beroda. Kontrol navigasi robot beroda tidak terbatas hanya bermanuver ke arah tertentu namun pada penelitian ini kontrol kecepatan robot beroda juga dapat dikendalikan berdasarkan gestur yang dihasilkan. *Delay* serta *response time* yang dibutuhkan untuk mengendalikan robot beroda berdasarkan gestur juga mempengaruhi kinerja dari sistem secara keseluruhan. Hasil penelitian ini diharapkan dapat menunjukan bahwa sebuah sistem pengontrolan navigasi robot berdasarkan gerakan tangan dapat diimplementasikan dan diaplikasikan secara riil, sehingga dapat memberikan pengalaman baru dalam berinteraksi dengan robot.

1.2 Rumusan masalah

Berdasarkan uraian latar belakang di atas, rumusan masalah pada penyusunan penelitian ini adalah sebagai berikut.

1. Bagaimana hasil implementasi metode *kalman filter* untuk meningkatkan akurasi sinyal pada pengendalian robot beroda menggunakan gestur tangan?
2. Bagaimana hasil kinerja perangkat pengendali berdasarkan gestur tangan yang dihasilkan?
3. Bagaimana tingkat ketepatan navigasi saat robot beroda dikendalikan dengan *input* sudut yang berbeda-beda pada perangkat pengendali ?
4. Berapa *delay* dan *response time* yang didapatkan pada saat proses pengendalian navigasi robot beroda ?

1.3 Tujuan

Berdasarkan rumusan masalah di atas, tujuan dari penyusunan penelitian ini adalah sebagai berikut.

1. Mengetahui hasil implementasi metode *kalman filter* untuk meningkatkan akurasi sinyal pada pengendalian robot beroda menggunakan gestur tangan.
2. Mengetahui hasil kinerja perangkat pengendali berdasarkan gestur tangan yang dihasilkan.
3. Mengetahui tingkat ketepatan navigasi saat robot beroda dikendalikan dengan *input* sudut yang berbeda-beda pada perangkat pengendali.
4. Mengetahui *delay* dan *response time* yang didapatkan pada saat proses pengendalian navigasi robot beroda.

1.4 Manfaat

Manfaat dari penyusunan penelitian Sistem Kendali Navigasi Robot Beroda dengan Gestur Tangan Menggunakan Metode *Kalman filter* dapat diuraikan sebagai berikut :

1. Meningkatkan ilmu pengetahuan dan teknologi dalam pengembangan robot beroda.
2. Dapat mengoperasikan robot beroda dengan gestur tangan.
3. Dapat meningkatkan akurasi sinyal dan meredam gangguan sinyal yang diperoleh dari sensor dengan mengimplementasikan metode *kalman filter*.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

4. Dengan menerapkan sensor *accelerometer* dan *gyroscope* diharapkan mampu mengendalikan robot secara sederhana, fleksibel dan nyaman digunakan.
5. Dapat menganalisis dan menguji solusi alternatif pergerakan dari robot beroda dengan gestur tangan.

1.5 Batasan masalah

Karena keragaman teknologi yang dibutuhkan untuk mengembangkan system maka penelitian ini hanya fokus pada implementasi sensor IMU MPU60-50 dan *kalman filter*. Sistem yang dirancang berupa perangkat pengendali dan robot beroda. Perangkat kendali berupa *wearable device* berbentuk sarung tangan (*Gloves*). Perangkat kendali robot diletakkan pada *gloves* dan terhubung ke robot beroda dengan media *wireless*. Penelitian ini menggunkan mikroprosesor arduino nano V3 untuk proses komputasi. Robot beroda dirancang sebagai objek yang dikendalikan. Robot beroda dirancang dengan menggunakan empat roda sebagai tenaga penggerak. Penggunaan daya dan ketahanan daya pada perangkat tidak termasuk dalam topik penelitian.

1.6 Sistematika pembahasan

Pembahasan dari penelitian Sistem Kendali Navigasi Robot Beroda dengan Gestur Tangan Menggunakan Metode *Kalman filter* yang disusun terdiri dari beberapa bab dan mengikuti sistematika sebagai berikut.

BAB I PENDAHULUAN

Bab Pendahuluan berisi uraian latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika pembahasan.

BAB II LANDASAN KEPUSTAKAAN

Bab Landasan Kepustakaan menjelaskan tinjauan pustaka mengenai penelitian-penelitian terdahulu dan dasar teori yang berhubungan serta menjadi dasar dalam pengembangan Sistem Kendali Navigasi Robot Beroda dengan Gestur Tangan Menggunakan Metode *Kalman filter*.

BAB III METODOLOGI PENELITIAN

Bab Metodologi Penelitian menjelaskan metode dan langkah kerja yang diterapkan dalam penelitian serta terdiri dari studi literatur, rekayasa kebutuhan, perancangan sistem, implementasi sistem, pengujian dan penutup.

BAB IV REKAYASA KEBUTUHAN

Bab Rekayasa Kebutuhan berisi gambaran umum sistem dan kebutuhan yang diperlukan pada sistem meliputi kebutuhan pengguna, kebutuhan sistem, kebutuhan fungsional dan kebutuhan non-fungsional.

BAB V PERANCANGAN DAN IMPLEMENTASI

Bab Perancangan dan Implementasi menjelaskan tahap rancangan arsitektur,representasi data dan detil implementasi dari fungsi-fungsi yang menjadi fokus dalam penelitian.

BAB VI PENGUJIAN DAN ANALISIS

Bab Pengujian dan Analisis menjelaskan skenario pengujian dari sistem yang dikembangkan dalam penelitian serta analisis dari hasil pegujian tersebut. Terdiri dari pengujian fungsional dan non-fungsional.

BAB VII PENUTUP

Bab Penutup menjelaskan kesimpulan dari Sistem Kendali Navigasi Robot Beroda dengan Gestur Tangan Menggunakan Metode *Kalman filter* dan saran yang terkait mengenai hal-hal yang dapat dilakukan pada penelitian selanjutnya.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

BAB 2 LANDASAN KEPUSTAKAAN

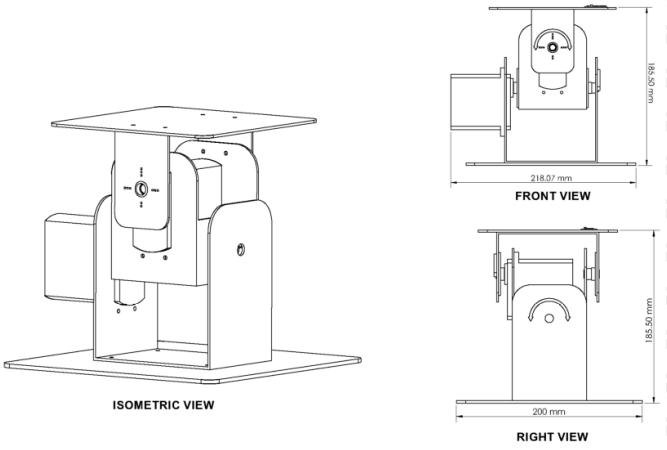
Pada bab ini berisi tinjauan pustaka dan dasar teori yang diperlukan dalam melakukan penelitian. Tinjauan pustaka akan membahas penelitian terdahulu dan berkaitan dengan penelitian yang diusulkan, sedangkan dasar teori akan membahas berbagai teori yang berkaitan dengan topik dalam menyusun penelitian.

2.1 Tinjauan Pustaka

Dalam penelitian diperlukan dukungan hasil-hasil penelitian terdahulu yang berkaitan dengan topik penelitian tersebut. Tinjauan pustaka juga menguraikan perbedaan atau kontribusi penelitian terhadap penelitian-penelitian yang menjadi dasar rujukan. Berikut merupakan beberapa penelitian yan berkaitan dengan topik bahasan penelitian Sistem Kendali Navigasi Robot Beroda dengan Gestur Tangan Menggunakan Metode *Kalman filter*.

2.1.1 *Embedded Sensor Fusion and Moving-average Filter for Inertial Measurement Unit (IMU) on the Microcontroller-based Stabilized*

Sensor *Fusion* (Komplementer dan Kalman filter) dan filter *Moving Average* diimplementasikan pada Arduino mikrokontroler berbasis akuisisi data berdasarkan rotasi (derajat sudut) dari sensor Inertial Measurement Unit (IMU) untuk penstabil platform aplikasi (Grace, et al., 2015). Sistem dikembangkan dengan menggunakan dua sensor IMU yakni sensor LSM303DLHC 3-*axis accelerometer* dan sensor L3GD20 3-*axis gyro*, yang digabungkan menjadi satu.



Gambar 2.1 Gambaran platform penstabil

Sumber: (Grace, et al., 2015)

Pada Gambar 2.1, Platform yang stabil dirancang untuk memiliki dua rotasi bebas pada *pitch* and *roll*. Platform terbuat dari aluminium dengan dua servo motor GWS S777CG / 6BB terpasang pada bingkai *pitch* dan bingkai *roll*. Setiap motor bekerja sebagai aktuator yang menggerakkan platform untuk mengkompensasi rotasi *pitch* atau *roll*. Untuk menganalisis data yang disediakan

oleh IMU, mentah data perlu dikonversi menjadi satuan sudut. Di inisialisasi, pembacaan sampel mentah rata-rata diambil sebagai mengimbangi kriteria untuk setiap sensor saat sistem dalam keadaan diam atau kondisi statis, karena sensitivitas suhu atau misalignment (ketidaklurusan antara kedua poros).

Tabel 2.1 Tabel perbandingan filter

Sumber: (Grace, et al., 2015)

Filter	Result		
	Overshoot	*Rise time (s)*	*Smoothness*
Complementary Filter $\alpha = 0.95$	14.17 %	0.24	0.18
Complementary Filter $\alpha = 0.5$	41 %	0.2	1.14
Complementary Filter $\alpha = 0.05$	64 %	0.2	2.66
Kalman Filter	107.22 %	0.16	0.29
Moving Average	0 %	0.42	0.21

Berdasarkan Tabel 2.1, hasil pengujian sebuah *Overshoot* tinggi di filter Kalman dihasilkan dari data *accelerometer*. Nilai yang lebih baik untuk varian *accelerometer* diperlukan untuk mendapatkan Kalman yang optimal. Nilai rotasi *pitch* dihasilkan dari komplementer filter menunjukkan *Overshoot* yang lebih kecil daripada filter Kalman. Lebih kecil *Overshoot* penting untuk membuat sistem platform menjadi lebih stabil. *Filter moving average* menghasilkan *Overshoot* 0% tetapi kurang responsif terhadap waktu. Sehingga *complementary filter*, *kalman filter* dan moving-average filter pada setiap sudut sensor diperoleh Sensor *gyroscope* menghasilkan nilai yang halus tetapi cenderung bergeser ke nol sedangkan *accelerometer* meskipun tidak bergeser, nilai yang dihasilkan berfluktuasi. Sehingga dapat dianalisis bahwa berdasarkan hasil terebut pemilihan konstanta pada tabel tersebut akan menjadi acuan dalam proses pengujian filter.

Persamaan dengan skripsi :

1. Implementasi *kalman filter* dan *complementary filter* (menggunakan konstanta 0.95 dan 0.5 untuk proses uji filter)
2. Pemanfaatan mikrokontroler Arduino dan aktuator berupa servo motor DC
3. Akuisisi data berdasarkan rotasi (derajat sudut) dari sensor Inertial Measurement Unit (IMU) serta pemilihan konstanta complementary filter

Perbedaan atau kontribusi skripsi :

1. Metode Implementasi *kalman filter* dan *complementary filter* tanpa menggunakan *filter Moving Average.* Meskipun pada penelitian tersebut presentase *Overshoot* kalman filter cenderung tinggi namun pada penelitian skripsi akan dikembangakan filter yang sesuai diterapkan pada sensor yang berbeda serta sistem yang berbeda pula.

2. Menggunakan sensor IMU MPU-6050 merupakan gabungan dari *Accelerometer* dan *Gyroscope* sehingga lebih efektif.

2.1.2 *Kalman filter and Its Application*

Kalman filter adalah estimasi varians minimum pada sistem yang dinamis untuk memperoleh keadaan optimal (Qiang, et al., 2015). Makalah ini secara singkat menyurvei perkembangan terkini tentang Kalman filter (KF). Persamaan *kalman filter* dikelompokkan menjadi dua bagian yakni, persamaan *update* waktu sebagai persamaan prediksi dan persamaan *update* pengukuran sebagai persamaan koreksi. Siklus kerja *Kalman filter* diskrit adalah *Update* waktu membuat prediksi nilai *state*. *Update* pengukuran, menyesuaikan nilai prediksi dengan nilai ukur aktual. Persamaan spesifik untuk *update* waktu dan pengukuran disertakan dalam persamaan.

Prediction (time update)

Estimation $\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$ (2.1)

Covariance error $P_k^- = AP_{k-1}A^T + Q$ (2.2)

Perhatikan persamaan **(2.1)** dan **(2.2)**, dimana $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^-$ menunjukkan estimasi keadaan apriori (tidak diketahui) dan pada waktu $\boldsymbol{k}$, sedangkan $\boldsymbol{P}_{\boldsymbol{k}}^-$ menunjukkan perhitungan estimasi sebelumnya dari prediksi ketidakpastian atau *covariance* galat (kesalahan) pada waktu $\boldsymbol{k}$. Pada awal algoritma, nilai-nilai $_{\boldsymbol{k-1}}$ untuk $\hat{\boldsymbol{x}}$ dan P berasal dari estimasi awal.

Update (measurement update)

Kalman Gain $K_k = P_k^- C^T (CP_k^- C^T + R)^{-1}$ (2.3)

Estimation $\hat{x}_k = \hat{x}_k^- + K(y_k - C\hat{x}_k^-)$ (2.4)

Covariance error $P_k = (1 - K_k C)P_k^-$ (2.5)

Langkah kedua algoritma pada persamaan (**2.3**),(**2.4**) dan (**2.5**) menggunakan perkiraan apriori yang dihitung dalam langkah prediksi dan memperbaruinya untuk menemukan perkiraan *posterior* tentang state *covariance error*. *Kalman gain* dihitung sedemikian rupa sehingga meminimalkan *covariance error posterior.* Pada langkah berikutnya lakukan update terhadap estimasi sebelumnya secara berulang. Pengukuran inilah yang membuat kalman filter secara umum disebut sebagai *recursos filter* atau filter dengan umpan-balik.

Berdasarkan hasil penelitian karakteristik kalman filter ditunjukkan sebagai berikut:

1. Kalman filter adalah alat untuk mengestimasi variabel yang dapat digunakan untuk memperkirakan status sistem linear.

2. Kalman filter adalah filter estimasi varian minimum. Tapi Kalman Filter juga memiliki keterbatasan yakni hanya dapat diterapkan pada linear sistem, dan membutuhkan noise misalnya *white noise Gaussian*.
3. Namun, hampir semua sistem adalah sistem non-linear dalam prakteknya; Oleh karena itu, penerapan standar Kalman filter terbatas.

Persamaan dengan skripsi :

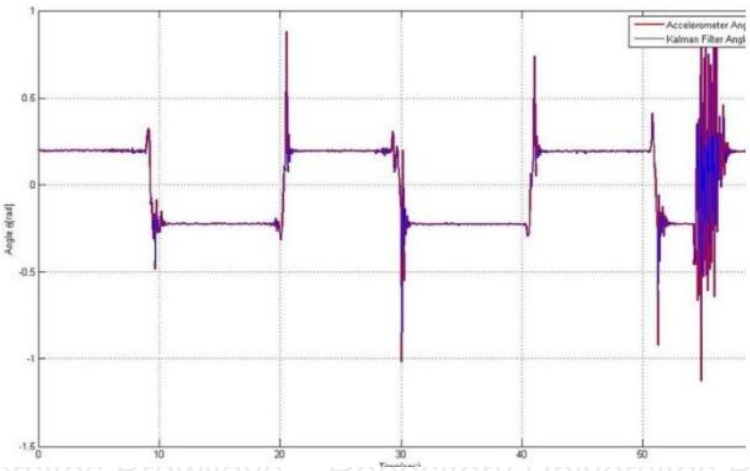
1. Implementasi *kalman filter* untuk penerapan pada linier sistem
2. Implementasi persamaan kalman *prediction* dan *update*

Perbedaan atau kontribusi skripsi :

1. Metode Implementasi *kalman filter* diterapkan pada sensor IMU sedangkan pada makalah merupakan konsep dasar algoritma kalman filter
2. Tentunya penerapan pada sensor akan mengalami pengembangan persamaan kalman *prediction* dan *update.*

2.1.3 Uji Performansi *Ensemble Kalman Filter* untuk Mengurangi *Noise* Pengukuran Sensor pada Robot

Kontrol PID dan Algoritma filter Kalman menjadi implementasi pengujian ideal model robot untuk mampu meningkatkan performa kontrol pada sistem (Prasetio & Kurniawan, 2015). Penelitian ini menggunakan *Ensemble Kalman Filter (EnKF)* dan *Proportional–Integral–Derivative controller (PID) controller* sebagai algoritma penyeimbang robot. Dilakukan *tunning* manual pada *covariance filter*. Percobaan dilakukan dengan metode *trial and error* dengan mengubah-ubah *Matrix Covariance noise process*. *Overshoot* sistem bisa dikurangi dengan cara mengatur *Matrix Covariance noise process*. Dari hasil percobaan sistem optimal pada Q_*accelerometer* : 0.001, Q_*gyroscope* : 0.05, R_pengukuran : 0.03.



Gambar 2.2 Filter *Covariance* pengukuran dalam keadaan optimal

Sumber: (Prasetio & Kurniawan, 2015)

Hasil pengukuran pada Gambar 2.2 menunjukkan bahwa solusi permasalahan *Overshoot*, dapat diselesaikan dengan menyesuaikan *Matrix* parameter *Covariance noise process* Q_*gyroscope*. Dari hasil percobaan nilai *matrix covariance process* optimal pada: Q_*accelerometer* : 0.001 Q_*gyroscope* : 0.05 serta nilai *Covariance noise* pengukuran Rv optimal pada 0.03. Namun metode *trial and error* membutuhkan banyak waktu serta percobaan tergantung objek penelitian. Oleh karena itu hasil *tunning* manual tidak selalu sama pada setiap perangkat yang diuji. Dalam hal ini, perangkat yang diujikan adalah sensor *gyroscope* ITG 3200 dan *accelerometer* ADXL 345.

Persamaan dengan skripsi :

1. Pengembangan sistem pengelanan perangkat robot dan konsep kendali menggunakan sensor IMU
2. Implementasi *kalman filter* untuk penerapan pada sistem menggunakan mekanisme percobaan *trial and error*
3. Implementasi persamaan kalman *prediction* dan *update*

Perbedaan atau kontribusi skripsi :

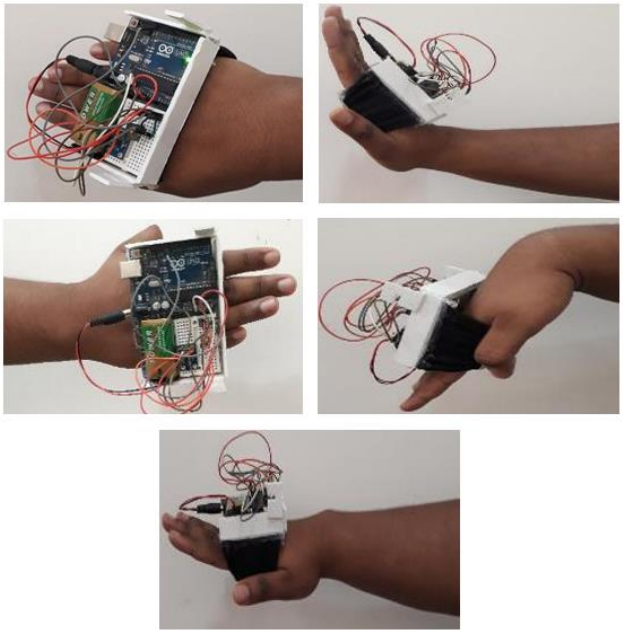
1. Pemanfaatan sensor IMU MPU-6050 yang menggabungkan fungsi *gyroscope* dan *accelerometer* sehingga penggunaan komponen menjadi lebih efektif dan efisien
2. Metode Implementasi *kalman filter* diterapkan pada sensor IMU tidak menggunakan PID
3. *Tunning Covariance noise* berbeda sesuai dengan objek penelitian dan kondisi sinyal keluaran.

2.1.4 *A simple and effective way of controlling a robot by hand gesture*

Desain dan implementasi kontrol robot yang bergerak berdasarkan gerakan tangan menghilangkan kerumitan menggunakan *joystick* atau sakelar (Hasan, et al., 2018). Kecepatan robot juga dikendalikan selama gerakan oleh gestur tangan. *Accelerometer* digunakan untuk mengubah perbedaan gestur tangan menjadi sinyal data. Robot terdiri dari dua bagian: perangkat transmisi dan perangkat penerima. Perangkat transmisi mentransmisikan sinyal gerakan dan perangkat penerima menerimanya.

Berdasarkan hasil pengujian diperoleh konsep mengendalikan kecepatan menggunakan gestur tangan sehingga membuat robot yang dirancang serbaguna dan mudah bermanuver. Namun karena hanya menerapkan sensor *accelerometer* untuk membaca akselerasi maka kurang efektif dalam memprediksi sudut yang dihasilkan. Serta sinyal yang dihasilkan oleh sensor tidak selalu stabil. Oleh karena itu perlunya memadukan kombinasi *gyroscope* untuk medeteksi kemiringan ke

dalam sistem serta menggunakan algoritma filter pada sistem untuk meredam *noise* yang ditimbulkan oleh sensor.



Gambar 2.3 Pengujian gestur tangan

Sumber: (Hasan, et al., 2018)

Berdasarkan hasil pengujian diperoleh gestur gerakan seperti yang ditunjukkan oleh Gambar 2.3 :

1. **Saat tangan sejajar dengan tanah**

   Ketika tangan tetap sejajar dengan tanah, robot akan tetap bergeming. Jika ada gerakan dalam satu dari empat arah tangan dibuat, maka robot dimulai bergerak ke arah itu.

2. **Saat tangan dimiringkan keatas**

   Saat tangan dimiringkan ke atas secara bertahap, robot bergerak maju dan juga kecepatannya meningkat bertahap. Saat tangan dimiringkan mendekati 90 derajat, robot bergerak dengan kecepatan maksimumnya. Jika tangan kembali ke bawah posisi sebelumnya, kecepatan menurun dan secara bertahap menjadi nol.

3. **Saat tangan dimiringkan kebawah**

   Ketika tangan dimiringkan ke bawah secara bertahap, robot sekarang mulai bergerak ke arah belakang dan juga kecepatannya atau rpm meningkat secara bertahap. Saat tangan dimiringkan mendekati 90 derajat dan pergelangan tangan dan jari-jari sekarang menunjuk ke bawah, robot bergerak dengan kecepatan tertingginya. Begitu pula saat di tangan membalikkan kembali ke posisi stasioner, kecepatan terus menurun akhirnya dan menjadi nol.

4. Saat tangan dimiringkan kekanan

Ketika tangan dimiringkan ke kanan, maka robot bergerak ke arah kanan. Ketika tangan dimiringkan ke belakang posisi stasioner, robot menghentikan gerakannya. Pada dasarnya, ketika tangan dimiringkan ke kanan, salah satu roda robot berhenti bergerak dan yang lainnya berputar dalam lingkaran gerakan.

5. Saat tangan dimiringkan kekiri

Ketika tangan dimiringkan ke kiri, maka robot bergerak ke arah kiri. Ketika tangan dimiringkan ke belakang posisi stasioner, robot menghentikan gerakannya. Pada dasarnya, saat tangan dimiringkan ke kiri, salah satu roda robot berhenti bergerak dan yang lainnya berputar dalam gerakan melingkar mirip dengan arah kanan.

Persamaan dengan skripsi :

1. Gerakan gestur tangan yang digunakan
2. Terdiri dari perangkat transmisi dan penerima
3. Perangkat transmisi merupakan perangkat kendali berupa *wearable device* diletakkan di telapak tangan atas serta perangkat robot beroda sebagai perangkat penerima
4. Gerakan gestur tangan juga mempengaruhi kecepatan robot
5. Komponen yang digunakan beberapa ada yang sama misal, arduino dan *motor driver* yang digunakan.

Perbedaan atau kontribusi skripsi :

1. Menggunakan sensor IMU MPU-6050 untuk mendeteksi gestur tangan yang dihasilkan
2. Implementasi kombinasi *gyroscope* dan *accelerometer* yang terdapat pada sensor sehingga pembacaan sudut dan akselarasi menjadi lebih optimal
3. Pemanfaatan Metode *kalman filter* untuk meredam *noise* yang ditimbulkan oleh sensor sehingga sinyal data yang dihasilkan akan lebih akurat

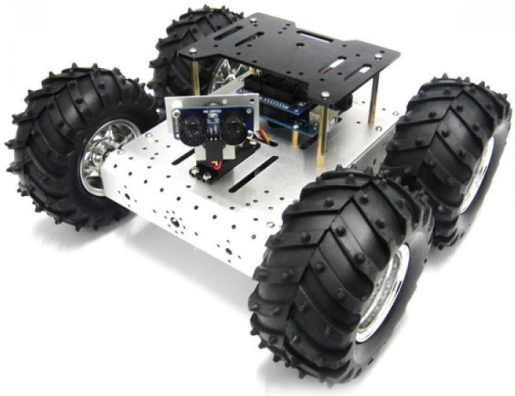
Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

2.2 Landasan Teori

Dasar teori akan membahas berbagai teori yang diperlukan dalam menyusun penelitian yang diusulkan.

2.2.1 Robot beroda

Robot beroda adalah sistem otomatis yang mampu bergerak. Robot beroda memiliki kemampuan untuk bergerak di lingkungan mereka dan tidak terikat pada satu lokasi fisik. Robot beroda memiliki ciri khas yakni roda, trek, kaki dan kombinasi dari beberapa aktuator tersebut (Assunta, et al., 2015). Gambar 2.4 memperlihatkan bentuk dari Robot beroda yang dilengkapi dengan roda.



Gambar 2.4 Platform robot beroda 4WD

Sumber: (ALSrobot, 2016)

Komponen-komponen Robot beroda adalah pengontrol, perangkat lunak kontrol, sensor dan aktuator. Perangkat pengontrol pada umumnya adalah mikroprosesor, mikrokontroler tertanam atau komputer pribadi (PC). Perangkat lunak kontrol seluler dapat berupa bahasa tingkat perakitan atau bahasa tingkat tinggi seperti C, C ++, Pascal, Fortran, atau *realtime software*. Sensor yang digunakan tergantung pada persyaratan robot. Persyaratannya dapat berupa sentuhan dan penginderaan jarak, triangulasi , penghindaran tabrakan, lokasi posisi dan aplikasi spesifik lainnya.

Menurut (Bozinovski, 1994) Berdasarkan tipe navigasi, robot dibedakan menjadi enam bagian diantaranya adalah :

1. Manual Remote or Tele-op

Secara manual sepenuhnya dikontrol oleh *driver* dengan *joystick* atau perangkat kontrol lainnya. Perangkat dapat dicolokkan langsung ke robot, misalnya *joystick* nirkabel, atau aksesori untuk komputer nirkabel atau pengontrol lainnya. Robot tele-op biasanya digunakan untuk menjauhkan operator dari bahaya. Contoh robot jarak jauh manual termasuk ANAT*ROLL*ER ARI-100 dan ARI-50 dari *Robotics Design*, Talon Foster-Miller, PackBot iRobot, dan KumoTek MK-705 Roosterbot.

2. *Guarded Tele-op*

Robot Tele-op Guarded memiliki kemampuan untuk merasakan dan menghindari rintangan, tetapi sebaliknya dari Robot *Remote Tele-op* akan bernavigasi dipandu oleh operator dan tidak *bias* dikendalikan jarak jauh maupun secara nirkabel.

3. *Line Following Car*

Beberapa dari *Automated Guided Vehicles* (AGV) yang pertama adalah Robot beroda mengikuti garis visual yang dicat atau tertanam di lantai, langit-langit atau kabel listrik di lantai. Sebagian besar robot ini mengoperasikan algoritma "menjaga garis dalam sensor pusat".

4. *Autonomously Randomized Robot*

Robot otonom dengan gerakan acak pada dasarnya mendeteksi lingkungan atau halangan dengan menggunakan sensor-sensor tertentu.

5. *Autonomously Guided Robot*

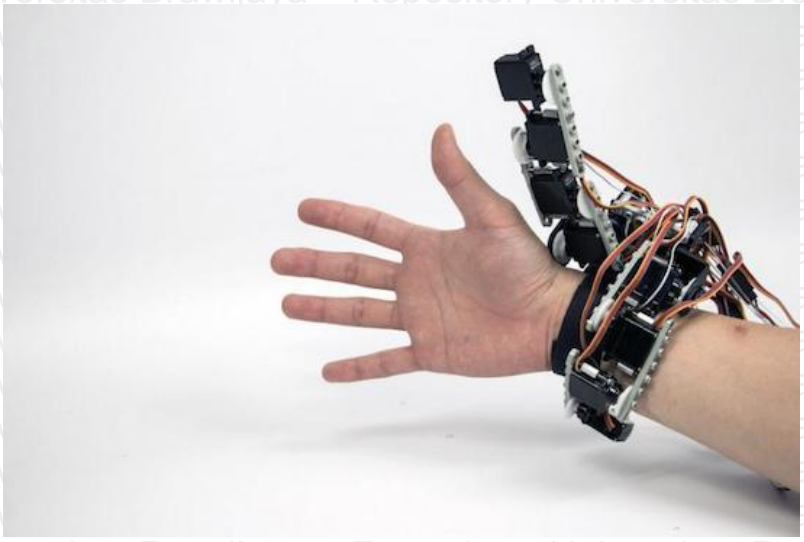
Robot yang dipandu secara otonom mengetahui setidaknya beberapa informasi tentang di mana dan bagaimana mencapai berbagai tujuan atau titik disepanjang perjalanan. Pengetahuan tentang lokasinya saat ini, dihitung dengan satu atau lebih sarana, menggunakan sensor seperti pembuat enkode motor, visi, Stereopsis, laser dan sistem penentuan *global posisition*. Sistem posisi sering menggunakan triangulasi, posisi relatif dan / atau Monte-Carlo / Markov lokalisasi untuk menentukan lokasi dan orientasi platform, dari mana ia dapat merencanakan jalur ke titik arah atau sasaran berikutnya.

6. *Sliding Autonomy*

Robot yang mampu menggabungkan beberapa tingkat navigasi di bawah sistem yang disebut *Sliding Autonomy*. Kebanyakan robot yang dipandu secara otonom, seperti robot rumah sakit *HelpMate*, juga menawarkan mode manual. Sistem operasi robot otonom *Motivity*, yang digunakan dalam ADAM, *PatrolBot*, *SpeciMinder, MapperBot* dan sejumlah robot lainnya, menawarkan *Sliding Autonomy* penuh, dari mode manual hingga dipandu secara otonom.

2.2.2 *Wereable Device*

Wearable Device adalah sebuah perangkat operasi komputer dan teknologi yang dapat digunakan pada bagian tubuh manusia (Techopedia, 2016). Perangkat tersebut juga dapat diimplementasikan dalam kehidupan sehari-hari berdasarkan fungsi dan kegunaannya masing-masing. Dalam segi ergonomis *weareble device* juga banyak sekali manfaatnya diantaranya adalah mudah dipakai dan nyaman. Adaptasi pengguna dan akses ke perangkat jauh lebih mudah.



Gambar 2.5 *Robotic Symbionts*

Sumber: (Glass, n.d.)

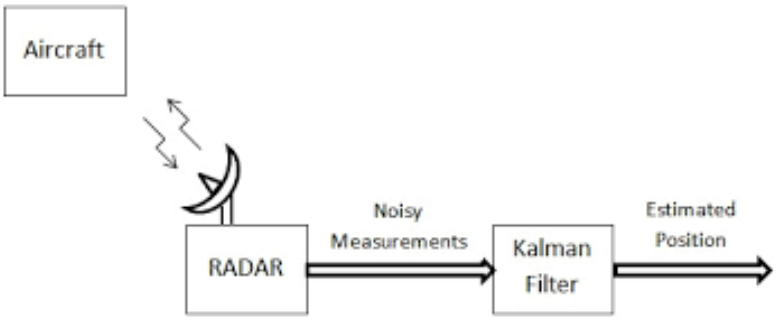
Pada gambar 2.5 merupakan *Robotic Symbionts* salah satu contoh penerapan *wearable device* pada tangan manusia. Mengembangkan sendi robot yang dapat diprogram yang dikenakan di pergelangan tangan untuk secara instan memberi kelingking ekstra, ibu jari ketiga, atau bahkan tangan lain. Perangkat seperti *cyborg* ini terdiri dari 11 motor yang dapat mendeteksi sinyal otak yang dikirim ke otot *brachioradialis* lengan bawah dan mengatur ulang diri agar sesuai dengan tugas yang berbeda. Begitu pula pada penelitian ini menerapkan konsep *wearable device* yang mana berbentuk *gloves* atau sarung tangan yang terdapat perangkat komputer untuk mengendalikan gerakan robot beroda berdasarkan gestur tangan.

2.2.3 *Kalman filter*

Kalman filter atau juga dikenal sebagai *Linear Quadratic Estimation* (LQE) (Mohinder, et al., 2001), merupakan algoritma yang menggunakan serangkaian pengukuran yang diamati dari waktu ke waktu, yang mengandung *noise* dan ketidakakuratan lainnya. Menghasilkan perkiraan variabel yang cenderung lebih akurat daripada yang didasarkan pada pengukuran tunggal saja , dengan memperkirakan distribusi probabilitas gabungan atas variabel untuk setiap jangka waktu (Paul & Howard, 2000).

Filter Kalman memiliki banyak aplikasi dalam teknologi. Aplikasi yang umum adalah untuk panduan, navigasi, dan kontrol kendaraan, khususnya pesawat terbang dan pesawat ruang angkasa (Paul & Howard, 2000). Selanjutnya, filter Kalman adalah konsep yang banyak digunakan dalam analisis deret waktu yang digunakan dalam bidang seperti pemrosesan sinyal dan ekonometrik. Filter Kalman juga merupakan salah satu topik utama di bidang perencanaan dan kontrol gerak robot, dan terkadang dimasukkan dalam optimalisasi lintasan. Filter Kalman juga berfungsi untuk memodelkan kontrol gerakan sistem saraf pusat. Karena

waktu tunda antara mengeluarkan perintah motor dan menerima umpan balik sensorik, penggunaan filter Kalman mendukung model realistis untuk membuat perkiraan status sistem motor saat ini dan mengeluarkan perintah yang diperbarui. Salah satu contohnya adalah implementasi *Kalman filter* pada Radar dibawah ini untuk memperkirakan posisi pesawat terbang. Berikut gambar 2.6 merupakan gambaran *kalman filter* pada radar



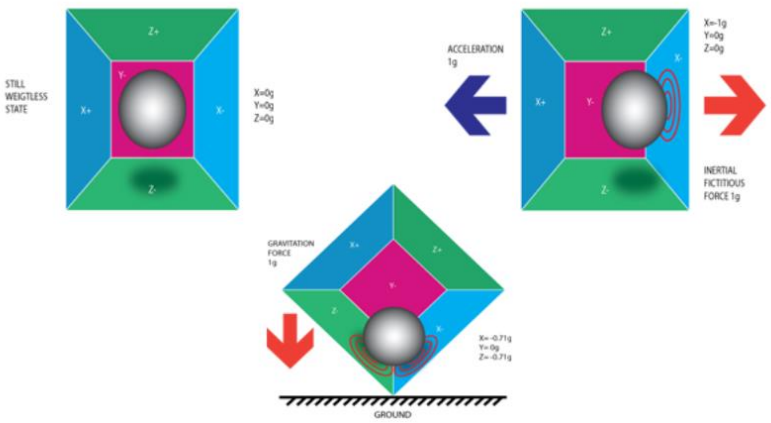
Gambar 2.6 *Kalman filter* pada Radar

Sumber: (Mathwork, 2018)

2.2.4 *Inertial Measurement Unit*

Inertial Measurement Unit (IMU)) merupakan modul eletronik yang dapat mengukur percepatan akselerasi linier dan *angular* dengan koordinat percepatan atau laju perubahan kecepatan (James, 2008). IMU sendiri terdiri dari kombinasi *Gyroscope* dan *Accelerometer*. *Accelerometer* adalah jenis perangkat sensor yang membaca percepatan dan menyediakan data saat bergerak di sepanjang salah satu dari sumbu 3-nya yaitu X, Y, Z (Mojeebi & Tulo, 2016).

Cara kerja *acce*lerometer berdasarkan prinsip kerja pada piezoelektrik. Terdiri dari kotak berbentuk kubus dengan bola di dalamnya, seperti pada Gambar 2.7. Dinding kotak ini terbentuk dari kristal piezoelektrik yang mana pada saat dimiringkan, bola dipaksa untuk bergerak ke arah kemiringan sebagai akibat dari gravitasi. Dinding yang berbenturan dengan bola tersebut menghasilkan arus piezoelektrik. Terdiri dari tiga pasang dinding yang saling berseberangan dalam satu ruang kubus. Setiap pasang dari dinding mewakili sesuai dengan sumbu dalam ruangan tersebut yakni sumbu X, Y, dan sumbu Z. Berdasarkan arus yang dihasilkan akibat benturan dari dinding piezoelektrik sehingga dapat menentukan arah kemiringan dan besarnya arus yan dihasilkan.



Gambar 2.7 Piezo Electric *Accelerometer*

Sumber : (Pro, 2018)

Cara kerja dari *gyroscope* sama dengan *accelerometer* namun mekanismenya seperti sebuah roda berputar dengan piringan yang tetap stabil. Mempertahankan orientasi dari ketetapan momentum sudut. Alat ini sering digunakan pada drone, robot dan alat-alat elektronik lainnya. Sebelum digunakan, sensor *gyro* harus dilakukan kalibrasi untuk memperoleh nilai faktor kalibrasi lingkungan. Keluaran dari sensor *gyroscope* berupa kecepatan sudut dari 3 sumbu, diantaranya adalah sumbu x sebagai sudut *phi* (kanan dan kiri), sumbu y sebagai *theta* (atas dan bawah), dan sumbu z nantinya menjadi *psi* (depan dan belakang).

Pada *gyroscope* terdapat sensor untuk menentukan orientasi gerak dengan bertumpu pada roda atau cakram yang berotasi pada sumbu (Lab, 2018). Karena berorientasi pada pada sumbu dan ketetapan sudut maka *gyroscope* dapat memberikan informasi sudut dengan lebih presisi sampai perputaran 360 derajat. Sehingga dapat mendeteksi gerakan dari segala arah serta tidak dipengaruhi oleh gravitasi namun hanya dapat digunakan jika ada *accelerator*.

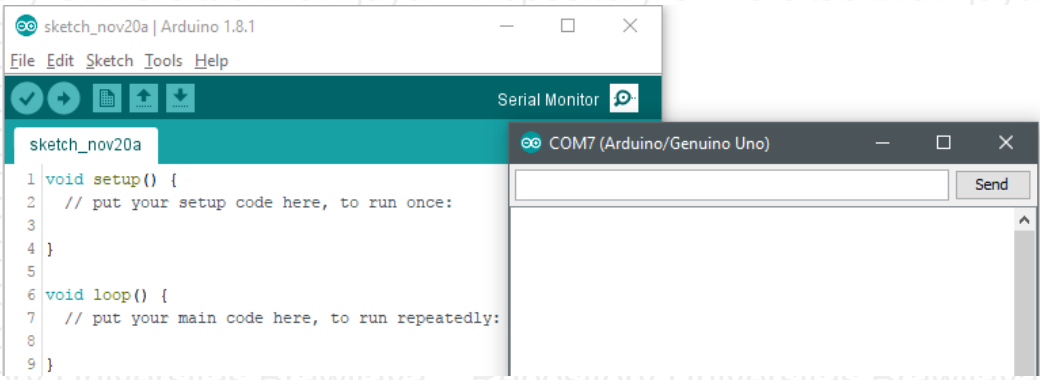
Oleh karena itu kombinasi *accelerometer* dan *gyroscope* sangat diperlukan untuk penelitian ini, salah satu diantaranya sebagai sensor pendeteksi gerakan gestur tangan. Seperti diketahui bahwa sensor yang memiliki kedua fungsi tersebut yakni MPU-6050. Berikut pada Gambar 2.8 merupakan bentuk dari sensor IMU MPU-6050.

Gambar 2.8 Sensor IMU MPU-6050

Sumber : (ElectronicWings, 2018)

2.2.5 Arduino

Arduino adalah platform *open source* yang digunakan untuk membangun proyek elektronik. Menggunakan mikrokontroler Atmega328 sebagai inti pengolahan.Arduino terdiri dari mikrokontroler dan perangkat lunak berupa *Integrated Development Environment (IDE)* yang berjalan di komputer, yang digunakan untuk menulis dan mengunggah kode program, yang disebut sebagai sketch atau sketsa, ke mikrokontroler.



Gambar 2.9 Arduino IDE dan Serial Monitor

Arduino IDE mendukung bahasa C dan C++ menggunakan aturan khusus pada penataan kode. Seperti yang terlihat pada Gambar 2.9, sebuah sketsa Arduino minimal terdiri dari dua fungsi berikut ini.

- ***setup():*** dipanggil sekali setelah *power-up* atau *reset,* digunakan untuk menginisialisasi: variabel, mode pin, dan pustaka lain yang dibutuhkan dalam sketsa.

- ***loop():*** dipanggil setelah fungsi *setup()* selesai dijalankan, dieksekusi berulang-ulang sampai power-off atau reset.

Gambar 2.9 juga memperlihatkan fitur Serial Monitor pada Arduino IDE, yang berfungsi sebagai terminal untuk berkomunikasi dengan mikrokontroler melalui komunikasi serial. Pengembang bisa menggunakan Serial Monitor untuk melakukan *debugging* terhadap sketsa yang sedang dikerjakan.

Dalam penelitian ini penulis menggunakan mikrokontroler arduino nano V3 yang mana memiliki dimensi lebih kecil dan kemampuan setara dengan arduino uno ataupun mega. Berikut pada Tabel 2.2 adalah perbandingan antara arduino nano V3 dengan arduino uno dan mega.

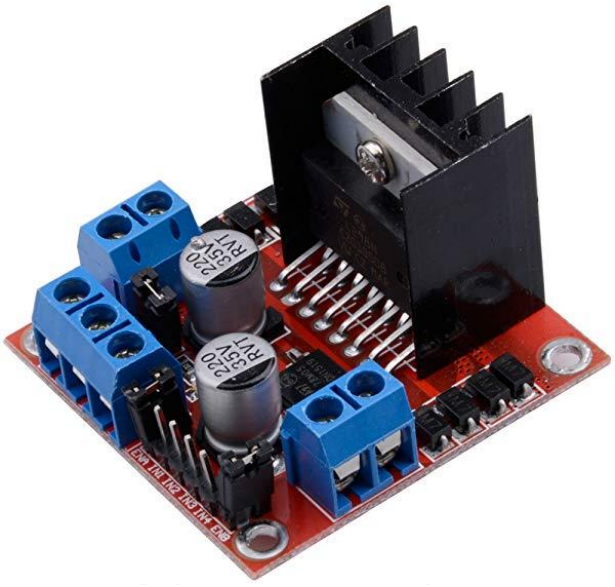
Tabel 2.2 Perbandingan mikrokontroler arduino

Sumber : (Components101, 2018)

Nama	Prosesor	Operasi / Tegangan *Input*	Kecepatan CPU	Analog In/Out	Digital IO/PWM	EEPROM / SRAM [kB]	Flash	USB	USART
Uno	Atmega 382P	5V / 7-12V	16 MHz	6 / 0	14 / 6	1 / 2	32	Regu-lar	1
Nano	Atmega 382P	5V / 7-12V	16 MHz	8 / 0	14 / 6	1 / 2	32	Mini	1
Mega	Atmega 2560	5V / 7-12V	16 MHz	16 / 0	54 / 15	4 / 8	256	Regu-lar	4

2.2.6 *Motor driver* L298D

Motor driver L298N merupakan modul elektronika yang berfungsi untuk mengendalikan arah putaran dan kecepatan motor DC ataupun motor *stepper*. Pemilihan modul *motor driver* L298N menjadi pengendali motor adalah karena tegangan *input* yang bisa ditampung oleh modul sebesar 5V – 12V maksimal dapat mencapai 35V. Sehingga *output* tegangan yang dihasilkan untuk mengendalikan motor DC cukup besar pula. Karena fokus penelitian adalah mengendalikan pergerakan robot beroda maka salah satu indikatornya adalah kecepatan bergerak. Berikut Gambar 2.10 merupakan *motor driver* L298N.



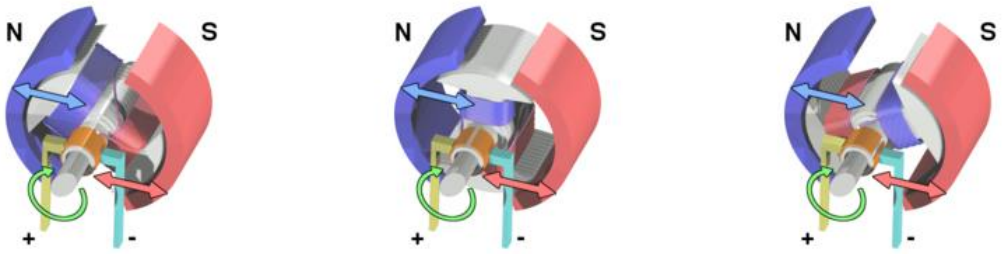
Gambar 2.10 *Motor driver* L298N

Sumber : (Instructable, 2016)

2.2.7 Motor DC *(Battery Operated)*

Pengoperasian Baterai motor DC (BO) adalah mengubah energi listrik menjadi energi mekanik. Konsep DC motor adalah di mana roda gigi mengurangi kecepatan kendaraan tetapi meningkatkan torsi yang dikenal sebagai pengurangan gigi. Di motor DC dirakit dengan pengaturan gigi ganda. Kecepatan motor dihitung dalam hal rotasi per menit disebut rotasi per menit (RPM). Susunan pemasangan membantu meningkatkan torsi dan mengurangi kecepatan motor. Semua robot berbasis mikro-pengontrol motor DC jenis ini dapat digunakan.

Cara kerja Motor DC (BO) Ketika arus melalui *loop* diregangkan pada inti, bagian kutub positif bekerja dengan gaya ke atas, sementara bagian yang lain bekerja dengan gaya ke bawah. Menurut Hukum Tangan Kanan Fleming, pisau listrik menghasilkan efek memutar pada *loop*, menyebabkannya berputar. Untuk memungkinkan motor berputar dalam satu arah, pengubah mode arus terus menerus membalik arah aliran setiap setengah siklus (untuk motor bipolar) dan dengan demikian menyebabkan motor terus berputar ke arah yang sama seperti pada gambar 2.11 dibawah ini.



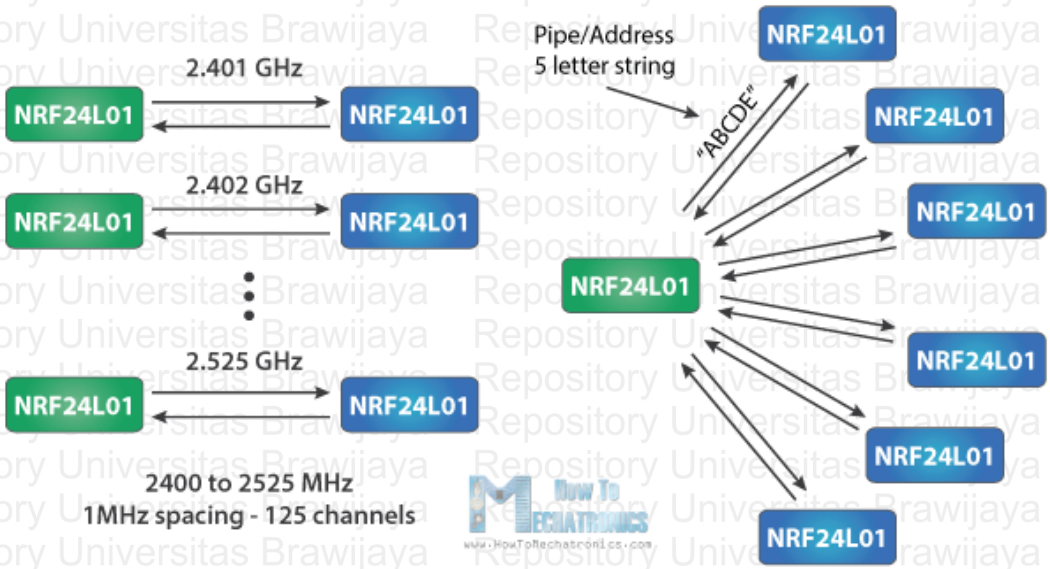
Gambar 2.11 Putaran Motor DC

Sumber : (Article, 2008)

2.2.8 NRF24L01 Module

Modul *transceiver* NRF24L01 merupakan komunikasi jarak jauh dengan menggunakan frekuensi pita gelombang radio sebesar 2,4 – 2.5GHz dan dapat beroperasi dengan kecepatan baud dari 250 kbps hingga 2 Mbps. Jika digunakan di ruang terbuka dan dengan baud rate yang lebih rendah jangkauannya bisa mencapai 100 meter.

Modul ini dapat menggunakan 125 saluran berbeda yang memungkinkan memiliki jaringan 125 modem yang bekerja secara independen di satu tempat. Setiap saluran dapat memiliki hingga 6 alamat, atau setiap unit dapat berkomunikasi hingga 6 unit lainnya secara bersamaan. Seperti pada Gambar 2.12 dibawah ini.



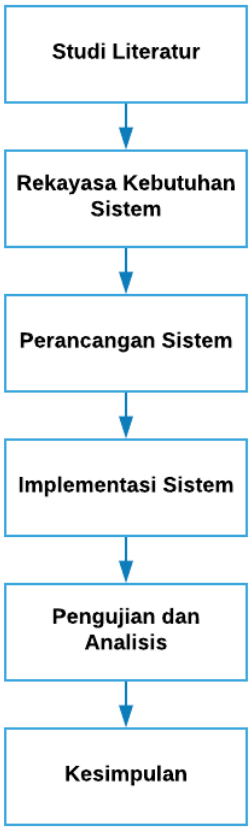
Gambar 2.12 Frekuensi dan *adressing* pada NRF24L01

Sumber : (Dejan, 2017)

Konsumsi daya modul ini hanya sekitar 12mA selama transmisi, yang bahkan lebih rendah dari satu lampu LED. Tegangan pengoperasian modul adalah dari 1,9 hingga 3,6V, tetapi hal baiknya adalah pin yang lain mentolerir logika 5V, sehingga kita dapat dengan mudah menghubungkannya ke Arduino tanpa menggunakan konverter level logika apa pun.

BAB 3 METODOLOGI PENELITIAN

Penelitian yang dilakukan oleh penulis bersifat implementatif dengan strategi atau metode pembuatan artefak teknologi informasi. Artefak teknologi yang diterapkan dalam peneitian ini berupa Sistem Kendali Navigasi Robot Beroda dengan Gestur Tangan Menggunakan Metode Kalman filter. Gambar 3.1 merupakan diagram alir yang memuat metode penelitian yang dimulai dari studi literatur, rekayasa kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis sistem serta diakhiri dengan penarikan kesimpulan.



Gambar 3.1 Diagram alir metodologi penelitian

3.1 Studi Literatur

Tahap ini meliputi studi literatur tentang dasar teori yang mendukung proses penyusunan naskah penelitian. Sumber rujukan yang digunakan berupa sumber refrensi dari buku, makalah penelitian, jurnal prosiding , seminar atau konferensi, dokumen skripsi penelitian terdahulu, atau tulisan lainnya termasuk yang ada di internet. Mengenai: Robot beroda, Gestur tangan, *Kalman filter*, sensor IMU (*Accelerometer* dan *Gyroscope*), dan mikrokontroler arduino.

3.2 Rekayasa Kebutuhan Sistem

Rekayasa kebutuhan sistem akan menjelaskan mengenai gambaran umum sistem yang terkait dengan segala kebutuhan yang diperlukan untuk membangun sistem. Adapun kebutuhan sistem meliputi kebutuhan pengguna, kebutuhan sistem, serta kebutuhan fungsional dan non-fungsional.

1. Kebutuhan pengguna

   Kebutuhan pengguna akan membahas tentang segala kebutuhan yang harus dimiliki pengguna untuk dapat memantau serta mengoreksi sistem dengan baik. Hal ini berhubungan dengan interaksi yang ditimbulkan antara pengguna dengan sistem. Pengguna harus dapat menentukan layanan mana yang harus dipenuhi agar pengguna mendapatkan hasil yang tepat sesuai dengan yang dirancang.

2. Kebutuhan Sistem

   Kebutuhan sistem akan membahas tentang persyaratan konfigurasi yang harus dimiliki suatu sistem agar perangkat keras atau aplikasi perangkat lunak dapat berjalan dengan lancar dan efisien. Kegagalan untuk memenuhi persyaratan ini dapat menyebabkan masalah pemasangan atau masalah kinerja. Kebutuhan sistem terdiri dari kebutuhan perangkat lunak yang digunakan untuk proses penelitian serta fungsi atau jenis perangkat lunak seperti apa yang dibutuhkan. Kebutuhan perangkat keras yang digunakan sebagai komponen menyusun artefak penelitian serta menjelaskan fungsi dan spesifikasi rinci perangkat keras. Serta kebutuhan perangkat komunikasi akan menjelaskan implementasi serta cara kerja modul komunikasi yang digunakan untuk menghubungan perangkat *transmitter* dan *receiver*.

3. Kebutuhan Fungsional

   Kebutuhan fungsional membahas tentang kebutuhan menentukan proses-proses yang berkaitan dengan prinsip kerja sistem mencakup tujuan serta keluaran yang harus sesuai dengan masukkan yang diberikan. Dengan mengacu pada fungsi-fungsi dan karekteristik yang telah disediakan oleh sistem. Kebutuhan fungsional juga bergantung kepada jenis perangkat serta pemilihan perangkat pada kebutuhan sistem untuk menggambarkan layanan apa saja yang bisa ditawarkan sistem terhadap pengguna secara rinci.

4. Kebutuhan Non-fungsional

   Kebutuhan non-fungsional mengacu pada properti perilaku yang dimiliki oleh sistem. Batasan layanan atau fungsi terkait perancangan dan implementasi juga akan dibahas pada tahap ini. Serta karakteristik pengguna yang sesuai untuk sistem serta lingkungan operasi yang mendukung kinerja sistem. Menjelaskan tentang asumsi dan ketergantungan suatu fungsi atau layanan terhadap pemanfaatan sistem secara keseluruhan.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

3.3 Perancangan Sistem

Perancangan sistem merupakan tahap setelah segala kebutuhan sistem telah didefinisikan pada tahap rekayasa kebutuhan sistem. Tahap perancangan membahas tentang rancangan sistem yang akan dibuat dalam bentuk diagram alir, skematik serta desain yang berfungsi untuk memodelkan sistem serta memberikan gambaran yang lengkap terhadap sistem yang akan dikembangkan. Adapun tahapan perancangan sistem yang akan dilakukan adalah sebagai berikut.

1. Perancangan desain perangkat keras

   Desain perangkat keras berfungsi untuk memodelkan sistem sehingga memberikan gambaran terhadap bentuk dan dimensi perangkat yang akan dibangun. Desain perangkat keras meliputi desain *gloves* (perangkat *transmitter*), Desain robot beroda (*perangkat receiver*).

2. Perancangan akuisisi sensor

   Pada tahap ini merupakan perancangan bagaimana sistem dapat mengakuisisi nilai pada sensor.

3. Perancangan pemrosesan

   Merupakan rancangan pengolahan pada sistem dimulai dari *input* sampai dengan *output*. Perancangan pemrosesan dibagai menjadi tiga bagian yakni perancangan *transmitter*, *receiver* dan *kalman filter*.

4. Perancangan komunikasi

   Perancangan komunikasi merupakan perancangan hubungan komunikasi antara perangkat *transmitter* dengan *receiver* guna mengirim data.

5. Perancangan indikator

   Merupakan perancangan indikator yang dibentuk oleh lampu LED sebagai penanda navigasi gerak robot beroda.

6. Perancangan kendali

   Merupakan perancangan kendali hubungan antara gerakan navigasi robot beroda berdasarkan gestur tangan yang dihasilkan. Berfungsi untuk memahami gerakan apa saja yang diperbolehkan untuk mengendalikan navigasi ataupun kecepatan pada robot beroda. Terdiri dari klasifikasi gestur tangan dan klasifikasi gerak robot beroda.

3.4 Implementasi Sistem

Implementasi sistem dilakukan dengan berpedoman pada perancangan yang telah dibuat. Tahap ini meliputi langkah-langkah praktis untuk menerapkan dan merealisasikan hasil rancangan sistem yang telah dibuat. Seperti Implementasi desain perangkat keras, komponen elektonika, skematik serta kode program yang digunakan pada masing-masing perangkat. Sehingga keberadaan

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

sistem dapat memiliki keandalan dalam mencegah kesalahan, kerusakan serta kegagalan proses. Adapun implementasi sistem berdasarkan perancangan sistem adalah sebagai berikut.

1. Implementasi desain perangkat keras

   Dilakukan dengan membangun perangkat *transmitter* (*gloves*) dan receiver (robot beroda) berdasarkan komponen yang telah dirancang pada bab perancangan. Desain yang terbentuk harus disesuaikan dengan desain pada perancangan agar hasil serta fungsi pada masing-masing perangkat dapat berjalan sesuai dengan tujuan penelitian.

2. Implementasi akuisisi sensor

   Dalam tahap implementasi, sistem harus mampu mengakuisisi data dari sensor. Serta mengimplementasikan desain, skematik serta kode program perangkat keras yang digunakan dalam menyusun perangkat akusisi sensor.

3. Implementasi pemrosesan

   Pada implementasi pemrosesan yang terbagi menjadi implementasi perangkat *transmitter*, receiver dan kalman filter harus mampu menjalankan fungsi-fungsi dasar sesuai dengan tahap perancangan sistem.

4. Implementasi komunikasi

   Merupakakan implementasi desain dan skematik perangkat keras dari komponen komunikasi. Pada tahap ini modul komunikasi harus mampu mengirim data dari perangkat *transmitter* menuju perangkat receiver.

5. Implementasi indikator

   Implementasi indikator merupakan penerapan seluruh komponen indikator yang telah disusun pada perancangan indikator. Serta pada tahap ini komponen berhasil menunjukkan indikator pergerakan pada robot beroda berdasarkan nyala padamnya lampu LED.

6. Implementasi kendali

   Klasifikasi gerakan gestur tangan dan gerak robot beroda telah di implementasikan pada tahap implementasi kendali.

3.5 Pengujian dan Analisis

Pengujian dimaksudkan untuk mengetahui keterpenuhan terhadap kebutuhan-kebutuhan serta spesifikasi sistem yang sebelumnya telah didefinisikan, serta bagaimana sistem bisa menjadi solusi dari permasalahan yang dikemukakan di bagian latar belakang. Berdasarkan kebutuhan dan spesifikasi sistem, maka dilakukan beberapa metode pengujian pada sistem. Tahap terakhir pada pengujian yakni menganalisis hasil yang telah diperoleh dari tahap pengujian.

Pada penelitian ini pengujian dibedakan menjadi dua, yakni pengujian fungsional dan pengujian non-fungsional.

1. Pengujian fungsional

   Meliputi pengujian mengenai proses filter, gestur tangan dan navigasi. Pengujian fungsional dilakukan untuk mengetahui apakah hasil pengujian terhadap implementasi dan rancangan sistem sesuai dengan tujuan dan kebutuhan sistem. Secara garis besar tujuan pengujian fungsional adalah untuk mengetahui apakah kendali navigasi robot beroda menggunakan gestur tangan dan metode kalman filter berhasil diterapkan pada sistem. Dengan melakukan pegujian proses filter untuk mengimplementasikan beberapa skenario pengujian diantaranya adalah dengan *tunning* variabel *covariance noise process* kemudian melakukan pengambil sampel data pada masing-masing skenario sehingga dapat dibandingkan antara skenario 1, 2 dan 3 yang cocok diterapkan pada sistem.

2. Pengujian non-fungsional

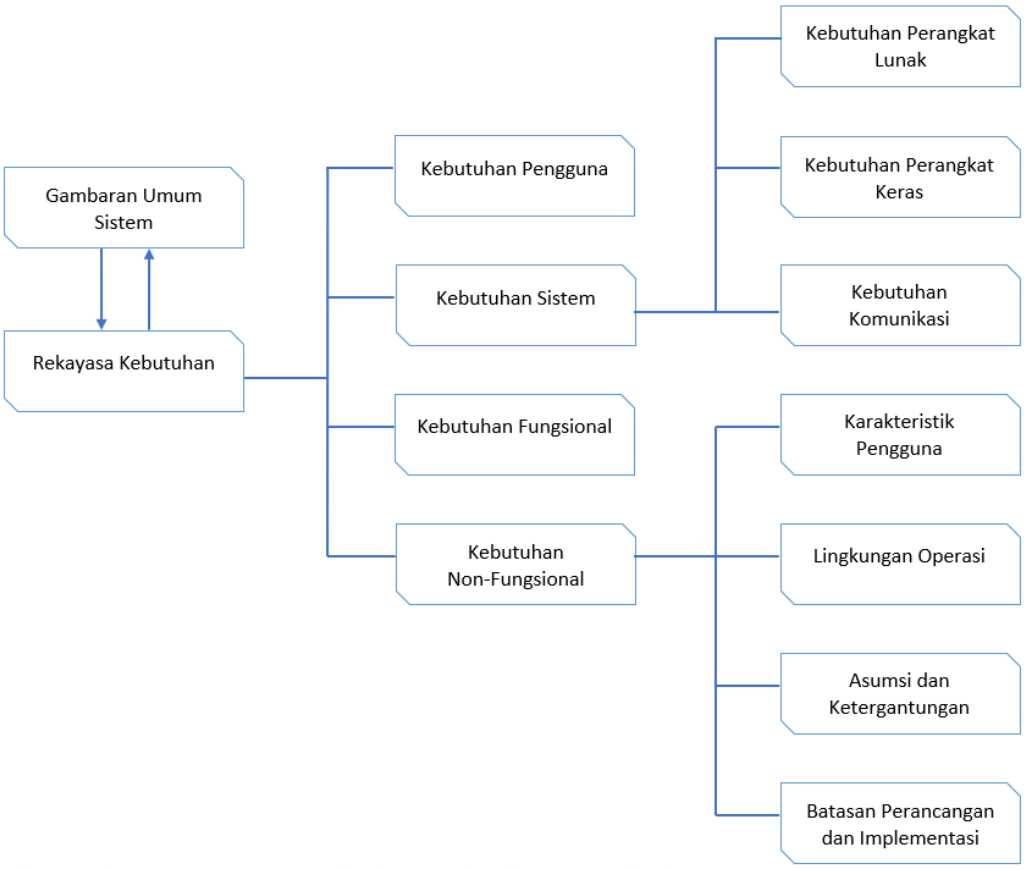
   Meliputi pengujian mengenai kinerja sistem. Hal tersebut berhubungan dengan menguji tingkat ketanggapan sistem dan jeda waktu yang dihasilkan pada sistem berdasarkan kebutuhan non-fungsional pada rekayasa kebutuhan sistem. Pengujian non-fungsional bertujuan untuk mengukur kinerja serta performa kendali navigasi robot beroda menggunakan gestur tangan dan metode kalman filter.

3.6 Kesimpulan

Penarikan kesimpulan dilakukan setelah segala tahapan pembangunan sistem telah selesai dikerjakan. Kesimpulan dapat ditarik dari analisis terhadap hasil pengujian yang dilakukan serta menjawab permasalahan pada penelitian. Perlunya penarikan saran untuk memperbaiki segala bentuk kesalahan yang terjadi pada penulisan naskah penelitian serta menyempurnakan penelitian untuk memberikan pendapat atau usulan terkait pengembangan sistem selanjutnya.

BAB 4 REKAYASA KEBUTUHAN

Rekayasa kebutuhan bertujuan untuk mengetahui apa saja kebutuhan yang diperlukan sistem pada penelitian ini. Pada tahap ini pembahasan akan dibagi menjadi lima bagian yaitu gambaran umum sistem, kebutuhan pengguna, kebutuhan sistem yang akan dijabarkan lagi berupa kebutuhan perangkat keras dan perangkat lunak, kebutuhan fungsional, serta kebutuhan non-fungsional yang akan dijabarkan lagi menjadi empat bagian yaitu karakteristik pengguna, lingkungan operasi, asumsi dan ketergantungan, serta batasan perancangan dan implementasi sesuai pada Gambar 4.1



Gambar 4.1 Diagram rekayasa kebutuhan

Rekayasa kebutuhan juga berfungsi mengidentifikasi kebutuhan-kebutuhan yang harus terpenuhi agar sistem yang dikembangkan dapat bekerja sesuai dengan yang diharapkan. Pada bab rekayasa kebutuhan sistem setiap kebutuhan dituliskan mengikuti format pada Tabel 3.1.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Tabel 4.1 Format penulisan rekayasa kebutuhan sistem

Judul					
Tipe:	Tipe	**Prioritas:**	Prioritas	**Metode Verifikasi:**	Metode Verifikasi
Rincian					
Catatan:	Catatan				

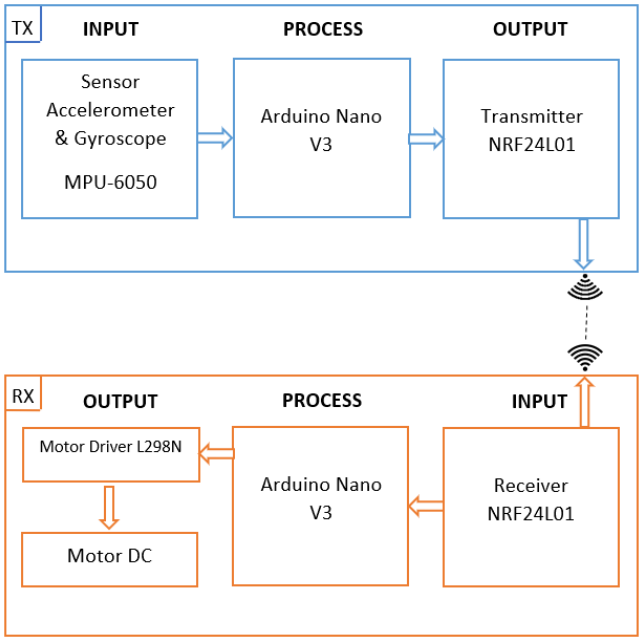
Penjelasan untuk masing-masing atribut di atas adalah sebagai berikut.

- **Judul**: judul yang menggambarkan kebutuhan secara singkat.
- **Tipe**: jenis kebutuhan.
- **Prioritas**: prioritas kebutuhan:
  - **"Tinggi"**: kebutuhan yang penting
  - **"Rendah"**: kebutuhan yang kurang penting
- **Metode Verifikasi**: metode uji keterpenuhan kebutuhan:
  - **"Demonstrasi"**: pengoperasian sistem yang bisa diamati tanpa memerlukan instrumen pengujian khusus.
  - **"Inspeksi"**: pemeriksaan secara visual terhadap *hardware*, *software code*, dokumentasi, dsb, tanpa memerlukan adanya manipulasi atau operasi.
- **Rincian**: penjelasan terperinci mengenai kebutuhan.
- **Catatan**: informasi tambahan mengenai kebutuhan.

4.1 Gambaran Umum Sistem

Pada penelitian ini akan membangun sistem kendali navigasi dari Robot beroda berdasarkan gestur tangan dengan metode *kalman filter*. Sehingga akan menghasilkan produk atau artefak teknologi informasi berupa perangkat keras. Perangkat keras yang dihasilkan berupa *wearable device* perangkat *transmitter* (TX) sebagai pengendali dan robot beroda sebagai perangkat receiver (RX) sebagai objek yang dikendalikan.

Berdasarkan diagram blok sistem pada Gambar 4.2 menunjukkan bahwa sistem terdiri dari dua perangkat utama yakni *transmitter* (TX) dan *receiver* (RX). Dalam subsistem *transmitter* terdiri dari sensor MPU-6050, arduino nano V3 dan NRF24L01(TX) sedangkan dalam subsistem *receiver* terdiri dari NRF24L01 (RX), arduino nano V3, *motor driver* dan motor DC . Masing-masing komponen terbagi dalam tiga kategori yakni *input, process* dan *output* . Dimana *input* pada TX berupa sensor MPU-6050 sebagai sensor *accelerometer* dan *gyroscope* pada gestur tangan. Kemudian sinyal data pembacaan sensor akan diolah pada mikrokontroler arduino nano V3. Setelah data diolah maka, akan di transmisikan oleh modul komunikasi NRF24L01 ke perangkat RX. Modul komunikasi NRF24L01 yang berperan sebagai *receiver* menerima data. Data tersebut akan diolah pada arduino nano V3 untuk mengendalikan *motor driver* dan motor DC.



Gambar 4.2 Diagram blok sistem

4.2 Rekayasa Kebutuhan

Mengacu pada Gambar 4.2 tahap rekayasa kebutuhan dibagi menjadi empat bagian yang bertujuan sebagai analisis mengenai kebutuhan sistem agar sistem dapat berjalan sesuai dengan tujuan penelitian. Berikut adalah uraian mengenai rekayasa kebutuhan sistem berdasarkan diagram alir rekayasa kebutuhan sistem pada gambar 4.1.

4.2.1 Kebutuhan Pengguna

Kebutuhan pengguna adalah kebutuhan yang harus dimiliki pengguna untuk dapat memantau serta mengoreksi sistem dengan baik. Pengguna harus dapat menentukan layanan mana yang harus dipenuhi agar pengguna mendapatkan hasil yang tepat sesuai dengan yang dirancang di sekitar pengguna dan sesuai dengan kebutuhannya. Berikut adalah tabel kebutuhan pengguna terhadap sistem.

Tabel 4.2 Mengetahui variasi gestur tangan

Variasi Gestur Tangan					
Tipe:	Pengguna	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
Pengguna harus mampu mengetahui variasi gestur tangan yang telah diterapkan pada sistem mulai dari gerakan gestur tangan sejajar dengan tanah, miring ke kiri, miring ke kanan, ditekuk ke atas dan ditekuk ke bawah.					
Catatan:	Variasi gestur telah didefinisikan dalam bentuk sudut dan dijelaskan lebih lanjut pada bab 6.6 perancangan kendali subbab 6.6.1 Klasifikasi gerak tangan				

Pada catatan yang ditunjukkan pada Tabel 4.2 dalam pererapan gerakan gestur tangan telah didefinisikan dalam bentuk sudut. Oleh karena itu gerakan tangan ke atas, bawah, kiri dan kanan telah direpresentasikan menjadi sudut -90^0 - 90^0 dan dibatasi pada masing-masing gerakan maksimal 90^0 dikarenakan batas lekukan sendi pergelangan tangan manusia rata-rata 90^0 .

Tabel 4.3 Indikator antarmuka pergerakan robot beroda

Indikator Pergerakan					
Tipe:	Pengguna	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
Indikator pegerakan Robot beroda dapat diketahui dengan indera penglihatan pengguna berdasarkan nyala atau padamnya Light Emmited Diode (LED) yang tertanam pada perangkat *transmitter* (TX)					
Catatan:	Indikator berupa LED dengan masing-masing LED mewakili arah gerakan gestur tangan				

Pada Tabel 4.3 diatas pada dasarnya lampu LED menjadi indikator terhadap gerakan gestur tangan pada pengguna. Sejalan dengan Robot beroda , misalkan apabila gerakan ke kiri maka robot akan bergerak ke kiri bersamaan dengan nyala lampu LED sebelah kiri. Hal ini akan memudahkan pengguna untuk mengetahui arah gerakan yang dilakukan.

4.2.2 Kebutuhan Sistem

Kebutuhan sistem yang juga dikenal sebagai *System Requirements Specification* (SRS) adalah dokumen atau serangkaian dokumentasi yang menjelaskan fitur dan perilaku sistem atau aplikasi perangkat lunak dan perangkat keras. Mencoba untuk menentukan fungsi yang diinginkan oleh pengguna berdasarkan kemampuan dan kebutuhan sistem. *Institute of Electrical and Electronics Engineers* (1994) mendefinisikan persyaratan dan kebutuhan sebagai (1) Kondisi atau kemampuan yang dibutuhkan oleh pengguna untuk memecahkan masalah atau mencapai tujuan. (2) Suatu kondisi atau kemampuan yang harus dipenuhi atau dimiliki oleh suatu sistem atau komponen sistem untuk memenuhi suatu kontrak, standar, spesifikasi, atau dokumen lain yang dikenakan secara formal. (3) Representasi kondisi atau kemampuan terdokumentasi sebagaimana dalam (1) atau (2). Kebutuhan sistem dibagi menjadi tiga bagian yaitu kebutuhan perangkat keras, perangkat lunak dan komunikasi.

4.2.2.1 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak merupakan kebutuhan keseluruhan aplikasi yang digunakan untuk pengembangan sistem. Berikut adalah perangkat lunak yang dibutuhkan oleh sistem :

1. **Arduino IDE**

Tabel 4.4 Fungsi Arduino IDE

Arduino IDE					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
Arduino IDE berfungsi sebagai lingkungan terintegrasi untuk mengembangkan pemrograman yang dibenamkan melalui sintaks pemrograman.					
Catatan:	Dalam arduino IDE juga dapat memantau proses sensor yang dilakukan oleh sistem serta dapat direpresentasikan dalam bentuk teks atau grafik				

Berdasarkan Tabel 4.4, *The Arduino Integrated Development Environment* atau Arduino *Software* (IDE) merupakan lingkungan terintegrasi berisi editor teks untuk menulis kode, area pesan, konsol teks dan tombol untuk melakukan pengembangan terhadap modul Arduino. Terhubung ke perangkat keras Arduino dan Genuino untuk mengunggah program dan berkomunikasi. Menggunakan bahasa pemograman yang menyerupai bahasa C. Dilengkapi dengan *library* C/C++ yang biasa disebut *wiring* sehingga dapat membuat operasi *input* dan *output* menjadi lebih mudah.

4.2.2.2 Kebutuhan Perangkat Keras

Seluruh perangkat keras dan komponen elektronika yang dibutuhkan dalam membangun sistem disebut dengan *hardware requirement* atau kebutuhan perangkat keras. Berikut adalah perangkat keras yang dibutuhkan :

1. **Arduino Nano V3**

Tabel 4.5 Spesifikasi kebutuhan Arduino Nano V3

Arduino Nano V3					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Mikrokontroler ATmega328 2. Tegangan Operasi (*level logic*): 5 V 3. Tegangan Masukan (disarankan): 7-12 V 4. Tegangan Masukan (batas): 6-20 V 5. Digital I / O Pin: 14 (dimana 6 menyediakan *output* PWM) 6. Masukan Pin Analog: 8 7. Arus DC I / O Pin: 40 mA 8. *Flash Memory* 32 KB (ATmega328) dimana 2 KB digunakan oleh *bootloader* 9. SRAM: 2 KB (ATmega328) 10. EEPROM: 1 KB (ATmega328) 11. *Clock Speed* : 16 MHz					

12. Dimensi: 0,73 "x 1,70"	
Catatan:	Dibutuhkan dua arduino nano pada sistem yang terletak pada perangkat TX dan RX

Pada Tabel 4.5 dijelaskan spesifikasi arduino nano V3 yang dibutuhkan oleh sistem. Dibutuhkan dua buah mikrokontroler arduino nano V3 dalam sistem. Berikut peranan kedua mikrokontroler tersebut.

Tabel 4.6 Fungsi Arduino Nano V3 (Terletak pada TX)

Arduino Nano V3					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Akuisisi sinyal analog dari sensor MPU-6050 berupa data *raw* 2. Proses filter sinyal data yang diterima 3. Konversi data *raw* sinyal menjadi bentuk derajat 4. Menyalakan indikator pergerakan Robot beroda 5. Mengontrol NRF24L01 untuk mengirim data sinyal dari TX menuju RX					
Catatan:	Fungsi dan kebutuhan arduino nano V3 pada TX				

Tabel 4.7 Fungsi Arduino Nano V3 (Terletak pada RX)

Arduino Nano V3					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Akuisisi data sinyal yang dikirimkan melalui modul komunikasi NRF24L01 *transmitter* 2. Mengolah sinyal data yang diterima dengan, 3. Mengklasifikasi sinyal data sesuai kategori gerakan gestur tangan yakni ; maju,mundur,kanan dan kiri 4. Mengatur PWM motor sesuai dengan sudut pada sinyal data					
Catatan:	Fungsi dan kebutuhan arduino nano V3 pada RX				

Jadi pada dasarnya arduino nano V3 pada Tabel 4.6 dan Tabel 4.7 memiliki fungsi dan tugas yang berbeda. Sesuai dengan apa yang dibutuhkan pada sistem baik itu pada perangkat TX ataupun RX. Alasan penggunaan arduino nano V3 dalam penelitian ini adalah :

1. Dalam segi dimensi atau ukuruan arduino nano V3 cukup kecil sehingga cocok digunakan pada tangan manusia.
2. Mendukung komunikasi SPI yakni telah terdapat pin khusus yang dibutuhkan untuk komunikasi miso / mosi.
3. Terdapat 14 pin digital dan 6 diantaranya PWM yang sangat berguna untuk kebutuhan modul komunikasi NRF24L01 yang membutuhkan 5 slot pin digital pada perangkat TX atau RX.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

4. USB *jack* Arduino Uno, menggunakan USB type A, sedangkan Nano menggunakan USB mini yang jauh lebih sederhana.
5. Dengan ukuran yang kecil namun kemampuan tidak jauh beda dengan mikrokontroler yang ukurannya lebih besar.
6. Dalam segi ekonomi dengan kemampuan yang baik harga arduino nano V3 juga sangat terjangkau sehingga dapat efisien dalam segi pendanaan.

2. *Motor driver* L298N

Tabel 4.8 Spesifikasi kebutuhan *Motor driver* L298N

***Motor driver* L298N**					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
Spesifikasi yang dibutuhkan : 1. Dual H *Bridge Motor driver* 2. L298N *driver motor* IC 3. Menghubungkan hingga 2 motor DC dua arah 4. Pengatur daya 5V terintegrasi 5. 5V - 35V tegangan penggerak 6. 2A arus penggerak maksimal 7. Daya Maksimal: 25W 8. Dimensi: 43 x 43 x 26mm 9. Berat: 26g					
Catatan:	Spesifikasi *Motor driver* L298N yang dibutuhkan oleh sistem				

Motor driver L298N merupakan modul elektronika yang berfungsi untuk mengendalikan arah putaran dan kecepatan motor DC ataupun motor *stepper.* Pada Tabel 4.8 merupakan spesifikasi *motor driver* yang dibutuhkan oleh sistem. Pemilihan modul *motor driver* L298N menjadi pengendali motor adalah karena tegangan *input* yang bisa ditampung oleh modul sebesar 5V – 12V maksimal dapat mencapai 35V. Sehingga *output* tegangan yang dihasilkan untuk mengendalikan motor DC cukup besar pula. Karena fokus penelitian adalah mengendalikan pergerakan Robot beroda maka salah satu indikatornya adalah kecepatan bergerak.

3. NRF24L01

Tabel 4.9 Spesifikasi kebutuhan NRF24L01

NRF24L01					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
1. *Working voltage :* DC 1.9V – 3.6V 2. *Modulation : GFSK* 3. *Maximum transmission rate : 2Mbps* 4. *Moment of maximum operating current : <15mA* 5. *Operating frequency : 2.400-2.524GHZ (industrial, Scientific, Medical)*					

6. *Interface : SPI* 7. Dimensi : 1,5 cm x 2,9 cm 8. Berat : 20 gr	
Catatan:	Spesifikasi NRF24L01 yang dibutuhkan oleh sistem

Modul nirkabel NRF24L01 adalah modul radio 2.4GHz yg memakai *Nordic Semiconductor* NRF24L01 yang telah dilengkapi 2.4GHz RF *transceiver*, RF *synsthesizer* dan *baseband logic* termasuk *Enhanced ShockBurst hardware protocol accelerator* yang mendukung interface SPI untuk aplikasi pengendalian jarak jauh. Untuk lebih jelasnya spesifikasi yang dibutuhkan seperti pada Tabel 4.9 diatas. Berikut merupakan kelebihan NRF24L01 sebagai modul komunikasi pada penelitian ini :

1. Menggunakan chip NRF24L01 yg bekerja pada ISM *(Industrial, Scientific and Medical) band monolithic transceiver* 2.4-2.5 GHz *Worlwide License-Free.*
2. Konsumsi daya rendah sehingga dapat menghemat daya pada baterai
3. *Output power channel selection dan setting protocol* dapat diatur ke konsumsi arus yang sangat rendah, melalui *interface* SPI
4. Kecepatan tinggi: opsi 250kbps, 1Mbps dan 2Mbps *on-air data-rate*
5. Biaya aplikasi yg rendah: NRF24L01 sudah mengintegrasikan semua komponen pemrosesan sinyal kecepatan tinggi yang berhubungan dengan *protocol* RF, seperti: re-transmisi otomatis paket data yg hilang dan secara otomatis menghasilkan sinyal respon, dll
6. Internal FIFO yg kompatibel dengan beragam *interface* mikroprosesor termasuk dalam penelitian ini arduino nano V3

4. Motor DC

Tabel 4.10 Spesifikasi kebutuhan Motor DC

Motor DC					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
1. Tegangan kerja: 3 - 7.4v (rekomendasi 5v) 2. Rasio kecepatan tanpa beban: 1:48 (saat 3v) 3. Torsi max: 0.8 kgcm 4. Kecepatan (meter/menit): 3v (26), 5v (44) 5. Dimensi *gearbox* : 70x22x18mm 6. Dimensi roda dan ban : diameter 66mm x 25mm					
Catatan:	Spesifikasi Motor DC yang dibutuhkan oleh sistem				

Banyak sekali jenis motor DC , mulai dari torsi tinggi hingga rendah serta motor DC yang cepat hingga lambat. Namun spesifikasi yang dibutuhkan seperti yang ditunjukkan pada Tabel 4.10 . Yaitu motor DC yang mampu berjalan pada tegangan 3 - 7.4 V sehingga kecepatan tidak terlalu kencang dan tidak terlalu lambat. Karena tegangan masukkan akan berdampak pada laju motor DC . Serta motor DC ini telah dilengkapi dengan gearbox dan ban sehingga lebih praktis dan sesuai dengan kebutuhan.

REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

5. Baterai Li-Po dan Baterai 9V

Tabel 4.11 Spesifikasi kebutuhan Baterai Li-Po

Baterai Li-Po					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
1. Paket 2-sel 7.4V 2. 2200mAh muatan 3. Tingkat pelepasan terus menerus 25C 4. Tingkat debit puncak 35C (10 detik) 5. Steker muatan JST-XH 6. Dimensi 4,13 x 1,30 x 0,67 ”(105 x 33 x 17mm)					
Catatan:	Baterai jenis ini digunakan pada perangkat RX sebagai penggerak motor DC				

Pada Tabel 4.11 menjelaskan spesifikasi baterai yang digunakan untuk perangkat RX. Pemilihan baterai bertipe Li-Po atau *Lithium Polymer* adalah memiliki daya tahan lebih baik dari *alkaline*, penggunaannya pun bisa sampai bertahun-tahun (*Durability*). Dengan kapasitas baterai sebesar 2200mAh membuat daya tahan baterai saat digunakan menjadi cukup lama. Serta tegangan 7.4 V yang sesuai dengan kebutuhan untuk menggerakkan motor DC.

Tabel 4.12 Spesifikasi kebutuhan Baterai 9v

Baterai 9v					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
1. Klasifikasi: *Alkalin* 2. Sistem Kimia: *Zinc-Manganese Dioxide* (Zn / MnO) 3. Tidak ditambahkan merkuri atau cadmium 4. Tegangan Nominal: 9,0 volt 5. Suhu Operasi: -18 ° C hingga 55 ° C 6. Berat Khas: 45,0 gram (1,58 oz.) 7. Volume Khas: 21,1 sentimeter kubik 8. *Shelf Life*: 5 tahun pada 21 ° C Kebutuhan Komponen : 1. Arduino : 6 - 12v 2. MPU-6050 : 1.9 - 3.6v 3. LED : 2.1v 4. NRF24L01 : DC 1.9v – 3.6v					
Catatan:	Baterai jenis ini digunakan pada perangkat TX sebagai penggerak motor DC				

Baterai jenis ini digunakan pada perangkat TX untuk menyalakan Arduino Nano V3, Sensor *Accelerometer* MPU-6050, NRF24L01 *Transmitter* dan LED . Pada Tabel 4.12 menunjukkan spesifikasi dari baterai 9v. Dengan keluaran tegangan

sebesar 9v sangat sesuai dengan kebutuhan yang diperlukan oleh komponen pada TX.

6. PCB *Matrix*

Tabel 4.13 Spesifikasi kebutuhan PCB *Matrix*

PCB *Matrix*					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
1. Ukuran: 5x7 cm 2. Jumlah lubang : 432 (18x24) 3. Diameter lubang: +/- 1mm 4. Jarak lubang: 2.54mm (Standar)					
Catatan:	Spesifikasi PCB *Matrix* sebagai penopang komponen pada sistem				

Dibutuhkan dua keping PCB *Matrix* sesuai dengan spesifikasi pada Tabel 4.13 untuk menopang dan menghubungkan komponen elektronika pada perangkat TX maupun RX. Menggunakan PCB *Matrix* membuat rangkaian komponen elektronika menjadi lebih praktis dan sistem perkabelan antar komponen dapat teratur. Serta yang paling penting adalah dapat mengurangi volume perangkat sehingga lebih kecil dan tidak memakan tempat yang luas ketika ditempatkan.

7. Sakelar

Tabel 4.14 Spesifikasi kebutuhan sakelar

Sakelar					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
1. Sakelar yang digunakan bertipe *On/Off* 2. Berjumlah dua untuk ditempatkan pada perangkat TX dan RX					
Catatan:	Spesifikasi kebutuhan sakelar				

Pada Tabel 4.14 dijelaskan bahwa sakelar yang dibutuhkan adalah bertipe *on/off* yang mana berfungsi untuk memutuskan jaringan listrik atau menghubungkannya.

8. Kabel *Jumper*

Tabel 4.15 Spesifikasi kebutuhan kabel *jumper*

Matlab					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
1. Untuk menghubungkan komponen elektronika					
Catatan:	Kebutuhan kabel *jumper* sebagai penguhubung				

Pada Tabel 4.15 dijelaskan fungsi dari kabel *jumper*. Pada sistem ini kabel yang digunakan ada dua jenis yakni kabel tembaga ukuran 0.5 mm dengan kabel

jumper bertipe *male to female* untuk menghubungkan rangkaian elektronika pada sistem.

9. Akrilik

Tabel 4.16 Spesifikasi dan fungsi akrilik

Akrilik					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Demonstrasi
1. Akrilik berfungsi untuk membuat kerangka robot 2. Akrilik yang digunakan dengan tebal 3 mm 3. Akrilik DOP degan dimensi : 13x17 cm					
Catatan:	Akrilik yang digunakan untuk membuat kerangka robot				

Seperti yang dijelaskan pada Tabel 4.16 akrilik berfungsi sebagai bahan pembuat kerangka pada robot. Pemilihan bahan akrilik pada kerangka robot dikarenakan adalah kuat, lentur, dan tahan lama, aman untuk makanan karena mikroorganisme tidak mungkin berkembang serta dapat dibentuk sesuai dengan kebutuhan.

4.2.2.3 Kebutuhan Komunikasi

Kebutuhan komunikasi merupakan kebutuhan akan kemampuan komponen menyampaikan informasi serta menjadi media penghubung antar komponen.

1. MPU-6050

Tabel 4.17 Spesifikasi kebutuhan dan fitur MPU-6050

MPU-6050					
Tipe:	Sistem	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
Spesifikasi yang dibutuhkan : 1. *Gyroscope* tiga sumbu dan *accelerometer* tiga sumbu 2. Catu Daya: 4.3 hingga 9 V (*regulator drop-out* KB33 rendah) 3. Mode Komunikasi: protokol komunikasi IIC (I2C) standar 4. Konverter ADC 16bit internal, *output* data 16-bit 5. Rentang *Gyroscope*: +/- 250, +/- 500, +/- 1000, +/- 2000 ° / dtk 6. Rentang akselerasi: +/- 2g, +/- 4g, +/- 8g, +/- 16g 7. Dimensi (tidak termasuk pin): 21,2mm (0,84 ") panjang x 16,4mm (0,65") lebar x 3,3mm (0,13 ") tinggi 8. Berat: 2.1g (0.07oz) Fitur : 1. I2C Digital-*output* dari data *Motion Fusion* 6 atau 9-sumbu dalam *matrix* rotasi, *quaternion, euler angle*, atau format data mentah					

2. Sensor kecepatan sudut 3-*Axis* (*gyro*) dengan sensitivitas hingga 131 LSBs / dps dan rentang skala penuh ± 250, ± 500, ± 1000, dan ± 2000dps 3. *Accelerometer* Tri-*Axis* dengan rentang skala penuh yang dapat diprogramkan ± 2g, ± 4g, ± 8g dan ± 16g 4. Mesin *Digital Motion Processing* ™ (DMP ™) mengeluarkan *Motion Fusion* yang kompleks, sinkronisasi pengaturan sensor, dan deteksi gerakan 5. Algoritma tertanam untuk *bias run-time* dan kalibrasi kompas. Tidak diperlukan intervensi pengguna
Catatan: Spesifikasi sensor *Acce*lererometer MPU-6050

Pada tabel 4.17 menunjukkan spesifikasi sensor MPU-6050 . Pemilihan *accelerometer* MPU-6050 dalam penelitian adalah modul sensor MPU-6050 dilengkapi Perangkat Pelacakan 6-*axis motion*. Ini menggabungkan 3-*axis Gyroscope*, 3-*axis Accelerometer* dan *Digital Motion Processor* semua dalam paket kecil. Selain itu, memiliki fitur tambahan sensor Suhu *on-chip*. Memiliki antarmuka bus I2C untuk berkomunikasi dengan mikrokontroler. Dengan mengukur akselerasi dengan kisaran skala penuh minimum ± 3 g. Alat ini dapat mengukur percepatan gravitasi statis dalam aplikasi *tilt-sensing*, serta akselerasi dinamis yang dihasilkan dari gerakan, guncangan, atau getaran. Sehingga sesuai diterapkan pada sistem yang mana mengandalkan akselerasi, ketepatan sudut serta kecepatan dalam gerakan gestur tangan.

4.2.3 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan menentukan proses-proses yang berkaitan dengan prinsip kerja sistem mencakup tujuan serta keluaran yang harus sesuai dengan masukkan yang dberikan. Berikut merupakan proses-proses yang disediakan oleh sistem.

1. Perangkat TX Terhubung dengan Perangkat RX

Tabel 4.18 Menjelaskan kebutuhan menghubungkan perangkat TX dan RX

Perangkat TX terhubung dengan perangkat RX					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Sistem harus mampu terhubung oleh komponen NRF24L01 dengan jarak frekuensi 2.4-2.5GHz. 2. Dengan menggunakan *library RadioHead* pada arduino nano V3 untuk menyediakan pustaka berorientasi objek lengkap untuk mengirim dan menerima pesan terkemas. 3. Mendukung komunikasi MISO dan MOSI pada arduino Nano dengan memanfaatkan pin digital 12 dan 11. 4. Indikator terhubungnya perangkat TX dan RX adalah dengan nyala led RX pada arduino nano yang berada pada perangkat RX					
Catatan:	Perangkat TX dan RX terhubung hanya jika catu daya dalam kedua perangkat tersebut dalam keadaan nyala.				

Dalam Tabel 4.18 kemampuan kedua perangkat TX dan RX terhubung merupakan proses yang sangat penting dalam sistem. Sistem harus mampu berkomunikasi searah dari TX menuju RX untuk mengirimkan nilai. Apabila perangkat TX dinyalakan kemudian selanjutnya perangkat RX dinyalakan maka kedua perangkat akan otomatis saling terhubung dengan waktu tunggu sekian detik. Bersamaan dengan munculnya indikator LED pada arduino nano yang berkedip tanda bahwa perangkat RX menerima data dari perangkat TX.

2. Gerakan Gestur Tangan

Tabel 4.19 Menjelaskan berbagai macam gerakan gestur tangan

Gerakan Gestur Tangan					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
Sistem harus mampu menerjemahkan gerakan gestur tangan menjadi besaran fisis yang didapat dari sensor MPU-6050. Gerakan tangan dibagi menjadi lima macam berdasarkan masukan nilai sensor yang telah diproses menjadi bentuk derajat. Berikut merupakan gerakan yang dianjurkan dalam sistem : 1. Gerakan gestur tangan sejajar dengan tanah yang artinya robot beroda dalam keadaan *idle* atau diam. 2. Gestur tangan yang terbentuk miring ke kiri yang artinya Robot beroda belok ke kiri. 3. Gestur tangan yang terbentuk miring ke kanan yang artinya Robot beroda belok ke kanan. 4. Gestur tangan yang terbentuk miring ke bawah yang artinya Robot beroda bergerak maju. 5. Gestur tangan yang terbentuk ditekuk ke atas yang artinya Robot beroda bergerak mundur.					
Catatan:	Dikarenakan sensitifitas yang tinggi dari sensor MPU-6050 maka gerakan sudut sekecil apapun pada sensor akan terbaca sehingga menyebabkan Robot beroda bergerak.				

Pada Tabel 4.19, gestur tangan hanya dibatasi lima gerakan saja agar pengguna dapat fokus menguji kendali gerakan oleh sensor terhadap Robot beroda. Oleh karena itu dilakukan klasifikasi sudut yang bertujuan sebagai rentang dimana setiap rentang sudut yang telah ditentukan dapat diterjemahkan menjadi minimal satu dan maksimal dua gerakan oleh sistem. Misalkan apabila kita ingin maju sambil belok kanan artinya lekukan telapak tangan kita bergerak ke bawah dan sedikit kearah kanan.

3. Akuisisi Nilai dari Sensor IMU MPU-6050

Tabel 4.20 Menjelaskan akuisisi nilai sensor

Akuisisi nilai dari sensor					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Dimulai dengan mengukur besaran fisis berupa gerakan gestur tangan					

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

2. Dibutuhkan suatu sensor jenis tran*duser* yang dapat mengubah energi mekanik menjadi energi listrik (piezoelektrik). 3. Sistem mampu akuisisi nilai dari sensor MPU-6050 4. Data akuisisi berupa data *raw* (analog) 5. Akuisisi data berupa nilai *pitch* dan *roll* 6. Dengan menyalakan sakelar pada perangkat TX, sistem akan otomatis dapat langsung mengakuisisi nilai pada sensor tersebut.	
Catatan:	Akuisisi nilai sensor merupakan langkah awal dalam proses pengolahan sinyal sebelum di transmisikan ke perangkat *receiver*

Pada Tabel 4.20 menjelaskan kemampuan sistem dalam mengakuisisi data serta kebutuhan untuk memperoleh nilai dari sensor. Akuisisi data *raw* berupa nilai dari sumbu "x" dan "y" lebih tepatnya nilai *pitch* dan *roll*. Pergerakan gestur tangan hanya bergerak miring ke kiri, kanan, ditekuk ke atas dan ke bawah.

4. Proses Filter Nilai Sensor

Tabel 4.21 Menjelaskan proses filter sinyal sensor MPU-6050

Filter nilai sensor					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Proses filter dilakukan dengan metode *kalman filter* 2. *Sensor Fusion* atau penggabungan *acce*lerometer dan *gyro*scope menggunakan *complementary filter* 3. Algotima filter menggabungkan model dan pengukuran sehingga estimasi selalu mendekati kondisi *real*. 4. Dibutuhkan *library kalman filter* untuk proses filter pada arduino uno v3 sesuai dengan konsep dasar *kalman filter* dan penerapannya pada sistem 5. Sinyal *pitch* dan *roll* merupakan objek proses filter sebelum nilai diolah oleh sistem 6. Dibutuhan nilai Q = *noise proses* dan nilai R = *noise pengukuran* untuk meredam *noise* pada sinyal					
Catatan:	Untuk tampilan antarmuka sinyal bisa diamati melalui *serial plotter* pada arduino nano V3 dan Microsoft Excel melalui PLX-DAQ				

Filter sangat berguna untuk meredam serta menghilangkan *noise* atau gangguan pada sinyal. Dikarenakan sinyal merambat pada suatu medium dan sangat mudah terkena gangguan, baik gangguan internal maupun eksternal. Selain itu gangguan juga terjadi akibat beberapa faktor diantaranya *bias errors, scale factor errors dan time-correlated noise* (Martin, 2017). Seperti yang dijelaskan pada Tabel 4.21 dibutuhkan kriteria untuk meredam gangguan pada sinyal sehingga dapat meningkatkan akurasi.

5. Konversi Nilai Sensor (Data *Raw*) Menjadi Sudut

Tabel 4.22 Menjelaskan konversi nilai sensor menjadi derajat

Tahapan konversi nilai Sensor yang dibutuhkan oleh sistem

Tipe:	Fungsional	Prioritas:	Tinggi	Metode Verifikasi:	Inspeksi
Sistem harus mampu menerima nilai dari sensor dalam bentuk data *Raw* atau biasa yang disebut dengan nilai analog. Tahap selanjutnya adalah proses filter dan kemudian dikonversi dengan perhitungan matematis sehingga dapat diterjemahkan menjadi bentuk derajat/sudut.					
Catatan:	Pada dasarnya kode program yang digunakan merupakan interpretasi matematis dari kode program maupun *libarary.*				

Kebutuhan akan konversi nilai sensor sangat penting oleh sistem untuk dapat memudahkan pengguna dalam memahami gerakan yang dibuat terhadap dampak yang dihasilkan sesuai yang dijelaskan pada Tabel 4.22.

6. Konversi Nilai Sensor sudut menjadi PWM *(Pulse Width Modulation)*

Tabel 4.23 Menjelaskan konversi nilai sensor menjadi PWM

Konversi nilai sensor yang dibutuhkan oleh sistem					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
Sistem harus mampu mengkonversi dan memetakan nilai dari derajat atau sudut dalam bentuk PWM (*Pulse Width Modulation*) sebelum dikirim oleh melalui modul komunikasi. Aplikasi pwm pada sistem untuk mengatur putaran motor DC. Semakin besar nilai *duty cycle* yang diberikan maka akan berpengaruh terhadap cepatnya putaran motor. Apabila nilai *duty cylce*-nya kecil maka motor akan bergerak lambat.					
Catatan:	Pada dasarnya nilai yang dikirimkan dari perangkat *transmitter* menuju perangkat *receiver* merupakan nilai pwm yang berfungsi mengendalikan motor DC pada perangkat *receiver*				

PWM merupakan salah satu teknik modulasi dengan mengubah lebar pulsa (*duty cylce*) dengan nilai amplitudo dan frekuensi yang tetap. Kebutuhan akan konversi nilai derajat atau sudut menjadi PWM sangat penting oleh sistem untuk dapat memudahkan dalam mengendalikan gerakan robot beroda seperti yang telah dijelaskan pada Tabel 4.23.

7. Pengiriman Nilai PWM dari *Transmitter* (TX) menuju *Receiver* (RX)

Tabel 4.24 Menjelaskan proses pengiriman nilai PWM melalui NRF24L01

Pengiriman nilai pwm					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Sistem harus mampu berkomunikasi untuk mengirimkan nilai dari modul komunikasi NRF24L01 *transmitter* pada perangkat TX 2. Data yang dikirim berupa nilai derajat atau sudut yang telah dikonversi menjadi nilai PWM.					
Catatan:	Nilai PWM bertujuan untuk mengendalikan gerakan navigasi pada robot beroda.				

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Pada Tabel 4.24 Gerakan navigasi tidak hanya berupa gerak maju, mundur, kanan dan kiri melainkan mengendalikan kecepatan motor sehingga kecepatan motor dapat dikendalikan berdasarkan PWM yang telah diterima.

8. Penerimaan Nilai dari *Transmitter*(TX) menuju *Receiver* (RX)

Tabel 4.25 Menjelaskan proses penerimaan nilai PWM melalui NRF24L01

Penerimaan nilai pwm					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Modul komunikasi RX *Receiver* harus dapat menerima nilai yang dikirimkan NRF24L01 *transmitter* 2. Data yang diterima berupa nilai derajat atau sudut yang telah dikonversi menjadi nilai PWM.					
Catatan:	Nilai PWM yang telah diterima kemudian diolah pada mikrokontroler arduino nano v3 pada robot beroda.				

Pada Tabel 4.25 menjelaskan kemampuan sistem dalam menerima nilai yang didapatkan oleh sensor MPU-6050 berupa nilai pwm.

9. Navigasi Robot beroda

Tabel 4.26 Menjelaskan kebutuhan navigasi robot

Navigasi Robot					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Sistem harus mampu mentransformasikan gestur tangan berupa nilai pwm yang diterima untuk mengendalikan navigasi Robot beroda. 2. Setiap perubahan gestur yang terjadi pada perangkat *gloves* akan mengirimkan nilai pwm pada robot beroda. Variasi nilai pwm yang diterima kemudian diolah untuk mengatur motor driver pada robot. 3. Peran motor driver untuk mengendalikan motor DC atau kecepatan motor 4. Sehingga robot beroda dapat bergerak sesuai dengan kecepatan *input* sudut dari pengguna 5. Semakin besar sudut yang diimplementasikan pada perangkat *gloves* maka akan semakin cepat gerakan pada robot beroda begitu juga sebalinya, apabila sudut yang di *input* kecil maka gerakan pada robot menjadi lambat. 6. Gerakan navigasi robot memungkinkan untuk bergerak dua arah secara bersama-sama. Misalkan gerakan maju dan belok bersama-sama maka akan menjadi gerakan serong pada robot beroda.					
Catatan:	Jadi pada dasarnya arah gerakan robot beroda dapat disesuaikan sesuai dengan *input* gestur tangan pengguna.				

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Pada Tabel 4.26, menjelaskan tentang mekanisme navigasi robot beroda mulai dari menerima nilai pwm sampai kendali navigasi berdasarkan *input* sudut gestur tangan pada penguna. Tahapan konversi dari data *raw* menjadi bentuk sudut kemudian dikonversi kembali menjadi bentuk nilai pwm. Untuk mempermudah penggunaan nilai dikoversi menjadi sudut, sedangkan untuk mempermudah pengendalian kecepatan navigasi robot beroda nilai sudut di konversi menjadi nilai pwm.

10. Indikator Navigasi Robot beroda

Tabel 4.27 Menjelaskan Indikator navigasi robot

Indikator Navigasi					
Tipe:	Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Sistem harus mampu memberikan indikator navigasi Robot beroda terhadap pengguna. 2. Indikator navigasi robot berupa nyala dan padamnya lampu LED pada perangkat pengendali yang diletakkan ditangan. 3. Dibutuhkan empat lampu LED yang akan mewakili kondisi navigasi Robot beroda yakni ; Maju, mundur, kiri, kanan. 4. Indikator pada perangkat pengendali harus sesuai dengan navigasi pada Robot beroda.					
Catatan:	Indikator navigasi sangat berguna untuk menginformasikan pergerakan Robot beroda kepada pengguna				

Pada Tabel 4.27 menjelaskan bagaimana kebutuhan indikator pada sistem terhadap pengguna. Dalam hal ini indikator navigasi terdiri dari empat lampu led yang diletakkan diperangkat pengendali.

4.2.4 Kebutuhan Non-Fungsional

Kebutuhan Non-Fungsional merupakan kebutuhan yang berkaitan dengan batasan layanan atau fungsi yang ditawarkan oleh sistem sesuai dengan perencanaan. Sehingga sistem dapat mengetahui peluang-peluang atau bagian apa saja yang dapat ditingkatkan untuk meningkatkan fungsi dari sistem tersebut. Berikut meupakan uraian mengenai kebutuhan non-fungsional.

4.2.4.1 Kebutuhan Karakteristik Pengguna

Tabel 4.28 Menjelaskan kebutuhan karakteristik pengguna

Karakteristik Pengguna					
Tipe:	Non-Fungsional	**Prioritas:**	Sedang	**Metode Verifikasi:**	Inspeksi
1. Lengkap, Artinya segala kebutuhan yang menyangkut pengguna ataupun sistem harus lengkap dan terpenuhi. Karena apabila tidak maka fungsionalitas yang diharapkan oleh pengguna akan tidak sesuai.					

2. Konsisten, Maksudnya adalah tidak ada kebutuhan yang bertentangan antara pengguna dengan sistem atau dengan faktor eksternal tertentu.
3. Jelas, Artinya penguna harus dapat memahami secara jelas dan spesifik makna dan kinerja dari sistem.
4. Bermutu, Suatu kebutuhan harus dapat diverifikasi dan divalidasi untuk menunujukkan bahwa kebutuhan tersebut dapat dibangun kedalam sistem. Artinya kebutuhan harus dinyatakan dalam istilah fisikal sebagai atribut yang dapat diukur.
5. Toleransi Kesalahan, Tujuan toleransi kesalahan sistem yang harus dipahami oleh pengguna adalah menjamin bahwa kinerja sistem secara keseluruhan mampu terus berfungsi meskipun terjadi kesalahan.
6. Terbatas, Seluruh sistem yang digunakan harus tunduk pada batasan waktu, kualitas dan sumber daya. Sehingga ruang lingkup proyek dapat dikendalikan dan memiliki prioritas berdasarkan batas-batas yang ada.
7. Kredibel, Maksudnya adalah pengguna dapat melihat proyek secara teknis dan dapat menilai layak tidaknya sistem dapat dicapai dalam keterbatasan yang ada.
8. Auditabilitas dan Kontrol, Pengguna harus memiliki kemampuan dalam pengontrolan sistem dan dapat menginterpretasikan hasil pengalaman/percobaan sehingga menjadi layak untuk di audit.
9. Dokumentasi, Pengguna atau peneliti setelah menilai bahwa sistem layak di audit maka langkah berikutnya adalah mendokumentasikan hasil/pengalamannya dalam bentuk tertulis ataupun tidak tertulis.

Catatan:	Diatas merupakan karakteristik yang disusun oleh penulis sebagai acuan atau batasan pemilihan pengguna

Seperti yang ditunjukkan pada Tabel 4.28**,** untuk memperoleh daftar kebutuhan yang baik dan berkualitas maka harus memperhatikan karakteristik kebutuhan yang dimiliki oleh pengguna. Tentunya dengan menekan masalah kontrol, komunikasi dan kontruksi pada sistem. Sehingga dapat menciptakan serangkaian rekayasa kebutuhan yang sesuai, solid, dan dapat memenuhi kebutuhan pengguna.

4.2.4.2 Kebutuhan Lingkungan Operasi

Tabel 4.29 Menjelaskan kebutuhan lingkungan operasi

Lingkungan Operasi					
Tipe:	Non-Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi

1. Sistem harus di operasikan pada permukaan yang datar dan rata. Dikarenakan Robot beroda menggunakan ban atau roda dalam bergerak.
2. Untuk penggunaan weareble *device* berupa *gloves* yang dirancang dalam bentuk sarung tangan digunakan pada tangan kanan manusia.
3. Jarak antara perangkat RX dan TX tidak melebihi 100 meter. Dikarenakan keterbatasan sinyal dan antena pada perangkat komunikasi.

Catatan:	Berbeda dengan ruang pengujian yang memiliki poin-poin pengukuran dan perhitungan dalam menguji sistem

Tabel 4.29 menjelaskan lingkungan operasi yang digunakan oleh sistem dan pengguna. Pada dasarnya lingkungan yang digunakan harus tidak menghambat mobilisasi pada robot. Serta tidak mengganggu jalur komunkasi *wireless* dari kedua komponen komunikasi.

4.2.4.3 Kebutuhan Asumsi dan Ketergantungan

Tabel 4.30 Menjelaskan ketergantungan serta asumsi sistem dan pengguna

Asumsi dan Ketergantungan					
Tipe:	Non-Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi
1. Sistem dapat berfungsi apabila RX dinyalakan terlebih dahulu kemudian TX. Hal ini untuk menghindari *miss* komunikasi data antara kedua perangkat tersebut serta prosedur keamanan menjalankan sistem. 2. Pengiriman data sensor IMU MPU-6050 hanya bisa dilakukan dengan mengatur *channel* dan *payload* dengan mengatur kecepatan data dalam bit per detik (*baud*) untuk transmisi data serial yang terletak pada perangkat RX dan TX. 3. Pergerakan sistem juga bergantung pada sumber daya pada baterai yang digunakan. Terutama pada perangkat RX yang memerlukan daya yang cukup besar untuk menggerakan motor. 4. Penggunaan *wearable device* berupa *gloves* hanya dapat digunakan apabila telah terhubung ke dalam sistem dan sudah dalam keadaan terpakai di tangan. 5. Kecepatan pada saat menggerakkan gestur tangan juga berpengaruh terhadap kecepatan motor DC pada robot beroda.					
Catatan:	Komunikasi data yang diterima oleh sistem dapat mengeksekusi gerakan gestur tangan secara bersamaan. Misalkan maju kedepan sekaligus belok ke kiri. Dibatasi hanya dua gerakan dikarenakan hanya dua parameter yakni *pitch* dan *roll* data yang dikirimkan.				

Pada Tabel 4.30 diatas menjelaskan bagaiman asumsi dan ketergantungan sangat dibutuhkan dalam menjalankan sistem. Tanpa adanya informasi teknis mengenai ketergantungan sistem maka pengguna akan sulit untuk menjalankan dan mengontrol sistem dengan benar.

4.2.4.4 Kebutuhan Batasan Perancangan dan Implementasi

Tabel 4.31 Menjelaskan batasan perancangan dan implementasi

Kebutuhan Perancangan dan Implementasi					
Tipe:	Non-Fungsional	**Prioritas:**	Tinggi	**Metode Verifikasi:**	Inspeksi

	1. Dirancang dalam kerangka berbahan akrilik untuk memudahkan dalam hal pementukkan bodi robot yang menyerupai mobil serta bahan yang cukup kuat. 2. Desain Robot beroda dengan empat roda penggerak untuk memaksimalkan mobilisasi dan memaksimalkan tenaga untuk menambah kecepatan. 3. *Gloves* sebagai media untuk menggerakkan sensor agar lebih fleksibel dan mudah digunakan. 4. Implementasi gerakan gestur tangan yang akan dideteksi direpresentasikan dalam gerakan Robot beroda untuk gerakan *pitch* sama dengan maju-mundur dan gerakan *roll* yakni kiri-kanan. 5. Lama bergerak sesuai dengan masukkan pada gestur tangan pengguna. Misal : Apabila gestur tangan pengguna ditekuk kebawah selama 5 detik maka Robot beroda juga akan bergerak maju selama 5 detik 6. Kinerja pada sistem juga harus cepat, efektif dan akurat. Waktu tanggap dan waktu tunda dalam menjalankan sistem mengendalikan robot beroda harus diperhatikan. Waktu tunda harus diminimalisir seminim mungkin agar kinerja sistem lebih responsif.
Catatan:	Perancangan sistem melalui tahap desain terlebih dahulu untuk menciptakan sistem yang sesuai dan proporsional ketika digunakan

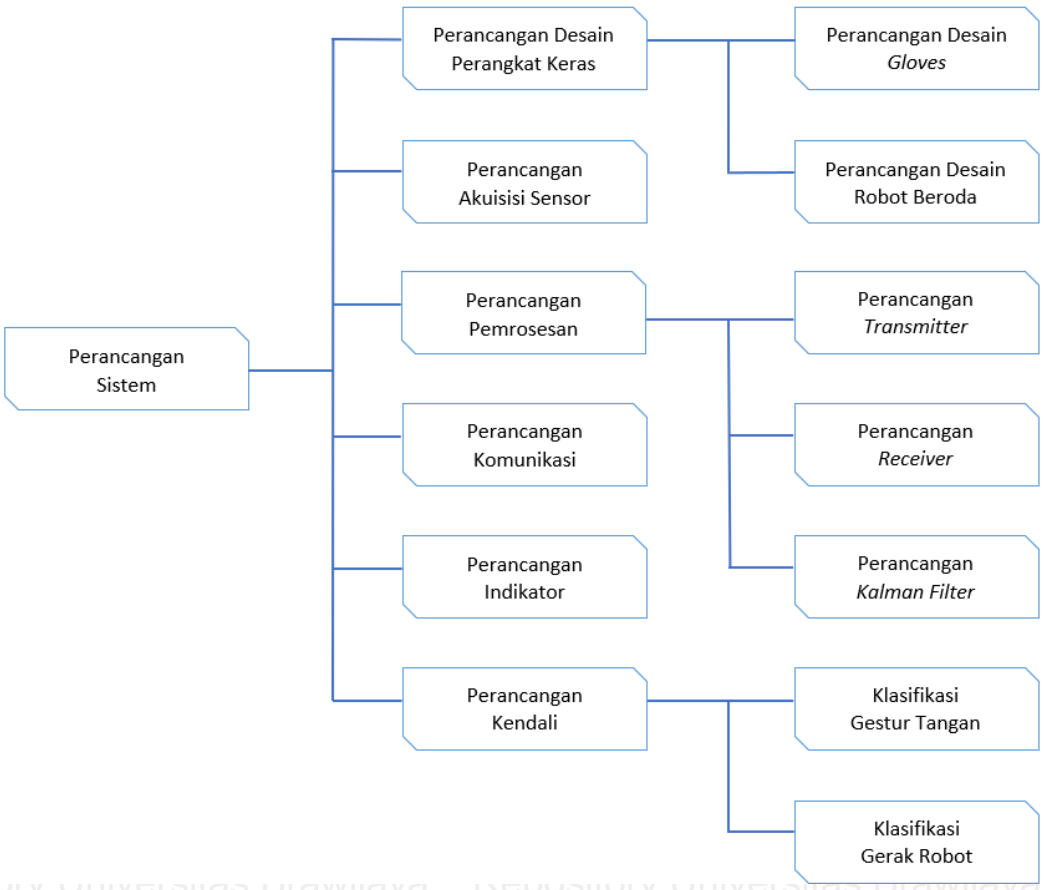
Pada Tabel 4.31 diatas dijelaskan mengenai kebutuhan batasan perancangan dan implementasi yang dijadikan acuan dalam membangun sistem. Fungsinya adalah untuk mehami teknis penelitian yang jauh lebih spesifik.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan menjelaskan proses perancangan sistem hingga bagaimana sistem tersebut dapat di implementasikan. Perancangan dan implementasi bertujuan untuk mengetahui bagaimana gambaran sistem yang akan dibuat serta implementasi segala komponen hardware maupun software agar terintegrasi dan dapat diterapkan sesuai dengan kebutuhan sistem. Pada tahap ini dibagi menjadi dua bagian yakni perancangan sistem dan implementasi sistem agar dapat menjelaskan secara *detail* juga mudah dipahami.

5.1 Perancangan Sistem

Perancangan sistem merupakan metode menganalisis, mendefinisikan dan merancang arsitektur informasi organisasi pada suatu sistem. Dalam proses perancangan semua rencana yang berkaitan dengan sistem harus ditulis secara rinci. Baik itu desain, struktur, bentuk dan komponen apa saja yang digunakan untuk membentuk suatu sistem yang sesuai dengan kebutuhan. Oleh karena itu dalam tahap ini perancangan dibagi menjadi enam bagian seperti pada Gambar 5.1



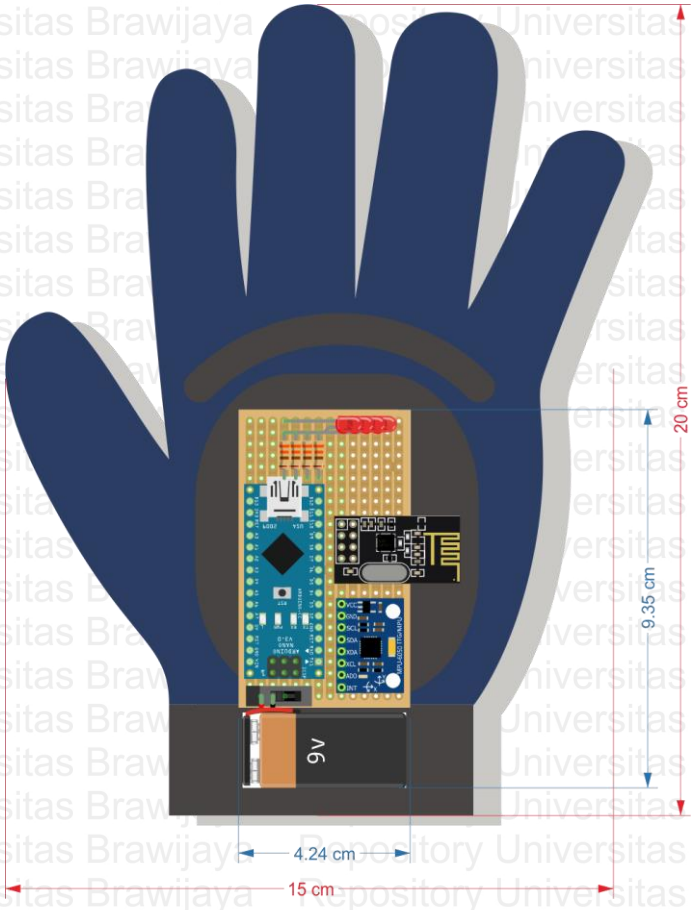
Gambar 5.1 Diagram alir perancangan sistem

5.1.1 Perancangan Desain Perangkat Keras

Desain perangkat keras adalah sebagai langkah awal perencanaan dalam pembuatan sebuah objek, sistem, komponen atau struktur agar sesuai dengan kegunaan,fungsi benda, ketetapan pemilihan bahan dan memperhatikan segi keindahan. Pada bagian desain perangkat keras dibagi menjadi dua bagian yakni desain *gloves* dan Robot beroda.

5.1.1.1 Desain *Gloves*

Gloves atau sarung tangan merupakan perangkat pengendali pada sistem yang digunakan oleh pengguna untuk mengatur gerak robot beroda. Pada tahap ini berisi tentang desain *gloves* yang dibutuhkan oleh sistem. Tujuan utama digunakannya *gloves* adalah untuk memudahkan persiapan pada saat uji coba atau mengendalikan sistem. Cara menggunakan *gloves* yakni hanya dengan memasangkan *gloves* pada tangan kanan pengguna. Pada tahap ini juga membahas mengenai penggabungan *gloves* dengan perangkat pengendali dan ukuran serta posisi yang tepat untuk menunjang kinerja sistem.



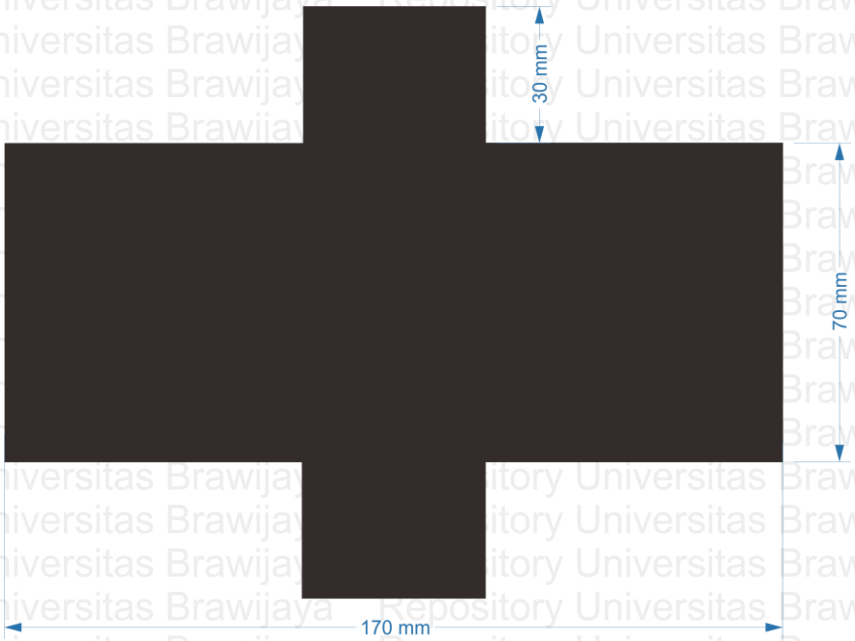
Gambar 5.2 Desain *gloves*

Pada gambar 5.2 terdiri dari *gloves* dan perangkat pengendali serta dimensi dari keduannya. Untuk ukuran *gloves* yakni 20 x 15cm sesuai dengan

ukuran normal tangan orang dewasa. Dalam hal ini *gloves* yang digunakan adalah sebelah kanan. Untuk ukuran perangkat pengendali yakni 4.24 x 9.35cm. Dengan ukuran perangkat pengendali yang tidak cukup besar sehingga cocok diletakkan dibagian atas telapak tangan pengguna dari pada dibagian tubuh manusia yang lain. Sehingga memudahkan dalam pengendalian dan gerakan gestur tangan menjadi lebih akurat serta objektif.

5.1.1.2 Desain Robot beroda

Robot beroda merupakan komponen *transmitter* serta objek yang dikendalikan pada sistem. Robot beroda dalam hal ini berbentuk mobil dengan empat roda penggerak didalamnya. Desain Robot beroda berfungsi untuk merancang ukuran serta peletakkan komponen-komponen elektro didalamnya sehingga dapat sesuai dengan fungsi dan kegunaannya. Desain juga menentukan kinerja dari Robot beroda itu sendiri, dengan desain yang proporsional serta mempertimbangkan tenaga yang dihasilkan oleh modul penggerak seperti motor DC. Dengan memperhatikan hal tersebut desain Robot beroda tidak boleh menghambat kinerja dari komponen yang lain. Berikut adalah gambaran dari desain Robot beroda.

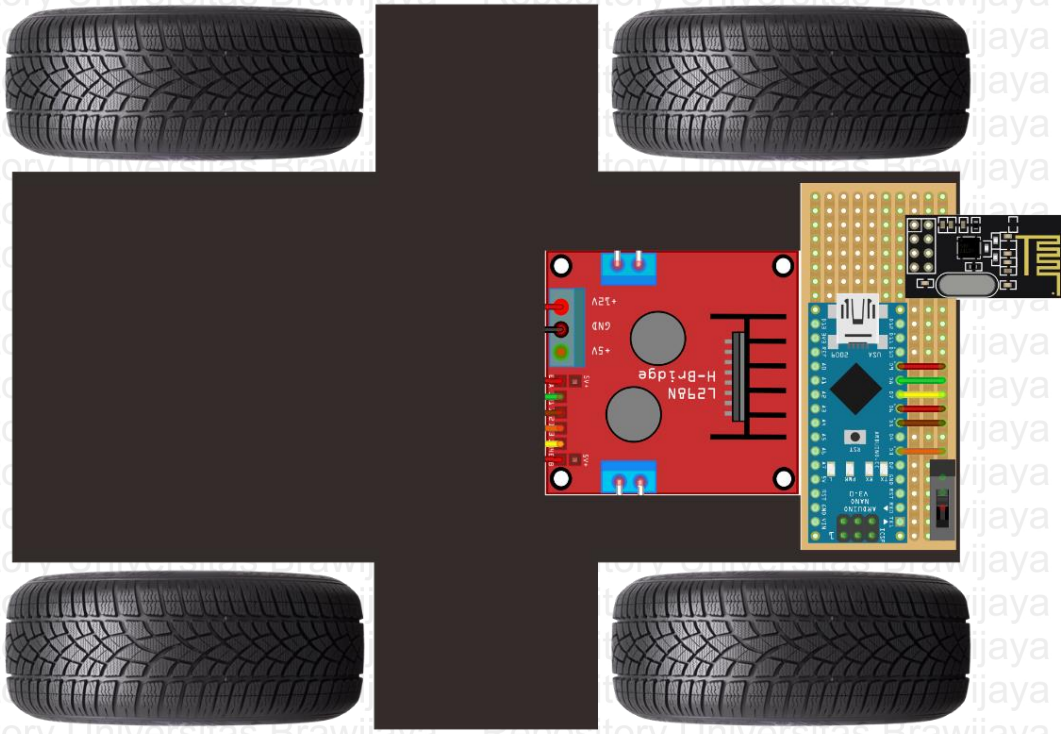


Gambar 5.3 Desain penampang Robot beroda

Pada Gambar 5.3 diatas merupakan desain penampang Robot beroda yang terbuat dari bahan akrilik dengan tebal 3mm. Dimensi yang digunakan adalah 170mm x 130mm yang cukup untuk menampung seluruh komponen elektronik pada sistem nantinya. Berbentuk seperti tanda *plus* atau tambah dikarenakan empat sisi dari penampang akan diisi dengan roda dengan ukuran 70mm x 30mm.

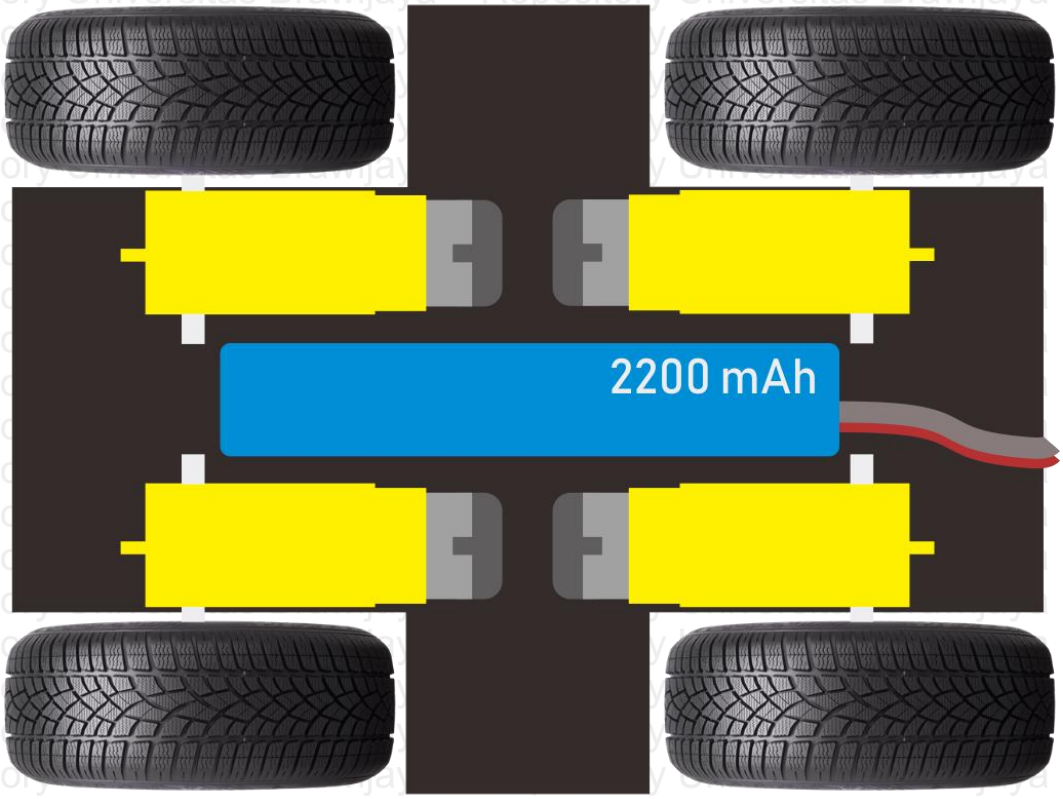
Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Kemudian berikut adalah gambaran desain penampang setelah diletakkan komponen elektronika.



Gambar 5.4 Desain Robot beroda (tampak atas)

Gambar 5.4 menunjukkan penampakkan desain Robot beroda dari atas. Pada desain tersebut terdapat komponen elektronika berupa Arduino Nano V3, Modul NRF24L01 *receiver*, saklar, roda dan driver motor L298N. Komponen elektronika terletak disebelah kanan karena merupakan bagian belakang pada Robot beroda yang mana bagian kiri merupakan bagian depan. Dimensi penampang dan dimensi komponen yang telah disesuaikan sehingga ukuran Robot beroda tersebut pas dan sesuai dengan kebutuhan sistem. Untuk letak modul penggerak seperti motor DC serta daya untuk sistem terletak dibagian bawah penampang.

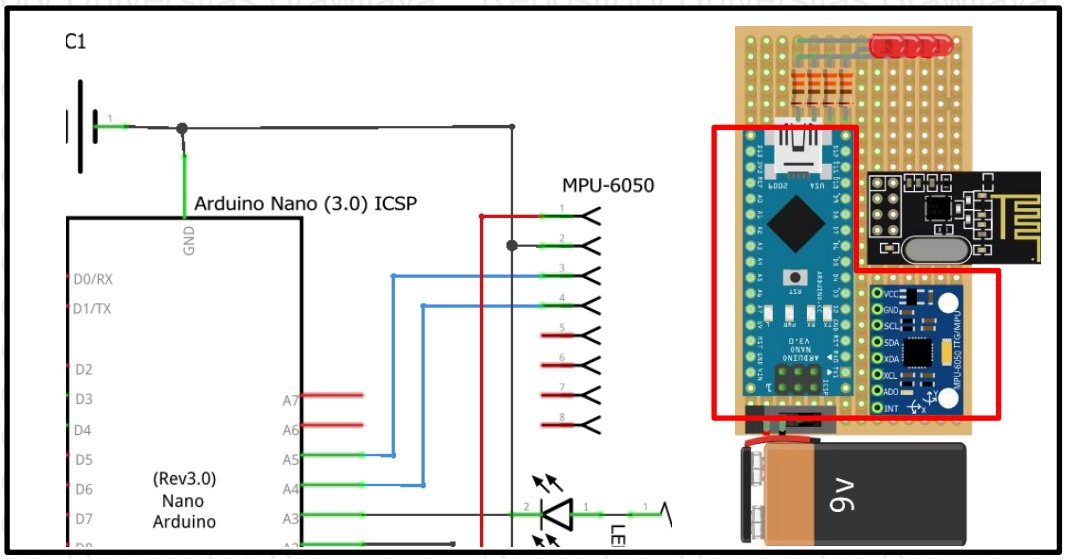


Gambar 5.5 Desain Robot beroda (tampak bawah)

Gambar 5.5 Desain tampak bawah dari penampang memperlihatkan komponen penggerak sistem seperti motor DC dan keempat roda serta sumber daya yakni baterai Li-Po (Lithium Polymer) 2200 mAh. Terletak dibawah dan tersembunyi dikarenakan roda motor dan dinamo sebagai penahan beban keseluruhan komponen yang ada diatasnya serta tersembunyi untuk meningkatkan nilai estetika dari Robot beroda dan juga untuk mengoptimalkan segala sisi dan ruang yang terbatas.

5.1.2 Perancangan Akuisisi Sensor

Akuisisi sensor merupakan tahap awal sebelum pemrosesan pada sistem. Pada tahap ini merupakan perancangan bagaimana sistem dapat mengakuisisi nilai pada sensor *accelerometer* MPU-6050. Melibatkan dua komponen utama dalam akuisisi nilai sensor yakni sensor dan mikrokontroler Arduino nano. Berikut adalah gambaran skematik dari kedua komponen tersebut.



Gambar 5.6 Skematik akuisisi sensor dan desain *transmitter*

Gambar 5.6 Menampilkan skematik akuisis sensor dan juga desain dari kedua komponen tersebut. Dalam hal ini, sensor MPU-6050 menghasilkan nilai berupa nilai X,Y dan Z. Namun yang dibutuhkan oleh sistem hanya data X dan Y. Seperti pemetaan pin pada Tabel 5.1 berikut.

Tabel 5.1 Skema pengkabelan perangkat akuisisi sensor

	Pin MPU-6050	Pin Arduino Nano V3
Nama Pin	VCC	VCC 3V
	GND	GND
	SCL	A4
	SDA	A5
	XDA	-
	XCL	-
	AD0	-
	INT	-

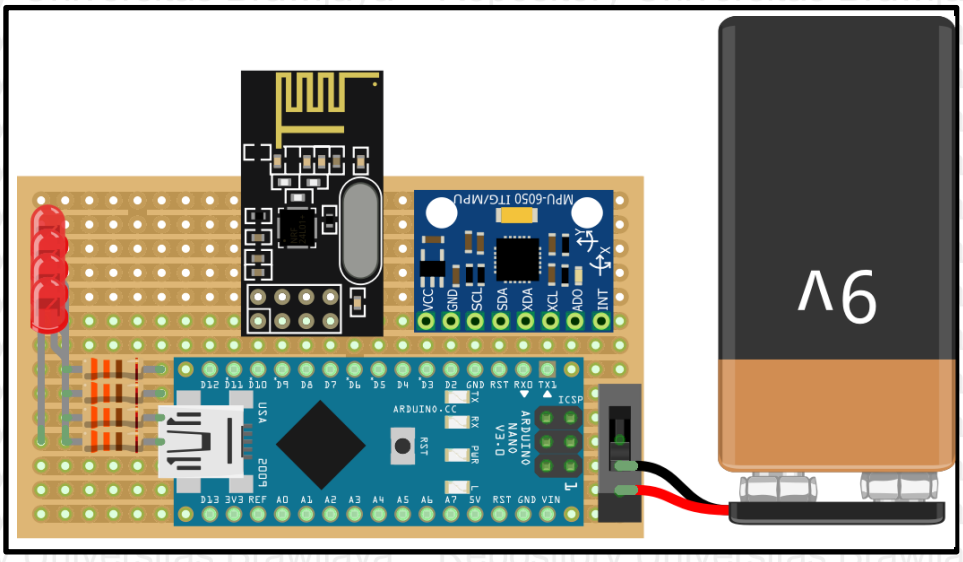
Pada Tabel 5.1 diatas, diketahui bahwa MPU-6050 terdiri dari 8 pin dimana yang digunakan dalam penilitian ini hanya pin VCC,GND,SCL dan SDA. Masing-masing pin memiliki fungsi tertentu yakni VCC sebagai sumber tegangan, GND berfungsi untuk meniadakan beda potensial, SCL (Serial Clock) merupakan saluran untuk menghantarkan sinyal clock sedangkan SDA (Serial Data) membawa informasi data antara I2C dengan pengontrolnya.

5.1.3 Perancangan Pemrosesan

Perancangan pemrosesan disini adalah rancangan pengolahan pada sistem dimulai dari *input* sampai dengan *output*. Dibagi menjadi tiga bagian yakni perancangan *transmitter*, perancangan *receiver* dan perancangan *kalman filter*.

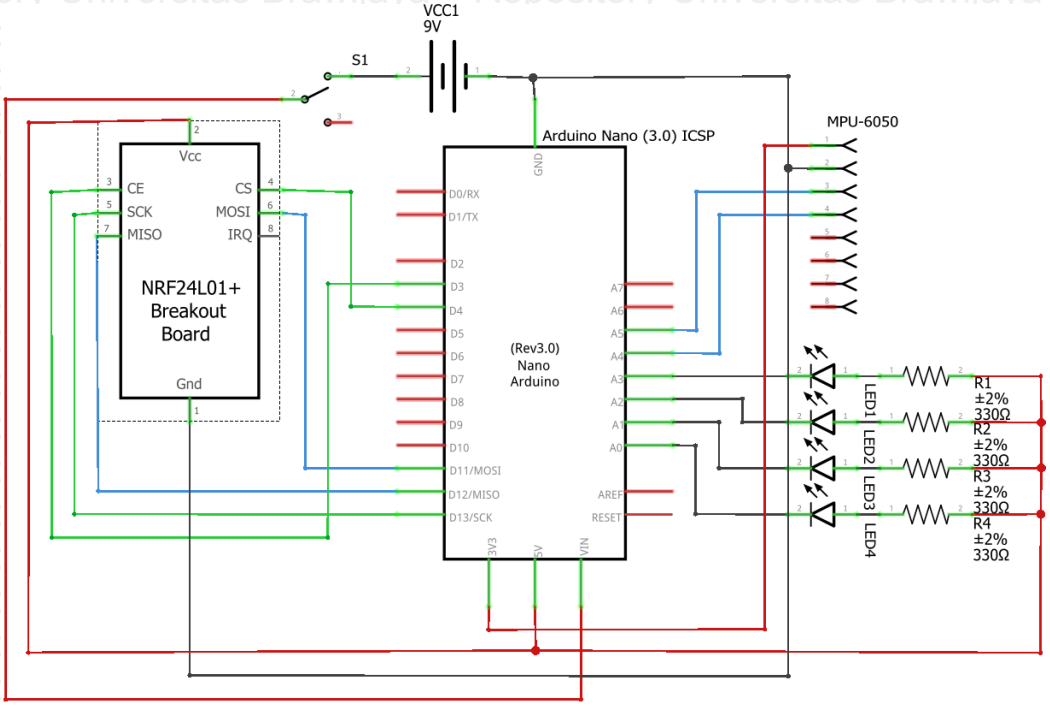
5.1.3.1 Perancangan *Transmitter*

Perancangan *transmitter* merupakan perancangan komponen pengendali atau Tranmission Data (TX) yang mana merupakan komponen yang berfungsi sebagai akuisisi dan pengolahan nilai sensor kemudian nilai tersebut dikirimkan ke perangkat *Receiver*. Dalam tahap ini akan menjelaskan bagaimana perangkat *transmitter* dibentuk serta komponen apa saja yang dibutuhkan dan bagaimana hubungan antar komponennya. *Transmitter* disini dibentuk dengan dimensi sekecil mungkin karena akan digunakan sebagai weareble *device* atau perangkat yang digunakan pada tubuh manusia. Komponen yang digunakan harus dapat memaksimalkan ruang pada *circuit board* namun tidak mengurangi nilai fungsi dari komponen tersebut. Hubungan antar komponen juga harus efektif dan efisien agar dapat menghemat ruang. Berikut pada Gambar 5.7 dan Gambar 5.8 adalah gambaran desain TX dan skematik perangkat *transmitter*.



Gambar 5.7 Desain dari perangkat *Transmitter*

Gambar 5.7 diatas merupakan desain perangkat *Transmitter* (TX) yang akan diletakkan pada *gloves*. Di desain dengan seminimal mungkin dengan memanfaatkan ruang yang terbatas pada PCB sehingga menciptakan dimensi yang tidak terlalu besar dan dapat di implementasikan pada *gloves*. Perangkat TX terdiri dari delapan komponen yakni arduino nano V3, NRF24L01, MPU-6050, saklar, resistor, lampu LED, PCB dan baterai. Untuk mengetahui secara detail hubungan antar komponen pada Gambar 5.8 merupakan gambaran skematik dari perangkat TX.



Gambar 5.8 Skematik perangkat *Transmitter*

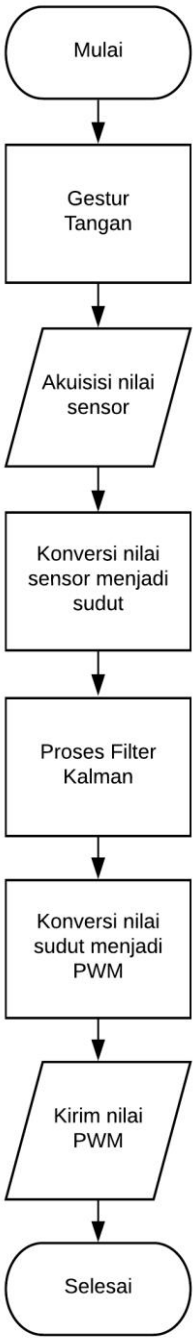
Pada Gambar 5.8, skematik perangkat *transmitter* (TX) diatas terdiri dari beberapa komponen. Masing-masing komponen memiliki fungsi dan kegunaan tertentu. Sensor MPU-6050 berfungsi sebagai pendeteksi gestur tangan pengguna kemudian data diolah oleh mikrokontroler arduino nano. Pengolahan didalamnya meliputi proses filter dan konversi nilai sensor tersebut sebelum dikirim melalui modul komunikasi NRF24L01. Kemudian lampu LED menjadi indikator dalam pergerakan robot beroda dan gestur tangan pengguna sehingga memudahkan dalam mengontrol pergerakan robot beroda. Keseluruhan daya komponen pada perangkat TX bersumber dari baterai 9-volt. Prinsip pengkabelan pada komponen-komponen penyusun perangkat TX tercantum pada Tabel 5.2.

Tabel 5.2 Skema pengkabelan perangkat *transmitter*

	Pin Arduino Nano V3	Pin MPU-6050	NRF24L01	LED
Nama Pin	VCC 3V	VCC	-	-
	VCC 5V	-	VCC	-
	VIN	-	-	-
	GND	GND	GND	*Chatode*
	A5	SCL	-	-
	A4	SDA	-	-
	A3	-	-	*Anode* LED1

	A2	-	-	*Anode* LED2
	A1	-	-	*Anode* LED3
	A0	-	-	*Anode* LED4
	D3	-	CE	-
	D4	-	CS	-
	D11/MOSI	-	MOSI	-
	D12/MISO	-	MISO	-
	D13/SCK	-	SCK	-

Terdapat juga diagram alir tahapan proses yang dilakukan dalam perangkat TX untuk mempermudah memahami alur proses dan cara bekerjanya. Tahap awal yakni pengguna mulai menggunakan perangkat TX untuk mengendalikan Robot beroda sesuai dengan ketentuan. Gestur tangan yang tercipta dibaca oleh sensor MPU-6050 kemudian data diakuisisi nilai *pitch* dan *roll* pada *accelerometer* dan *gyroscope*. Kemudian data akan diteruskan ke mikrokontroler arduino nano V3 untuk melakukan konversi nilai sensor menjadi nilai sudut. Nilai yang telah berubah menjadi sudut kemudian melalui tahap filter dengan menggunakan metode *kalman filter*. Filter sinyal akan berjalan apabila nilai telah diubah menjadi bentuk sudut dan laju harus dalam satuan derajat/detik. Proses filter kalman bermanfaat untuk meningkatkan akurasi dan meredam *noise*. Selanjutnya sinyal yang telah difilter diubah/dipetakan kedalam bentuk PWM lalu kemudian dikirim melalui modul komunikasi NRF24L01 *transmitter* ke perangkat NRF24L01 *receiver*. Gambar 5.9 memperlihatkan diagram alir proses yang terjadi pada perangkat pengendali atau TX.

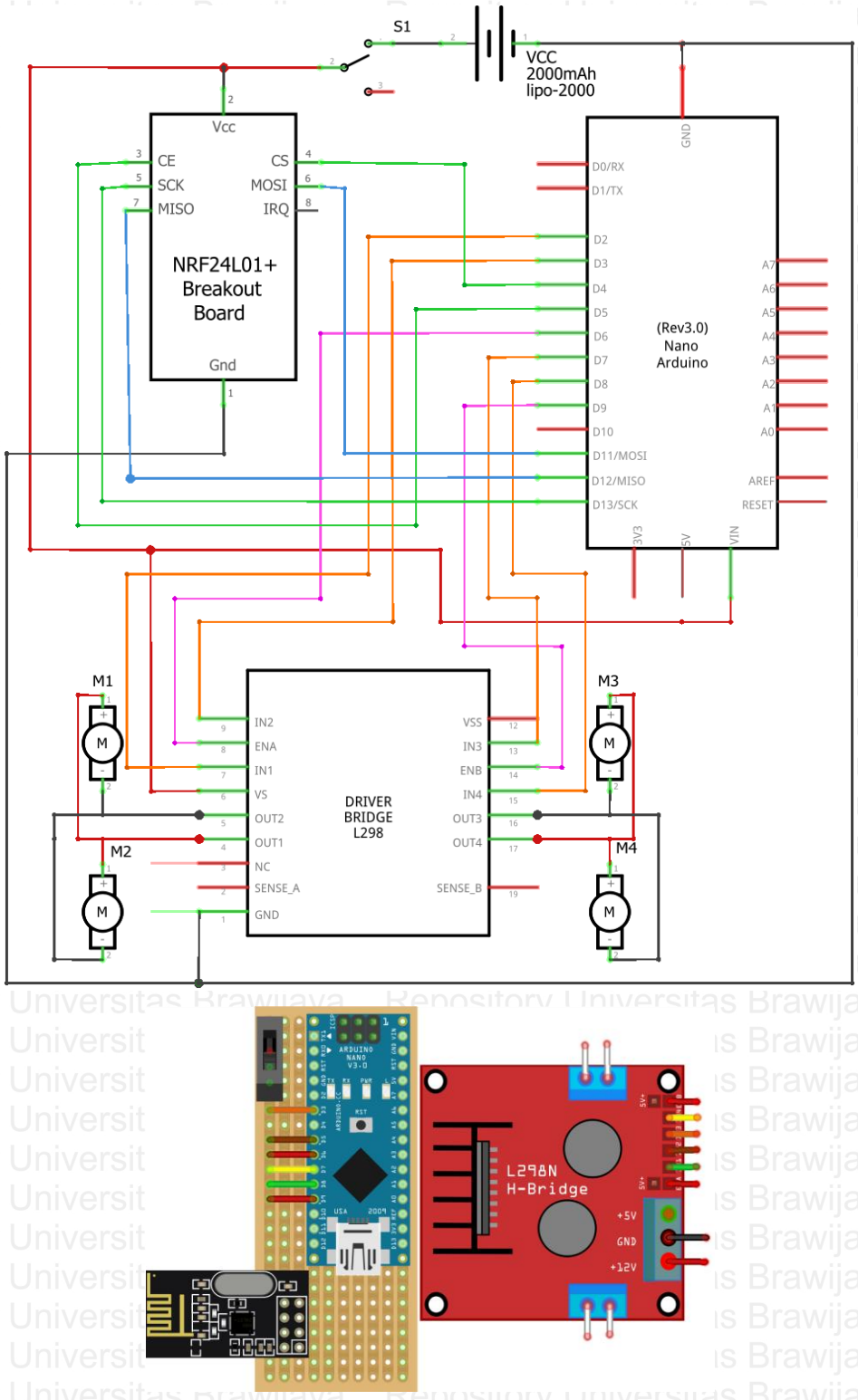


Gambar 5.9 Diagram alir perangkat *transmitter*

5.1.3.2 Perancangan *Receiver*

Pada tahap ini perancangan *receiver* berguna untuk menerima sinyal data yang dikirimkan oleh *transmitter*. Kemudian data yang telah diterima akan diolah oleh mikrokontroler arduino nano yang terletak pada perangkat *receiver*. Sehingga dapat mengendalikan motor DC pada perangkat tersebut sesuai dengan *input*

yang diterima pada *receiver*. Berikut ini pada Gambar 5.10 akan menampilkan skematik dari perangkat *receiver*.



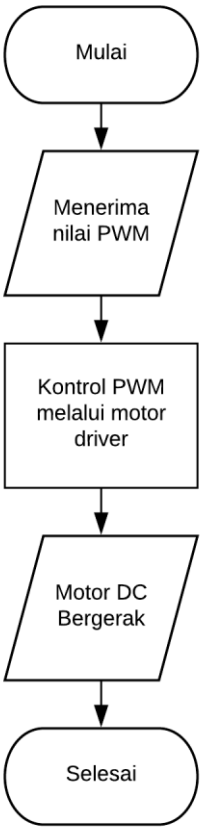
Gambar 5.10 Skematik dan desain perangkat *receiver*

Pada Gambar 5.10 diatas merupakan skematik dan desain pemorosesan pada perangkat *receiver* yang terdapat pada Robot beroda. Pada desain dan skematik diatas tersusun dari beberapa komponen elektronik, diantaranya adalah arduino nano v3, modul komunikasi NRF24L01, Driver motor L298D, 4 motor DC, saklar, PCB dan baterai Lithium-Polymer 2000 mAh. Komponen-komponen tersebut memiliki fungsi dan peran tertentu dalam perangkat *receiver*. NRF24L01 sebagai modul penerima sinyal data dari *transmitter*. Dalam hal ini perangkat *receiver* (RX) bertindak sebagai *master* sebagai penerima data dari *slave* (Perangkat *transmitter*). Arduino nano V3 bertugas memproses sinyal yang masuk. Driver motor dan motor DC bekerja sebagai aktuator dan merupakan ujung dari proses panjang kendali gerakan berbasis Robot beroda. Komponen perangkat keras pada robot beroda disusun berdasarkan pemetaan pengkabelan seperti yang tercantum pada Tabel 5.3.

Tabel 5.3 Skema pengkabelan perangkat *receiver*

	Pin Arduino Nano V3	***Motor Driver***	**Motor DC**	**NRF24L01**	**Li-Po *Battery***
Nama Pin	VIN	VS	-	VCC	*Anode*
	GND	GND	-	GND	*Chatode*
	D2	IN1	-	-	-
	D3	IN2	-	-	-
	D4	-	-	CS	-
	D5	-	-	CE	-
	D6	ENA	-	-	-
	D7	IN3	-	-	-
	D8	IN4	-	-	-
	D9	ENB	-	-	-
	D11	-	-	MOSI	-
	D12	-	-	MISO	-
	D13	-	-	SCK	-
	-	OUT1	*Anode* M1 *Anode* M2	-	-
	-	OUT2	*Chatode* M1 *Chatode* M2	-	-
	-	OUT3	*Anode* M3 *Anode* M4	-	-
	-	OUT4	*Chatode* M3 *Chatode* M4	-	-

Secara garis besar sistem bekerja menerima sinyal data dari komponen NRF24L01 kemudian data akan di proses ke dalam mikrokontroler arduino nano V3. Sinyal yang dikirim berupa nilai PWM maka peran arduino disini adalah memproses sinyal sehingga dapat mengendalikan *motor driver* untuk menggerakkan motor DC. Sehingga motor DC dapat bergerak sesuai dengan *input* pada gestur tangan yang sebelumnya telah di proses pada perangkat *transmitter*. Untuk lebih jelasnya terdapat diagram alir pada Gambar 5.11 dibawah ini.



Gambar 5.11 Diagram alir perangkat *receiver*

5.1.3.3 Perancangan *Kalman filter*

Kalman filter memiliki algoritma estimasi menyerupai prediksi-koreksi untuk menyelesaikan masalah numerik. Karena pada dasarnya *kalman filter* mengestimasi suatu proses melalui mekanisme umpan-balik yang bekerja mengestimasi *state* dari proses kemudian mendapat upan balik berupa nilai hasil pengukuran yang bercampur *noise*. *Kalman filter* menggunakan serangkaian pengukuran yang diamati dari waktu ke waktu, dalam konteks perancangan ini adalah *accelerometer* dan *gyroscope*. Pengukuran ini akan mengandung *noise* yang akan berkontribusi pada kesalahan pengukuran.

Dalam konteks ini masalahnya adalah bahwa *accelerometer* secara umum sangat banyak menghasilkan *noise* ketika digunakan untuk mengukur percepatan gravitasi karena robot bergerak bolak-balik. Masalah dengan *gyro* adalah seperti melayang seiring berjalannya waktu sama seperti roda berputar, *gyro* akan mulai jatuh ketika kehilangan kecepatan.

Singkatnya dalam perancangan proses filter penulis mengatasi masalah tersebut dengan mengimplementasikan *complementary filter* atau *low-pass filter* dan *high-pass filter* untuk proses *sensor fusion* (penggabungan sensor *accelero* dan *gyro*). *Complementary filter* mengelola *filter high-pass* dan *low-pass* secara bersamaan. Low-pass *filter* menyaring sinyal frekuensi tinggi (seperti *accelerometer* jika terjadi getaran) dan *low-pass filter* yang menyaring sinyal frekuensi rendah (seperti penyimpangan *gyroscope*).

Pada bagian ini akan dirancang menggunakan persamaan untuk menerapkan filter ke dalam kode "C" sederhana yang dapat digunakan untuk menstabilkan robot beroda. Data sensor untuk sistem diambil dari titik *sampel* waktu diskrit baik itu pengukuran *noise* atau *noise* proses. *Noise* proses dan *noise* pengukuran dapat digambarkan dengan *matrix covariance* Q dan R seperti yang tercantum pada persamaan (5.1) dan (5.2).

covariance noise process $$Q = E\left[w_k \quad {w_k}^T\right] = var(w_k) \qquad (5.1)$$

and mesurenment $$R = E\left[v_k \quad {v_k}^T\right] = var(v_k) \qquad (5.2)$$

Keterangan :

T adalah *matrix transpose*
E adalah *estimation output*
Q adalah *covariance noise process*
R adalah *covariance noise*
v adalah *measurement variance noise*
w adalah *process noise*

Step 1

Time update (Prediction)

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + B\dot{\theta}_k \qquad (5.3)$$

$$\begin{bmatrix}\theta \\ \dot{\theta}_b\end{bmatrix}_{k|k-1} = \begin{bmatrix}1 & -\Delta t \\ 0 & 1\end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta}_b\end{bmatrix}_{k-1|k-1} + \begin{bmatrix}\Delta t \\ 0\end{bmatrix}\dot{\theta}_k \qquad (5.4)$$

$$\dots$$

$$= \begin{bmatrix}\theta + \Delta t(\dot{\theta} - \dot{\theta}_b) \\ \dot{\theta}\end{bmatrix} \qquad (5.5)$$

Dalam tiga persamaan pertama kita akan mencoba untuk memprediksi keadaan saat ini dan *covariance error matrix* pada waktu k. Pertama, filter akan

mencoba memperkirakan kondisi saat ini berdasarkan semua status sebelumnya dan pengukuran *gyro* pada persamaan (5.3).

Digunakan sebagai *input* tambahan untuk memperkirakan keadaan saat ini k yang disebut keadaan *apriori* $\hat{x}_{k|k-1}$. $\hat{x}_{k|k-1}$ Keadaan *priori* dimana estimasi *matrix* keadaan pada saat saat ini k berdasarkan pada keadaan sistem sebelumnya dan perkiraan keadaan sebelumnya. Diperoleh dengan mengalikan parameter atau notasi F dengan $\hat{x}_{k-1|k-1}$ yaitu merupakan estimasi sebelumnya. Kemudian menjumlahkannya dengan pengukuran pada *bias* $B\dot{\theta}_k$. Berarti bahwa mendapatkan *rate* yang sebenarnya dengan mengurangi *bias* dari pengukuran *gyro.* Sehingga hasil transisi model dalam bentuk matrix adalah persamaan (5.4) dan (5.5).

Step 2

Update estimation error covariance (Prediction)

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q_k \quad \textbf{(5.6)}$$

$$\dots$$

$$= \begin{bmatrix} P_{00} + \Delta t(\Delta t P_{11} - P_{01} - P_{10} + Q_\theta) & P_{01} - \Delta t P_{11} \\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b}\Delta t \end{bmatrix} \quad \textbf{(5.7)}$$

Pada persamaan (5.6) menunjukkan proses *Update estimation error covariance.* Memproyeksikan *matrix error covariance* sebelum ke tahap koreksi. Hal berikutnya adalah mencoba memperkirakan *matrix error covariance priori* $P_{k|k-1}$berdasarkan pada *matrix error covariance* sebelumnya $FP_{k-1|k-1}$.

Prinsip persamaan di atas sebenarnya cukup mudah dimengerti, karena cukup jelas bahwa *error covariance* akan meningkat sejak terakhir kali memperbarui estimasi keadaan, oleh karena itu perlu mengalikan *matrix error covariance* dengan model transisi keadaan F dan *transpose* dari F^T dan menjumlahkan dengan *noise proses* Q_k saat ini pada waktu k. Hasil transisi persamaan (5.6) adalah pada persamaan (5.7).

Step 3

Calculation angle and bias (Correction)

$$\tilde{y}_k = Z_k - H\hat{x}_{k|k-1} \quad \textbf{(5.8)}$$

$$\dots$$

$$= z_k - \theta_{k|k-1} \quad \textbf{(5.9)}$$

Hal pertama yang akan kita lakukan adalah menghitung perbedaan antara pengukuran Z_k dan status *apriori* $\hat{x}_{k|k-1}$ seperti yang ditunjukkan oleh persamaan (5.8). Model observasi H digunakan untuk memetakan keadaan *apriori* $\hat{x}_{k|k-1}$ ke dalam ruang pengamatan yang merupakan pengukuran dari *accelerometer*. Hasil transisi persamaan (5.8) dapat berupa persamaan (5.9).

Step 4

Measurement update estimation error (Correction)

$$S_k = HP_{k|k-1}H^T + R \quad \textbf{(5.10)}$$

...

$$P_{00k|k-1} + var(v) \quad \textbf{(5.11)}$$

Hal berikutnya yang akan dilakukan adalah menghitung apa yang disebut inovasi *covariance* pada persamaan (5.10). Pada persamaan tersebut mencoba untuk memprediksi berapa banyak kita harus mempercayai pengukuran berdasarkan *apriori matrix error covariance* $P_{k|k-1}$ dan *matrix error covariance* R. Model pengamatan H digunakan untuk memetakan *apriori matrix error covariance* $P_{k|k-1}$ ke dalam ruang yang diamati. Semakin besar nilai noise pengukuran, semakin besar nilai S_k, ini berarti kita tidak terlalu mempercayai pengukuran yang masuk. Dalam hal ini S bukan *matrix* sehingga menghasilkan transisi akhir pada persamaan (5.11).

Step 5

Calculate Kalman gain (Correction)

$$K_k = P_{k|k-1}H^T S_k{}^{-1} \quad \textbf{(5.12)}$$

...

$$\frac{\begin{bmatrix} P_{01} \\ P_{10} \end{bmatrix}_{k|k-1}}{S_k} \quad \textbf{(5.13)}$$

Langkah selanjutnya adalah menghitung *Kalman gain* pada persamaan (5.12). Keuntungan Kalman digunakan untuk menunjukkan seberapa besar mempercayai proses pada persamaan tersebut. Dapat dilihat bahwa jika kita tidak mempercayai inovasi yang banyak inovasi *covariance* S akan tinggi dan jika kita mempercayai *estimation state* maka *matrix error covariance* P akan kecil, maka keuntungan Kalman akan menjadi kecil dan berlawanan jika kita percaya inovasi tetapi tidak mempercayai *estimation state* saat ini.

Jika melihat lebih dalam, *transpose* model observasi H digunakan untuk memetakan keadaan *matrix error covariance* P ke dalam ruang yang diamati. kemudian membandingkan *matrix error covariance* dengan mengalikan dengan kebalikan dari inovasi *covariance* S. Sehingga menghasilkan persamaan (5.13). Ini masuk akal karena penulis akan menggunakan model observasi H untuk mengekstraksi data dari *error covariance* state dan membandingkannya dengan setimasi saat ini dari *covariance* inovasi.

Step 6

Calculate angle and bias

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y}_k \quad \textbf{(5.14)}$$

...

From kalman gain and Angle difference

$$\begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_{k|k-1} + \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}_k \tilde{y} \quad \textbf{(5.15)}$$

(Update Posteriori Estimate)

Persamaan (5.14) Ini dilakukan dengan menambahkan status *apriori* $\hat{x}_{k|k-1}$ dengan *Kalman gain* dikalikan dengan inovasi $\tilde{y}_k$. Ingatlah pada persamaan sembelumnya bahwa inovasi $\tilde{y}_k$ adalah perbedaan antara pengukuran Z_k dan perkiraan keadaan *apriori* $\hat{x}_{k|k-1}$, sehingga inovasi bisa berdampak positif dan negatif.

Sedikit persamaan yang disederhanakan dapat dipahami karena cukup memperbaiki perkiraan keadaan *apriori* $\hat{x}_{k|k-1}$, yang dihitung menggunakan status sebelumnya dan pengukuran *gyro*, dengan pengukuran, dalam hal ini adalah *accelerometer*. Diperoleh transisi dalam bentuk *matrix* pada persamaan (5.15).

Step 7

Update posteriori error covariance matrix

$$P_{k|k} = (I - K_k H) P_{k|k-1} \quad \textbf{(5.16)}$$

...

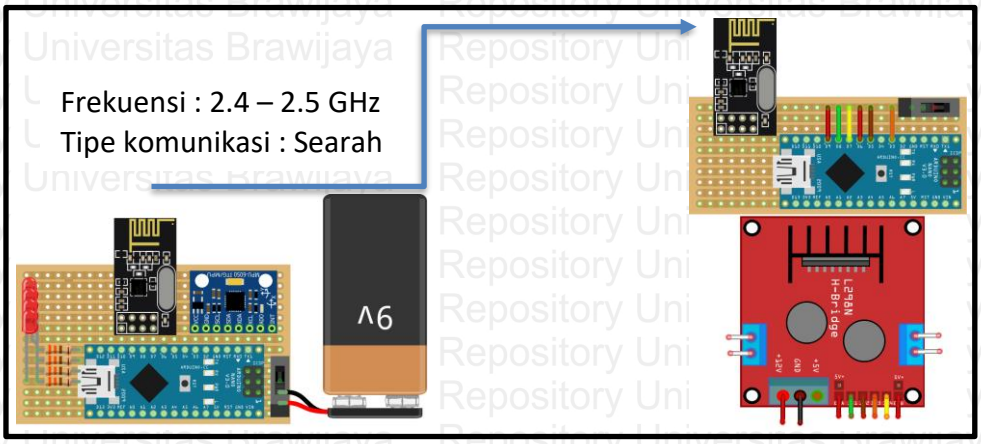
$$\begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix}_{k|k-1} - \begin{bmatrix} K_0 P_{00} & K_0 P_{01} \\ K_1 P_{00} & K_1 P_{01} \end{bmatrix} \quad \textbf{(5.17)}$$

Hal terakhir yang akan dilakukan adalah *update matrix error covariance posteriori* seperti yang ditunjukkan pada persamaan (5.16). Dimana $\boldsymbol{I}$ disebut dengan matrix identitas. Apa yang dilakukan filter adalah pada dasarnya mengoreksi sendiri *matrix error covariance* berdasarkan seberapa banyak kita mengoreksi estimasi. Ini masuk akal ketika kita memperbaiki keadaan berdasarkan *apriori matrix error covariance* $\boldsymbol{P_{k|k-1}}$, tetapi juga inovasi *covariance* $\boldsymbol{S_k}$. Menghasilkan transisi *matrix* pada persamaan (5.17).

Tahap diatas merupakan *step* proses filter yang terdapat pada *library kalman filter*. Setiap langkah pada proses tersebut dapat ditransformasikan dalam bentuk bahasa pemograman. Sehingga pada bab implementasi kalman filter akan membahas mengenai transformasi bentuk matematika diskrit dari kalman filter serta penerapan library kalman filter untuk menentukan variabel *Q_angle*, *Q_bias* dan *R_measure* dengan metode *try and error*.

5.1.4 Perancangan Komunikasi

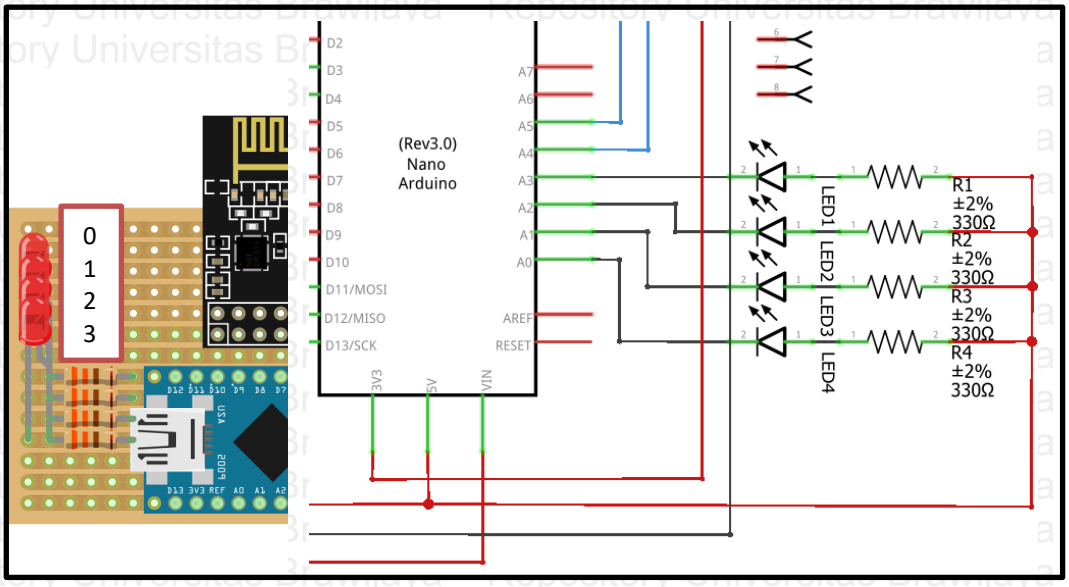
Perancangan komunikasi merupakan perancangan hubungan komunikasi antara perangkat TX dengan RX. Dimana data yang dikirim merupakan data yang telah diolah olah perangkat TX. Media komunikasi menggunakan modul NRF24L01 *transmitter* dan *receiver* dengan frekuensi 2.4-2.5 Ghz. Komunikasi dilakukan secara satu arah yakni dari TX menuju RX seperti yang ditunjukkan pada Gambar 5.12.



Gambar 5.12 Komunikasi searah TX menuju RX modul NRF24L01

5.1.5 Perancangan Indikator

Perancangan indikator disini dibentuk oleh lampu LED sebagai penanda navigasi pada Robot beroda. Pada Gambar 5.13 ditunjukkan susunan dan letak dari indikator navigasi. Terletak diperangkat TX atau *gloves* dikarenakan perangkat tersebut yang digunakan sebagai kontrol dan juga digunakan ditubuh pengguna sehingga lebih mudah diketahui.



Gambar 5.13 Indikator pada perangkat TX

Terdapat 4 lampu LED berwarna merah sebagai indikator yang mewakili 4 arah gerakan pada Robot beroda yakni maju,mundur,kiri dan kanan. Nyala LED hanya mewakili satu gerakan dari Robot beroda. Misalkan apabila lampu LED yang berada paling atas menyala maka Robot beroda bergerak maju. Berikut pada Tabel 5.4 menampilkan indikator pada perangkat TX.

No	Nyala LED	Indikator
1	Nomor 1	Robot beroda bergerak maju
2	Nomor 2	Robot beroda bergerak ke kanan
3	Nomor 3	Robot beroda bergerak ke kiri
4	Nomor 4	Robot beroda bergerak mundur

Tabel 5.4 Indikator nyala lampu LED

Namun apabila ingin menggerakkan Robot beroda maju sekaligus belok kanan maka lampu led nomor 1 dan 2 akan menyala secara bersamaan. Begitu juga dengan kondisi yang lain kecuali hanya belok kanan saja maka lampu 2 yang akan menyala.

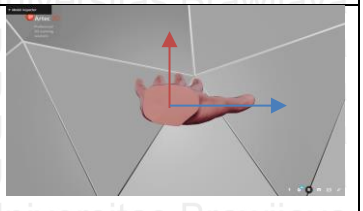
5.1.6 Perancangan Kendali

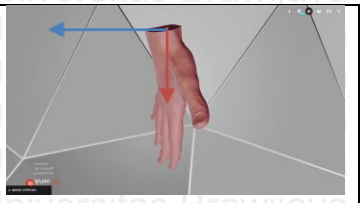
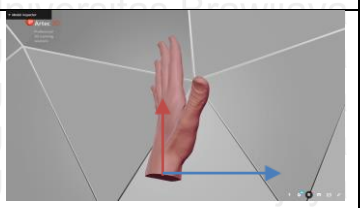
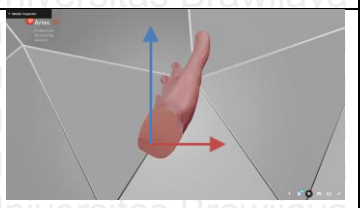
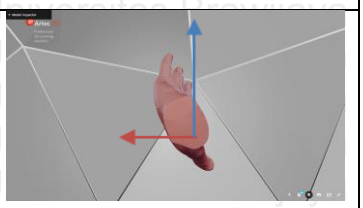
Perlunya perancangan kendali dalam sistem ini bertujuan untuk memperoleh gerakan dan hasil uji yang sesuai dengan yang diinginkan. Perancangan kendali meliputi gerakan gestur tangan oleh pengguna. Jadi ada batasan dan gerakan tertentu yang hanya diperbolehkan dilakukan oleh sistem. Untuk menghindari gerakan bebas atau yang tak beraturan sehingga menyebabkan sistem gagal. Terdapat juga hubungan antara gerak tangan dan gerak motor sehingga dapat memudahkan pengguna untuk memahami gerakan apa saja yang diperbolehkan untuk mengendalikan sistem.

5.1.6.1 Klasifikasi Gestur Tangan

Klasifikasi gerak tangan bertujuan untuk menentukan gerakan atau gestur tangan apa saja yang diperbolehkan dan dibutuhkan oleh sistem. Gestur tangan yang disesuaikan menurut cara kerja sistem terdiri dari lima bagian yaitu gestur flat, gestur maju, gestur mundur, gestur miring ke kiri dan gestur miring ke kanan. Tabel 5.5 dibawah ini menunjukkan klasifikasi gestur tangan berdasarkan arah gerakan.

Tabel 5.5 Klasifikasi gerak tangan

No	Gerakan Tangan	Sudut	Keterangan	Gambar simulasi gerakan
1	Sejajar dengan tanah	0°	Digunakan untuk mengontrol robot beroda agar dalam keadaan idle. Dimana robot tidak bergerak sama sekali serta posisi tangan datar atau sejajar dengan tanah	 Sumbu X (*Pitch*) = biru Sumbu Y (*Roll*) = merah

2	Tangan miring ke bawah	0° - (-90°)	Digunakan untuk mengontrol robot beroda agar bergerak maju sesuai dengan sudut yang telah ditentukan serta posisi tangan menghadap ke bawah. Besar sudut yang dihasilkan berpengaruh terhadap laju kecepatan robot beroda semakin tinggi sudut yang dihasilkan maka semakin cepat pula robot beroda berjalan maju begitu pula sebaliknya.	 Sumbu X (*Pitch*) = biru Sumbu Y (*Roll*) = merah
3	Tangan miring ke atas	0° - 90°	Digunakan untuk mengontrol robot beroda agar bergerak mundur sesuai dengan sudut yang telah ditentukan serta posisi tangan menghadap ke atas. Besar sudut yang dihasilkan berpengaruh terhadap laju kecepatan robot beroda semakin tinggi sudut yang dihasilkan maka semakin cepat pula robot beroda berjalan mundur begitu pula sebaliknya.	 Sumbu X (*Pitch*) = biru Sumbu Y (*Roll*) = merah
4	Tangan miring ke kiri	0° - (-90°)	Digunakan untuk mengontrol robot beroda agar bermanuver ke kiri sesuai dengan sudut yang telah ditentukan serta posisi tangan miring ke kiri. Besar sudut yang dihasilkan berpengaruh terhadap laju kecepatan robot beroda semakin tinggi sudut yang dihasilkan maka semakin cepat pula robot beroda berputar ke kiri begitu pula sebaliknya.	 Sumbu X (*Pitch*) = biru Sumbu Y (*Roll*) = merah
5	Tangan miring ke kanan	0° - 90°	Digunakan untuk mengontrol robot beroda agar bermanuver ke kanan sesuai dengan sudut yang telah ditentukan serta posisi tangan miring ke kanan. Besar sudut yang dihasilkan berpengaruh terhadap laju kecepatan robot beroda semakin tinggi sudut yang dihasilkan maka semakin cepat pula robot beroda berputar ke kanan begitu pula sebaliknya.	 Sumbu X (*Pitch*) = biru Sumbu Y (*Roll*) = merah

Klasifikasi gerak tangan pada tabel diatas dihitung menurut nilai konversi dari data *raw* sensor MPU-6050 yang telah diubah kedalam bentuk sudut maksimal 90° derajat terhadap masing-masing gerakan. Bernilai negatif apabila berlawan dengan gerakan yang sebaliknya. Untuk gerakan vertikal (Sumbu X atau

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

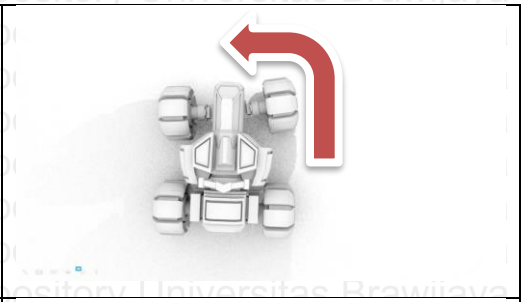
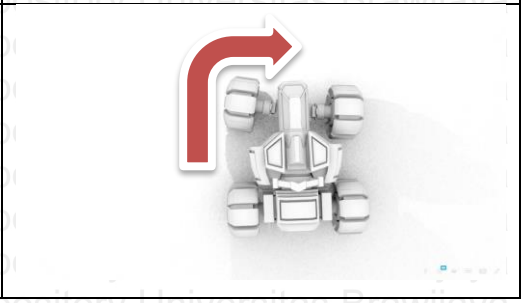
pitch) menginterpretasikan gerakan maju dan mundur. Sedangkan untuk gerakan horisontal (Sumbu Y atau *roll*) meningterpretasikan gerakan kiri dan kanan. Konversi data *raw* menjadi total pada setiap sumbu 180°(setengah lingkaran penuh) sehingga dalam posisi *idle* atau sejajar dengan tanah maka sudutnya adalah 0°. Keterbatasan tekukan telapak tangan manusia sebagai dasar pembentukan sudut total 180° pada sistem.

5.1.6.2 Klasifikasi Gerak Robot

Klasifikasi gerak motor bertujuan untuk mengetahui navigasi dari Robot beroda telah sesuai atau tidak dengan gerakan pada gestur tangan pengguna. Dikarenakan gerakan gestur tangan berupa kecepatan dan sudut juga mempengaruhi laju dari motor maka dilakukan klafisikasi pada Tabel 5.6 dibawah ini.

Tabel 5.6 Klasifikasi gerak Robot

No	Gestur Tangan	Sudut Acuan	Nilai PWM	Nilai Tegangan	Simulasi gerakan robot beroda
1	Sejajar dengan tanah	0°	0	0 V	
2	Tangan miring ke bawah	0° - (-90°)	85-255	1.7 - 5V	
3	Tangan miring ke atas	0° - 90°	85-255	1.7 - 5V	

4	Tangan miring ke kiri	0° - (-90°)	85-255	1.7 - 5V	
5	Tangan miring ke kanan	0° - 90°	85-255	1.7 - 5V	

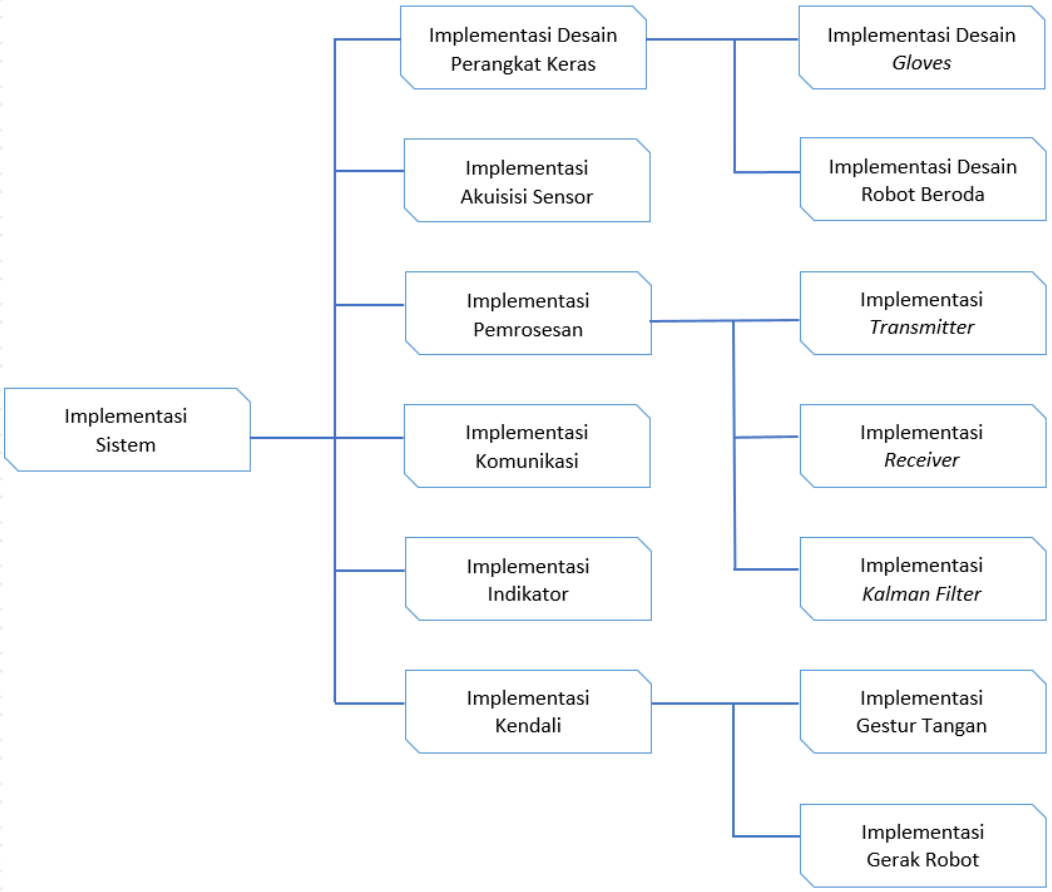
Dalam klasifikasi gerak robot sudut dari sensor (0° - 90°) telah dipetakan dalam bentuk PWM (85 - 255) sehingga apabila sensor mencapai gerak sudut maksimal (90°) maka PWM yang dihasilkan sama dengan 255 (gerakan maksimal motor DC). Karena gerakan maksimal maka besar tegangan yang dibutuhkan 5v dan arus 1A . Batas minimum PWM adalah 85 PWM jadi pada saat pengguna mengendalikan perangkat pengendali setinggi > 0° maka kecepatan awal robot beroda adalah 85 PWM. Hal ini berpengaruh dalam waktu tanggap robot beroda terhadap gerakan gestur tangan yang dihasilkan. Apabila PWM di atur kurang dari 85 maka waktu tanggap robot akan semakin lambat dan mengakibatkan *delay* yang cukup tinggi. Arah gerakan robot beroda dibagi menjadi lima gerakan berdasarkan *input* gerakan gestur tangan.

1. **Sejajar dengan tanah**, Dalam kondisi tersebut robot beroda dalam kondisi diam dikarenakan perubahan sudut yang dihasilkan sebesar 0°. Hal ini berfungsi untuk menghentikan laju robot atau ingin menghentikan proses pengendalian.
2. **Tangan miring ke bawah**, Pada saat kondisi tersebut robot beroda akan bergerak maju sesuai *input* pada gestur tangan.
3. **Tangan miring ke atas**, Pada saat kondisi tersebut robot beroda akan bergerak mundur sesuai *input* pada gestur tangan.
4. **Tangan miring ke kiri**, Pada saat kondisi tersebut robot beroda akan bermanuver ke kiri sesuai *input* pada gestur tangan. Mengakibatkan robot akan berbelok atau dapat berputar.
5. **Tangan miring ke kanan**, Pada saat kondisi tersebut robot beroda akan bermanuver ke kanan sesuai *input* pada gestur tangan. Mengakibatkan robot akan berbelok atau dapat berputar.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

5.2 Implementasi Sistem

Bagian ini menjelaskan bagaimana proses implementasi dari desain dan perancangan sistem terhadap kebutuhan-kebutuhan yang telah dikumpulkan. Implementasi sistem dilakukan dengan berpedoman pada perancangan yang telah dibuat. Tahap ini meliputi langkah-langkah praktis untuk menerapkan dan memenuhi rancangan sistem yang telah dibuat. Tujuannya agar keberadaan sistem dapat memiliki keandalan dalam mencegah kesalahan, kerusakan serta kegagalan proses. Untuk lebih jelasnya Gambar 5.14 berikut merupakan alur implementasi pada sistem.



Gambar 5.14 Alur implementasi pada sistem

Pada diagram alir diatas dapat dilihat bahwa implementasi sistem dibagi menjadi lima bagian yakni implementasi akusisi sensor, implementasi komunikasi, implementasi pemrosesan, implementasi indikator dan implementasi kendali. Implementasi desain perangkat keras meliputi desain *gloves* dan desain robot beroda. Implementasi pemrosesan dibagi menjadi tiga bagian yakni implementasi

transmitter,*receiver* dan *kalman filter* sedangkan implementasi kendali dibagi menjadi dua bagian yakni implementasi gestur tangan dan gerak robot.

5.2.1 Implementasi Desain Perangkat Keras

Dilakukan dengan membangun perangkat *transmitter* (*gloves*) dan *receiver* (robot beroda) berdasarkan komponen yang telah dirancang pada bab perancangan. Desain yang terbentuk harus disesuaikan dengan desain pada perancangan agar hasil serta fungsi pada masing-masing perangkat dapat berjalan sesuai dengan tujuan penelitian.

5.2.1.1 Implementasi Desain *Gloves*

Berdasarkan perancangan desain *gloves* maka desain implementasi *gloves* berupa sarung tangan akan digunakan pada tubuh manusia. Penggabungan *gloves* dengan perangkat pengendali telah diimplementasikan dengan menggunakan media perekat selotip berbahan spons. Ukuran serta posisi antara perangkat pengendali dan *gloves* telah sesuai untuk menunjang kinerja sistem. Seperti yang ditunjukkan pada Gambar 5.15.

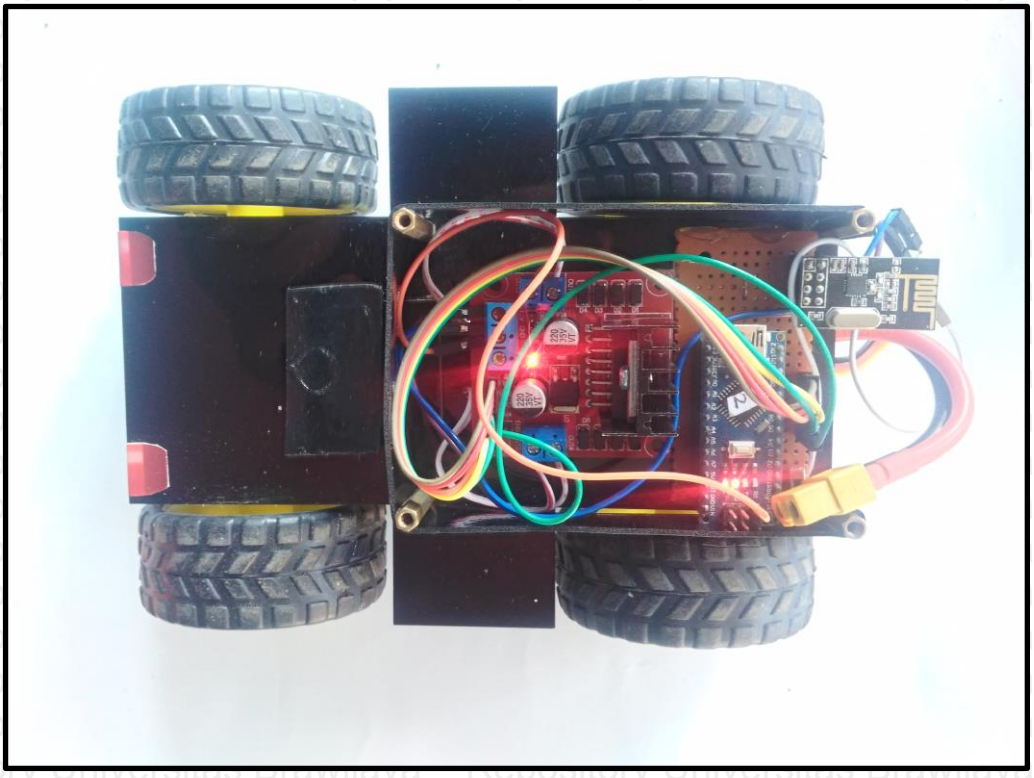
Gambar 5.15 Implementasi desain *Gloves*

Dimensi perangkat pengendali 6cm x 10 cm sedikit berbeda pada tahap peracangan yakni dengan dimensi 4.24 x 9.35cm dikarenakan adanya penampang pelastik sebagai pelindung perangkat kendali. Serta sedikit terdapat perubahan tata letak pada Baterai, lampu LED dan resistor namun, tidak mengurangi fungsi dan kinerja dari komponen tersebut. Untuk desain ukuran *gloves* sama antara perancangan dan imlementasi yakni sebesar 20 x 15cm . Dengan ukuran perangkat pengendali yang tidak cukup besar sehingga cocok diletakkan dibagian atas telapak tangan pengguna. Sehingga memudahkan dalam pengendalian dan gerakan gestur tangan menjadi lebih akurat serta objektif.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

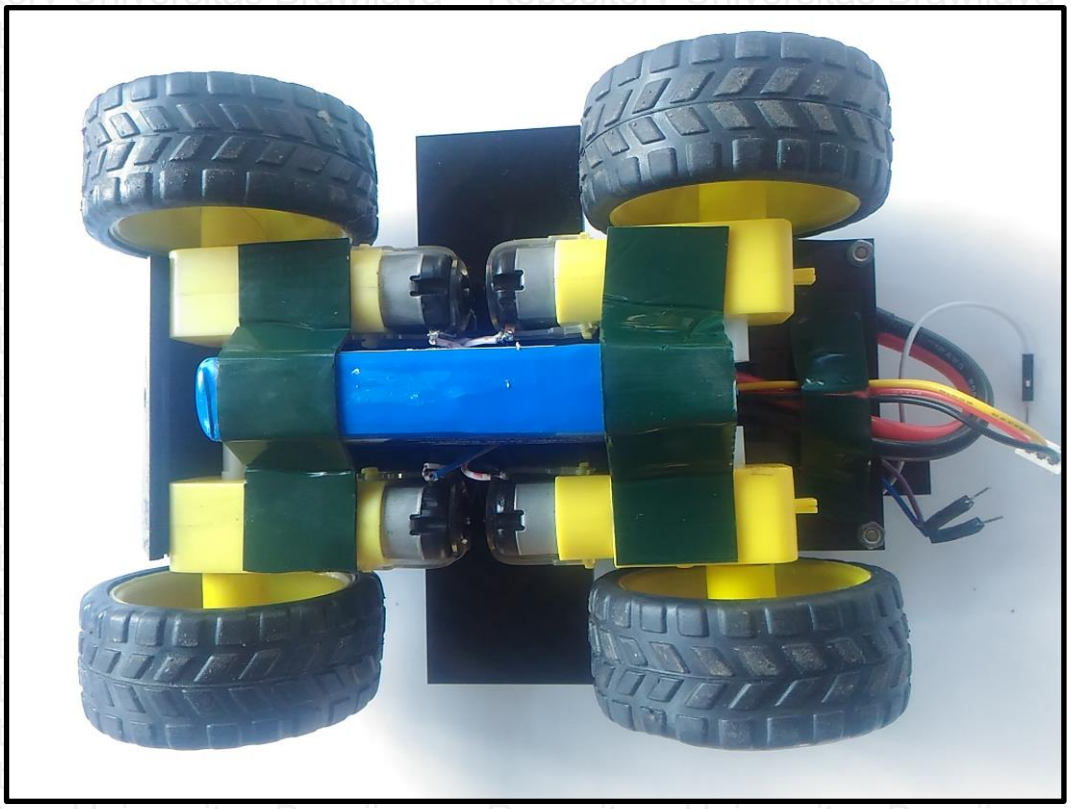
5.2.1.2 Implementasi Desain Robot Beroda

Berdasarkan perancangan desain robot beroda maka desain implementasi robot berupa kendaraan dengan empat roda penggerak. Dimensi yang digunakan pada implementasi desain robot adalah 170mm x 130mm dan dapat menampung seluruh komponen elektronik pada sistem sesuai dengan perancangan. Kemudian berikut adalah gambaran desain penampang setelah diletakkan komponen elektronika.



Gambar 5.16 Implementasi Robot beroda (tampak atas)

Gambar 5.16 menunjukkan penampakkan implementasi desain Robot beroda dari atas. Pada implementasi desain tersebut tata letak komponen elektronika berupa Arduino Nano V3, Modul NRF24L01 receiver, saklar, roda dan driver motor L298N tidak mengalami perubahan namun impelementasi pengkabelan masih tidak rapi. Karena ketidakrapian dalam pengkabelan maka penulis memberikan tutup pada bagian sistem sehingga komponen-komponen tersebut tidak dapat terlihat. Komponen elektronika terletak disebelah kanan karena merupakan bagian belakang pada Robot beroda yang mana bagian kiri merupakan bagian depan. Implementasi dimensi penampang dan dimensi komponen yang telah diimplementasikan sesuai, sehingga ukuran Robot beroda tersebut pas dan sesuai dengan kebutuhan sistem. Untuk letak modul penggerak seperti motor DC serta daya untuk sistem terletak dibagian bawah penampang, berikut adalah gambar implementasi desainnya.



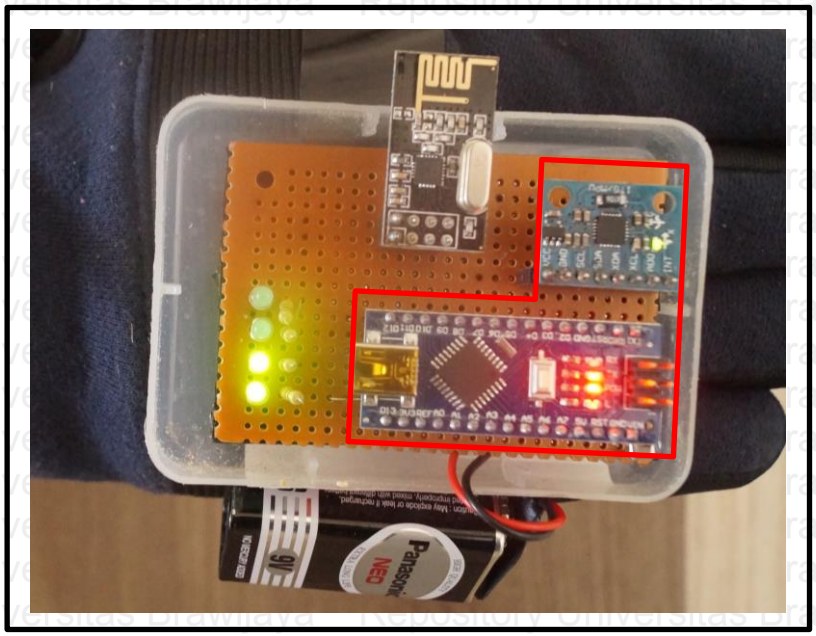
Gambar 5.17 Implementasi desain robot beroda (tampak bawah)

Gambar 5.17 menunjukkan implementasi desain tampak bawah dari robot beroda memperlihatkan komponen penggerak sistem seperti motor DC dan keempat roda serta sumber daya yakni baterai Li-Po (Lithium Polymer) 2200 mAh. Terletak dibawah dan tersembunyi dikarenakan roda motor dan dinamo sebagai penahan beban keseluruhan komponen yang ada diatasnya serta tersembunyi untuk meningkatkan nilai estetika dari Robot beroda dan juga untuk mengoptimalkan segala sisi dan ruang yang terbatas.

Pada implementasi desain sesuai dengan perancangan. Tidak ada perubahan yang signifikan kecuali penambahan media perekat selotip untuk merekatkan komponen-komponen yang terdapat pada robot beroda. Sistem pengkabelan yang terbilang cukup rumit dikarenakan perangkat yang berada dibawah penampang harus terhubung dengan sistem yang berda di atas penampang.

5.2.2 Implementasi Akuisisi Sensor

Implementasi akuisisi sensor dilakukan untuk mengimplementasikan sensor MPU-6050 sebagai komponen sensing gerak gestur tangan pada sistem. Konfigurasi antar komponen yang digunakan dalam implementasi juga disesuaikan dengan perancangan. Pada implementasi akusisi sensor dilakukan percobaan komunikasi antara arduino nano V3 dengan sensor MPU-6050 seperti pada Gambar 5.18 dibawah ini.



Gambar 5.18 Implementasi sensor MPU-6050 dan Arduino uno V3

Implementasi akuisisi sensor juga meliputi implementasi kode program pada arduino nano untuk mengakuisisi nilai sensor yang diinginkan pada sensor MPU-6050. Pada dasarnya komunikasi antara sensor dengan mikrokontroler menggunakan protokol *I2c (Inter Integrated Circuit)* merupakan protokol komunikasi serial yang digunakan untuk komunikasi antara mikrokontroler dan perangkat periferal seperti memori, sensor temperatur dan *I/O expander*.Berikut merupakan kode program yang digunakan untuk mengakuisisi nilai sensor.

Tabel 5.7 Kode program konfigurasi komunikasi dengan protokol I2c

Kode program : konfigurasi komunikasi dengan protokol I2c

```
1  const uint8_t IMUAddress = 0x68;
2  const uint16_t I2C_TIMEOUT = 1000;
3
4  uint8_t i2cWrite(uint8_t registerAddress, uint8_t data, bool
5  sendStop) {
6    return i2cWrite(registerAddress, &data, 1, sendStop);
7  }
8
9  uint8_t i2cWrite(uint8_t registerAddress, uint8_t *data,
10 uint8_t length, bool sendStop) {
11   Wire.beginTransmission(IMUAddress);
12   Wire.write(registerAddress);
13   Wire.write(data, length);
14   uint8_t rcode = Wire.endTransmission(sendStop);
15   if (rcode) {
16     Serial.print(F("i2cWrite failed: "));
17     Serial.println(rcode);
```

```
18    }
19    return rcode;
20  }
21
22  uint8_t i2cRead(uint8_t registerAddress, uint8_t *data,
24  uint8_t nbytes) {
25    uint32_t timeOutTimer;
26    Wire.beginTransmission(IMUAddress);
27    Wire.write(registerAddress);
28    uint8_t rcode = Wire.endTransmission(false); // Don't
29  release the bus
30    if (rcode) {
31      Serial.print(F("i2cRead failed: "));
32      Serial.println(rcode);
33      return rcode;
34    }
35    Wire.requestFrom(IMUAddress, nbytes, (uint8_t)true); //
36  Send a repeated start and then release the bus after reading
37    for (uint8_t i = 0; i < nbytes; i++) {
38      if (Wire.available())
39        data[i] = Wire.read();
40      else {
41        timeOutTimer = micros();
42        while (((micros() - timeOutTimer) < I2C_TIMEOUT) &&
43  !Wire.available());
44        if (Wire.available())
45          data[i] = Wire.read();
46        else {
47          Serial.println(F("i2cRead timeout"));
48          return 5; // This error value is not already taken
49  by endTransmission
50        }
51      }
52    }
53    return 0; // Success
54  }
```

Dalam kode program pada Tabel 5.7 diatas merupakan konfigurasi protokol I2c untuk komunikasi antara sensor denan mikrokontroler arduino nano V3. Arduino nano V3 bertindak sebagai *master* sedangkan sensor IMU MPU-6050 menjadi *slave*. Pertama-tama dengan menginisialisasi IMU *address* = 0x68 yakni alamat yang digunakan untuk pengiriman data dari sensor IMU. Menggunakan `uint8_t` yang artinya *unsigned* int pada bagian kode kita akan melihat bahwa variabel *number* 8 memiliki hasil 255. Karena 255 adalah nilai maksimum *integer unsigned* atau `uint8_t`. Jadi jika kita memberi nilai 256, hasilnya adalah 0. Memang, setelah 255 kita kembali ke 0 begitu juga terus berulang kelipatannya. Jadi kita tidak akan dapat memiliki nilai 256 dalam *char* (atau *byte*). Jika kita ingin memiliki nilai seperti itu, kita harus menggunakan tipe lain, misalnya *unsigned int* uint16_t sama dengan 2 *byte* atau 16 bit.

Pada dasarnya kesimpulan dari kode program diatas adalah mengatur pengiriman nilai dari alamat pada sensor menuju mikrokontroler. Menginisialisasi *register adress* dan panjang data yang akan ditampung atau diakuisisi. Kondisi logis

apabila terjadi request terhadap *slave* maka data akan dikirimkan melalui jaringan yang telah tersedia dalam waktu tertentu. Apabila tidak tersedia jaringan, maka akan mengirim pesan *error* kepada *master* (Arduino nano V3).

Kemudian setelah melakukan konfigurasi i2c langkah selanjutnya adalah memanggil data sensor ke kode program utama. Pada kode program utama perangkat ini berisi proses-proses yang digunakan mulai dari akusisi nilai sensor, konversi nilai menjadi sudut dan pwm, proses *kalman filter* hingga nilai tersebut dikirim ke perangkat *receiver* pada Robot beroda. Namun, dalam hal ini yang akan dibahas adalah bagaimana mekanisme akuisisi nilai sensor oleh perangkat arduino.

Tabel 5.8 Kode program akuisisi nilai sensor

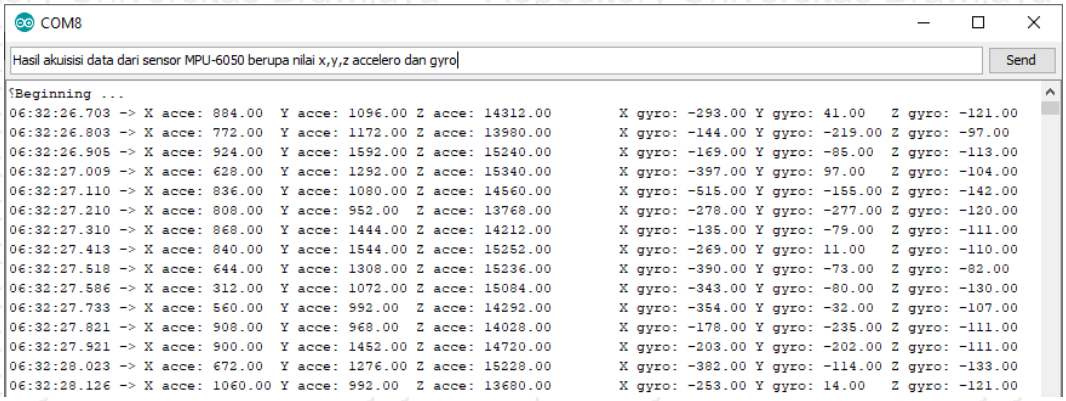
Kode program : Kode program utama (Perangkat TX)

```
1  uint32_t timer;
2  uint8_t i2cData[14];
3
4  void setup(){
5    Serial.begin(9600);
6
7  Wire.begin();
8    #if ARDUINO >= 157
9      Wire.setClock(400000UL); // Set I2C frequency to 400kHz
10   #else
11     TWBR = ((F_CPU / 400000UL) - 16) / 2; // Set I2C
12 frequency to 400kHz
13   #endif
14
15   i2cData[0] = 7; // Set the sampel rate to 1000Hz -
16 8kHz/(7+1) = 1000Hz
17   i2cData[1] = 0x00; // Disable FSYNC and set 260 Hz Acc
18 filtering, 256 Hz Gyro filtering, 8 KHz sampling
19   i2cData[2] = 0x00; // Set Gyro Full Scale Range to
20 ±250deg/s
21   i2cData[3] = 0x00; // Set Accelerometer Full Scale Range
22 to ±2g
24   while (i2cWrite(0x19, i2cData, 4, false)); // Write to all
25 four registers at once
26   while (i2cWrite(0x6B, 0x01, true)); // PLL with X axis
27 gyroscope reference and disable sleep mode
28
29   while (i2cRead(0x75, i2cData, 1));
30   if (i2cData[0] != 0x68) { // Read "WHO_AM_I" register
31     Serial.print(F("Error reading sensor"));
32     while (1);
33   }
34
35   delay(100); // Wait for sensor to stabilize
36
37   /* Set kalman and gyro starting angle */
38   while (i2cRead(0x3B, i2cData, 6));
39   accX = (int16_t)((i2cData[0] << 8) | i2cData[1]);
40   accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
41   accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);
42 }
43
```

```
44  void loop(){
45  while (i2cRead(0x3B, i2cData, 14));
46    accX = (int16_t)((i2cData[0] << 8) | i2cData[1]); // 8 bit
47  merupakan banyaknya data yang dikirim per detak, dikarenakan
48  arduino hanya mampum menampung data 8 bit
49    accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
50    accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);
51    tempRaw = (int16_t)((i2cData[6] << 8) | i2cData[7]);
52    gyroX = (int16_t)((i2cData[8] << 8) | i2cData[9]);
53    gyroY = (int16_t)((i2cData[10] << 8) | i2cData[11]);
54    gyroZ = (int16_t)((i2cData[12] << 8) | i2cData[13]);;
55
56    double dt = (double)(micros() - timer) / 1000000; //
57  Calculate delta time
58    timer = micros();
59   }
60  }
```

Pada Tabel 5.8, merupakan potongan kode program pada program utama untuk akusisi nilai sensor. Pertama-tama dengan melakukan deklarasi terhadap *timer* dan alamat I2c yang telah dibuat pada kode proram I2c sebelumnya. Kemudian pada baris berikutnya adalah mengatur frekuensi I2c menajadi 400 Khz. Pada baris ke-15 mengatur tingkat sampel dari 1000Hz sampai 8KHz pada data ke 0 pada *array*. Selanjutnya pada baris ke-16 menon-aktifkan *FSYNC* dan mengaur 260Hz sebagai frekuensi untuk *accelerometer filtering*, 256 untuk *gyroscope filtering* dan 8Khz *sampling*. Pada baris ke-17, pada data ke 3 *array* mengatur *gyroscope* rentang skala penuh hingga ± 250 derajat/detik. Pada baris-18 mengatur *accelerometer* pada rentang skala penuh hingga ±2g. Kemudian perhatikan baris 37-41 merupaka proses akuisisi data untuk mengatur *kalman filter* dan sudut awal pada *gyro*. Selanjutnya adalah akuisisi nilai *gyro* dan *acc* dari sensor MPU-6050 pada baris 45-54. Pada bagian ini I2c membaca data pada sensor dengan *register map* 0x3B yang merupakan alamat *register* untuk mengambil nilai x_*out* pada *accelerometer* begitu seterusnya dengan mengambil data dengan panjang 16 bit atau sama dengan 2 *byte* data *accelerometer* dan *gyro* serta *temp* diakuisisi data oleh arduino. Namun arduino hanya mampu menampung data sebanyak 8 bit per detak / *Clock* jadi data yang diterima 8 bit per detak.

Dengan mengimplementasikan kedua kode program diatas maka pada Gambar 5.19 berikut merupakan gambar akusisi nilai sensor yang telah di ambil dari MPU-6050 dan direpresentasikan dalam bentuk nilai *gyro* dalam sudut x,y dan z serta nilai *acce*lero dalam sudut x,y dan z. Namun, namun nilai yang digunakan oleh sistem merupakan nilai x dan y (*pitch* dan *roll*) dari masing-masing *accelero* dan *gyro.*



Gambar 5.19 Hasil akuisisi data sensor MPU-6050

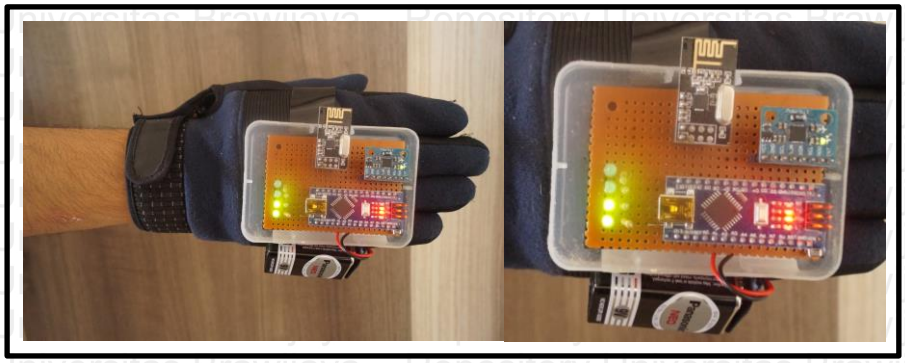
5.2.3 Implementasi Pemrosesan

Implementasi pemrosesan berarti mererapkan seluruh proses yang terjadi pada sistem menurut perancangan yang telah disusun. Imlementasi dibagi menjadi tiga bagian mulai dari implmentasi *transmitter*, *receiver* sampai *kalman filter*.

5.2.3.1 Implementasi *Transmitter*

Implementasi *transmitter* secara garis besar merupakan implementasi perangkat TX sebagai fungsi pengirim data atau *slave* terhadap sistem. Dalam hal ini implementasi juga merupakan penerapan seluruh sub sistem yang terdapat pada perangkat *transmitter*. Seperti akuisisi nilai pada sensor, konversi nilai, proses filter sinyal dan pengiriman nilai serta pengaturan indikator LED.

Pada bab perancangan sebelumnya telah dirancang komponen apa saja yang akan dibentuk dan diimplementasikan serta bagaimana desain keseluruhan perangkat *transmitter*. Gambar 5.17 berikut merupakan gambar perangkat *transmitter* yang telah menerapkan keseluruhan rancangan.



Gambar 5.17 Implementasi perangkat TX

Pada perangkat *transmitter* juga mengimplementasikan kode program untuk memproses sinyal yang telah diterima sebelumnya pada akuisisi sensor. Berikut merupakan potongan kode program pada perangkat TX mengenai proses—proses yang ada didalamnya.

Tabel 5.9 Kode program proses konversi nilai

Kode program : Kode Perangkat TX

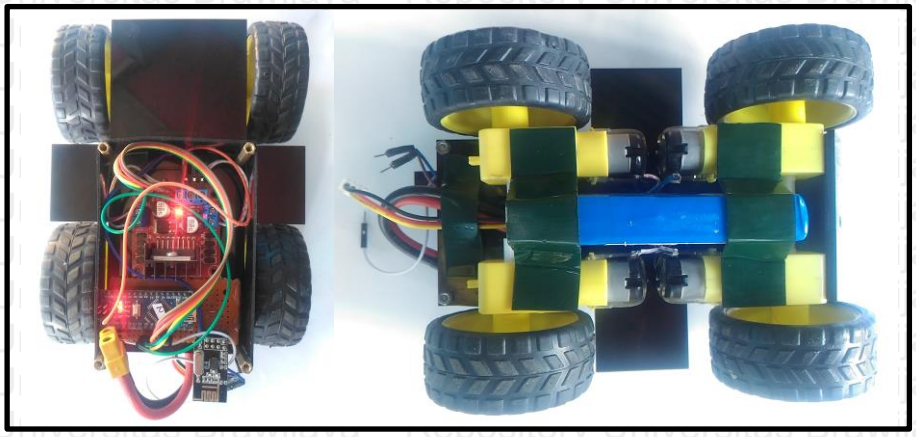
```
1  void setup(){
2    double roll  = atan2(accY, accZ) * RAD_TO_DEG;
3    double pitch = atan(-accX / sqrt(accY * accY + accZ *
4  accZ)) * RAD_TO_DEG;
5    gyroXangle = roll;
6    gyroYangle = pitch;
7    compAngleX = roll;
8    compAngleY = pitch;
9  }
10
11 void loop(){
12   double roll  = atan2(accY, accZ) * RAD_TO_DEG;
13   double pitch = atan(-accX / sqrt(accY * accY + accZ *
14 accZ)) * RAD_TO_DEG;
15
16   double gyroXrate = gyroX / 131.0; // Convert to deg/s
17   double gyroYrate = gyroY / 131.0; // Convert to deg/s
18
19   if (gyroXangle < -180 || gyroXangle > 180)
20     gyroXangle = kalAngleX;
21   if (gyroYangle < -180 || gyroYangle > 180)
22     gyroYangle = kalAngleY;
24
25   int defaultPwm = 85;
26   int pwmRef;
27
28   int pwmKiri = 0;
29   int pwmKanan = 0;
30
31   if(kalAngleX >= 0)
32   {
33     pwmRef = int(kalAngleX) + defaultPwm;
34     pwmKiri = pwmRef - int(kalAngleY);
35     pwmKanan = pwmRef + int(kalAngleY);
36
37     if(pwmKiri < 0 ){ pwmKiri = 0; }
38     if(pwmKiri > 255) { pwmKiri = 255; }
39
40     if(pwmKanan < 0 ){ pwmKanan = 0; }
41     if(pwmKanan > 255) { pwmKanan = 255; }
42   }
43
44   if(kalAngleX < 0)
45   {
46     pwmRef = int(kalAngleX) - defaultPwm;
47     pwmKiri = pwmRef + int(kalAngleY);
48     pwmKanan = pwmRef - int(kalAngleY);
49
50     if(pwmKiri > 0 ){ pwmKiri = 0; }
51     if(pwmKiri < -255) { pwmKiri = -255; }
52
53     if(pwmKanan > 0 ){ pwmKanan = 0; }
54     if(pwmKanan < -255) { pwmKanan = -255; }
55   }
56 }
```

Pada Tabel 5.9 diatas merupakan potongan kode program pada perangkat TX yang mengolah sinyal dan mengkonversi sinyal data kedalam bentuk pwm. Tentu terdapat banyak proses sebelum nilai *raw* dari sinyal diubah kedalam bentuk pwm. Langkah awal adalah dengan mengubah (maping) nilai *raw* dari sensor MPU-6050 ke dalam bentuk sudut 180° untuk *pitch* dan *roll* seperti pada baris 12-17. Sedangkan pada baris berikutnya kondisi apabila *gyro*Xangle atau sudut x *gyro* kurang dari -180 dan kurang dari 180 maka sama dengan kalangleX. Pada baris berikutnya juga terdapat kondisi apabila *gyro*Yangel atau sudut y *gyro* kurang dari -180 dan kurang dari 180 maka sama dengan kalangleY.

Pengaturan *default* pwm pada kode program diatas adalah 85 agar responsif pada saat pwm naik akibat gerakan gestur tangan pengguna. Jikat pwm di *set* 0 maka akan terjadi *delay* yang cukup besar antara gestur dan Robot beroda. Oleh karena itu pada baris 31-56 merupakan kondisi yang terjadi apabila variabel kalangleX kurang dari sama dengan nol atau lebih dari nol. Terdapat juga konversi dari sudut menjadi nilai pwm dengan maping pwm *default*. Jadi sudut 0-90 derajat sama dengan nilai pwm dari 85-255.

5.2.3.2 Implementasi *Receiver*

Gambar 5.20 berikut merupakan implementasi keseluruhan komponen yang disusun pada perancangan *receiver* atau robot beroda.



Gambar 5.20 Implementasi perangkat RX

Implementasi *receiver* dalam hal ini adalah penerapan dari keseluruhan rancangan sistem untuk mengolah data yang diterima dari perangkat *transmitter*. Secara garis besar proses yang berada pada perangkat ini adalah mengolah data yang diterima berupa nilai pwm. Nilai pwm tersebut berfungsi sebagai *input* pada sistem untuk mengendalikan *motor driver* dan motor DC agar dapat bergerak. Pergerakan tersebut diatur berdasarkan gerakan gestur (*pitch* dan *roll*) tangan pengguna yang telah di konversi menjadi nilai pwm. Berikut pada Tabel 5.10 dibawah ini merupakan implementasi nilai pwm pada kode program perangkat *receiver*.

Tabel 5.10 Kode program Implementasi nilai pwm

Kode program : Implementasi Kode Perangkat RX

```
1  #define pwmKanan  9
2  #define in1       8
3  #define in2       7
4  #define in3       3
5  #define in4       2
6  #define pwmKiri   6
7
8  char data[32];
9  String valSplit[5];
10
11 void setup(){
12   Serial.begin(9600);
13
14   pinMode(in1, OUTPUT);
15   pinMode(in2, OUTPUT);
16   pinMode(in3, OUTPUT);
17   pinMode(in4, OUTPUT);
18 }
19
20 void loop(){
21
22   if(!Mirf.isSending() && Mirf.dataReady())
24   {
25
26     Mirf.getData((byte *)&data);
27     parseData(String(data), ",");
28
29     if(valSplit[0].toInt() > 0 && valSplit[1].toInt() > 0)
30     {
31       digitalWrite(in1, LOW);
32       digitalWrite(in2, HIGH);
33       digitalWrite(in3, LOW);
34       digitalWrite(in4, HIGH);
35       analogWrite(pwmKiri, valSplit[0].toInt());
36       analogWrite(pwmKanan, valSplit[1].toInt());
37
38     }
39     if(valSplit[0].toInt() < 0 && valSplit[1].toInt() < 0)
40     {
41       digitalWrite(in1, HIGH);
42       digitalWrite(in2, LOW);
43       digitalWrite(in3, HIGH);
44       digitalWrite(in4, LOW);
45       analogWrite(pwmKiri, abs(valSplit[0].toInt()));
46       analogWrite(pwmKanan, abs(valSplit[1].toInt()));
47     }
48     if(valSplit[0].toInt() == 0 && valSplit[1].toInt() == 0)
49     {
50       digitalWrite(in1, HIGH);
51       digitalWrite(in2, LOW);
52       digitalWrite(in3, HIGH);
53       digitalWrite(in4, LOW);
54       analogWrite(pwmKiri, 0);
55       analogWrite(pwmKanan, 0);
56     }
57
```

```
58      Serial.print(valSplit[0].toInt()); Serial.print("\t");
59      Serial.print(valSplit[1].toInt()); Serial.print("\t");
60      Serial.println();
61      //Serial.println(data);
62    }
63  }
64
65  void parseData(String text, String key)
66  {
67    int countSplitSecond=0;
68    int lastIndexSecond=0;
69    text += key;
70
71    for(int j = 0; j < text.length(); j++)
72    {
72      if(text.substring(j, j+1) == key)
73      {
74        valSplit[countSplitSecond] =
75  text.substring(lastIndexSecond,j);
76        lastIndexSecond = j + 1;
77        countSplitSecond++;
78      }
79    }
80  }
```

Pada Tabel 5.10 diatas merupakan proses implementasi nilai pwm untuk mengatur gerakan pada Robot beroda. Langkah awal adalah mendefinisikan pin pada *driver motor* dengan pin pada arduino nano v3. Kemudian pada fungsi *setup* mengatur pin mode pada *input* satu sampai empat sebagai *output*. Fungsi perulangan berisi penerimaan data dari *transmitter* kemudian data diolah dalam kondisi-kondisi tertentu sesuai dengan *input* data pwm untuk memberi kondisi "*HIGH*" dan "*LOW*" pada keempat motor DC. Tentunya pemberian kondisi berdasarkan nilai pwm kiri dan kanan yang diberikan dari *input* data yang masuk pada *receiver*. Maksud dari pwmKanan adalah nilai yang diberikan pada motor 1 dan 2 sedangkan pwmKiri adalah nilai yang diberikan pada motor 3 dan 4. Dalam penelitian ini menggunakan empat motor DC sebagai tenaga penggerak. Dalam keadaan belok ke kanan maka roda sebelah kiri (1 dan 2) yang bergerak begitu juga sebaliknya. Apabila kondisi maju atau mundur maka keempat roda akan bergerak secara bersamaan.

5.2.3.3 Implementasi *Kalman filter*

Implementasi *kalman filter* merupakan penerapan dari proses filter pada sistem. Filter sendiri terjadi pada perangkat *transmitter*. Proses filter merupakan langkah awal sebelum sinyal diolah menjadi sudut dan pwm. *Kalman filter* bertujuan untuk meningkatkan akurasi sinyal dan menghilangkan gangguan pada sinyal. Pada tahap ini proses filter menggunakan *library kalman.h* serta mengkonver persamaan pada tahap perancangan menjadi bahasa pemograman "C" pada arduino IDE. Berikut implementasi kode program pada *library kalman.h* sesuai dengan persamaan pada perancangan *kalman filter*.

Tabel 5.11 *Step 1 Time update*

Time update (Prediction)	`rate = newRate - bias;` `angle += dt * rate;`

Kode program *time update* untuk *kalman filter* dapat diartikan sebagai proses prediksi. Dalam dua persamaan pertama akan mencoba untuk memprediksi keadaan saat ini dan *matrix covariance error* pada perubahan waktu `dt`. Pertama, filter akan mencoba memperkirakan kondisi saat ini berdasarkan semua status sebelumnya dan pengukuran *gyro*.

Keterangan :

`angle` = $\hat{x}_{k|k-1}$ estimasi pengukuran

`angle + dt` = $F\hat{x}_{k-1|k-1}$ estimasi waktu (prediksi)

`rate` = $B\dot{\theta}_k$ pengukuran pada *gyro*

Tabel 5.12 *Step 2 Update estimation error covariance*

Update estimation error covariance (Prediction)	`P[0][0] += dt * (dt*P[1][1] - P[0][1] - P[1][0] + Q_angle);`
	`P[0][1] -= dt * P[1][1];`
	`P[1][0] -= dt * P[1][1];`
	`P[1][1] += Q_bias * dt;`

Pada kode program Tabel 5.4 menunjukkan proses *Update estimation error covariance.* Memproyeksikan *error covariance* sebelum ke tahap koreksi. Prinsip kode program diatas adalah *error covariance* akan meningkat sejak terakhir kali memperbarui estimasi keadaan, oleh karena itu perlu mengalikan *matrix error covariance* dengan model transisi keadaan `dt` dan *transpose* dari `(dt*P[1][1] - P[0][1] - P[1][0])` dan menjumlahkan dengan *noise proses* `Q_angle`.

Keterangan :

`P[0][0]` = $P_{k|k-1}$ merupakan *matrix error covariance* 2x2 untuk *Update estimation error*

`dt * (dt*P[1][1] - P[0][1] - P[1][0])- P[1][0]` = $FP_{k-1|k-1}F^T$ merupakan proses perhitungan estimasi keadaan dengan mengalikan estimasi waktu (`dt`) dengan matrix *transpose* dari *matrix error covariance* `P[1][1]`, `P[0][1]`, `P[1][0]` dan `P[1][0]`.

`Q_angle` = Q_k *noise proses* `Q_angle` yang dihasilkan *accelerometer*

Tabel 5.13 *Step 3 Calculation angle and bias*

Calculation angle and bias (Correction)	`float y = newAngle - angle;`

Step 3 pada Tabel 5.13 merupakan kode program proses kalkulasi untuk mencari perbedaan sudut dengan mendefinisikan variabel `y`. Jadi sudut baru

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

(`newAngle`) hasil pengukuran dikurangi dengan sudut hasil pengukuran dari *accelerometer* yang sebelumnya untuk memperoleh perbedaan sudut atau sudut yang menjadi hasil perhitungan kalman.

Keterangan :

`y` = $\tilde{y}_k$ variabel untuk menentukan sudut dari perhitungan kalman

`newAngle` = Z_k sudut baru

$H\hat{x}_{k|k-1}$ = merupakan ruang observasi untuk memetakan keadaan *apriori* $\hat{x}_{k|k-1}$ ke dalam ruang pengamatan yang merupakan pengukuran dari *accelerometer*

Tabel 5.14 *Step 4 Measurement update estimation error*

Measurement update estimation error (Correction)	`float S = P[0][0] + R_measure`

Tabel 5.14 merupakan proses koreksi dengan *Discrete Kalman filter measurement update* atau memperbarui pengukuran diskrit *kalman filter*. Selanjutnya dengan mencari *estimation error* S_k dengan menjumlahkan prediksi dari matrix $P_{00k|k-1}$ dengan *covariance noise measurenment*.

Keterangan :

`float S` = S_k definisi variabel S untuk proses perhitungan inovasi covariance

`P[0][0]` = $HP_{k|k-1}H^T$ *matrix error covariance* dengan model pengamatan variabel H

`R_measure` = R *covariance noise measurenment* yang didapatkan dari kesalahan estimasi pengukuran

Tabel 5.15 *Step 5 Calculate Kalman gain*

Calculate Kalman gain (Correction)	`K[0] = P[0][0] / S;` `K[1] = P[1][0] / S;`

Sehingga pada Tabel 5.15 diperoleh kode program untuk kalkulasi *kalman gain* dengan P *matrix* 2x1. Namun, perhatikan bahwa S dapat berupa *matrix* dan tidak dapat membagi P dengan S sebagai gantinya harus diubah kedalam bentuk *inverse matrix*.

Keterangan :

`K[0] dan K[1]` = K_k merupakan *matrix* untuk kalkulasi *kalman gain*

`P[0][0] / S dan P[1][0] / S` = $\frac{\begin{bmatrix}P_{01}\\P_{10}\end{bmatrix}_{k|k-1}}{S_k}$ merupakan *invers matrix*

Tabel 5.16 *Step 6 Update Posteriori Estimate*

Calculate angle and bias *From kalman gain and Angle difference (Update Posteriori Estimate)*	`angle += K[0] * y;` `bias += K[1] * y;`

Kemudian dalam tahap ke-enam, Pada Tabel 5.16 menunjukkan kalkulasi pada sudut dan *bias* dengan mengalikan *kalman gain* terhadap variabel $\tilde{y}$ (perbedaan sudut) pada *step-3.* Dalam tahap ini diperoleh *angle* dari pembacaan *accelero* dan *bias* dari pembacaan gyro yang telah diperbarui dan telah melalui proses filter.

Tabel 5.17 *Step 7 Update posteriori error covariance matrix*

Update posteriori error covariance matrix	`float P00_temp = P[0][0];`
	`float P01_temp = P[0][1];`
	
	`P[0][0] -= K[0] * P00_temp;`
	`P[0][1] -= K[0] * P01_temp;`
	`P[1][0] -= K[1] * P00_temp;`
	`P[1][1] -= K[1] * P01_temp;`

Sehingga pada Tabel 5.17 diperoleh kode program untuk kalkulasi *estimation error covariance* untuk proses *Update the error covariance* . Apa yang dilakukan filter adalah pada dasarnya mengoreksi sendiri *matrix error covariance* berdasarkan seberapa banyak kita mengoreksi estimasi. Ini masuk akal ketika kita memperbaiki keadaan berdasarkan *apriori matrix error covariance* P, tetapi juga inovasi covariance S.

Tahap implementasi *kalman filter* telah menentukan variabel *Q_angle*, *Q_bias* dan *R_measure* dengan metode *try and error*. Diperoleh Nilai *Q_angle* = 0.001, *Q_bias* = 0.003 dan Nilai *R_measure* = 0.03. Penentuan variabel tersebut akan dibahas lebih lanjut pada bab pengujian.

Selain implementasi *library* kalman.h sistem juga mengimplementasikan kode program untuk proses konversi nilai yang dibutuhkan dalam proses *kalman filter* kedalam bentuk derajat. Serta *complmentary filter* untuk proses *sensor fusion* untuk menunjang kinerja *kalman filter*.

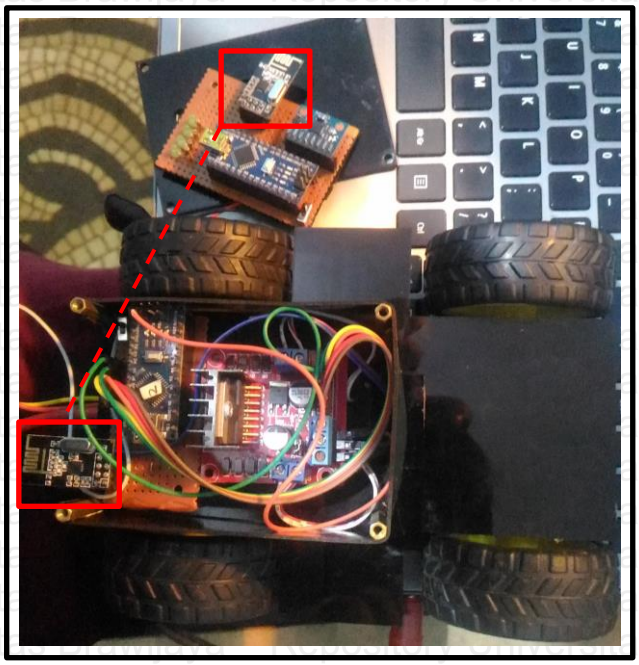
Tabel 5.18 Kode program proses *kalman filter*

Kode program : Kode Perangkat TX	
1	#include <Kalman.h>
2	
3	double *roll* = atan2(accY, accZ) * RAD_TO_DEG;
4	double *pitch* = atan(-accX / sqrt(accY * accY + accZ *
5	accZ)) * RAD_TO_DEG;
6	
7	double *gyroXrate* = *gyroX* / 131.0; // Convert to deg/s
8	double *gyroYrate* = *gyroY* / 131.0; // Convert to deg/s
9	
10	if ((*roll* < -90 && kalAngleX > 90) || (*roll* > 90 &&
11	kalAngleX < -90)) {
12	kalmanX.setAngle(*roll*);
13	compAngleX = *roll*;
14	kalAngleX = *roll*;
15	*gyroXangle* = *roll*;
16	} else
17	kalAngleX = kalmanX.getAngle(*roll*, *gyroXrate*, dt); //
18	Calculate the angle using a *Kalman filter*
19	
20	if (abs(kalAngleX) > 90)
21	*gyroYrate* = -*gyroYrate*; // Invert rate, so it fits the
22	restriced *accelerometer* reading
24	kalAngleY = kalmanY.getAngle(*pitch*, *gyroYrate*, dt);
25	
26	*gyroXangle* += *gyroXrate* * dt; // Calculate *gyro* angle
27	without any filter
28	*gyroYangle* += *gyroYrate* * dt;
29	K = 0.93;
30	K1 = 1 - K;
31	compAngleX = K * (compAngleX + *gyroXrate* * dt) + K1 *
32	*roll*; // Calculate the angle using a Complimentary filter
33	compAngleY = K * (compAngleY + *gyroYrate* * dt) + K1 *
34	*pitch*;
35	

Pada Tabel 5.18 merupakan potongan kode program pada perangkat *transmitter* untuk proses filter. Pada baris ke-1 merupakan *library* yang digunakan untuk proses filter oleh *kalman filter*. Kemudian pada baris 3-8 merupakan kode program untuk konversi nilai *pitch* dan *roll* yang di dapat pada *accelerometer* dan *gyroscope* menjadi nilai sudut 90° sampai -90° (total 180°). Pada baris ke-10 sampai dengan 28 merupakan kondisi tertentu apabila sudut mencapai batas 90° maka sistem akan melakukan perhitungan sudut menggunakan *kalman filter* untuk proses filter. Kemudian konstanta yang digunakan adalah 0.93 dan konstanta untuk *compementary filter* (K1) adalah 0.07 sesuai dengan sisa 1 – K. *Output* dari filter tersebut berupa nilai *pitch* dan *roll* yang telah minim terhadap *noise*.

5.2.4 Implementasi Komunikasi

Implementasi komunikasi merupakan penerapan sistem komunikasi pada perangkat *Transmitter* (TX) dan *Receiver* (RX) pada sistem. Berkaitan dengan bagaimana menghubungkan antara perangkat TX dan RX serta mekanisme komunikasi yang digunakan. Menggunakan modul komunikasi NRF24L01 pada kedua perangkat serta protokol SPI untuk mengatur komunikasi data. Gambar 5.17 berikut merupakan implementasi modul komunikasi NRF24L01 pada sistem.



Gambar 5.21 Implementasi modul komunikasi NRF24L01

Setelah implementasi modul komunikasi NRF24L01 tahap berikutnya adalah implementasi kode program pada mikrokontroler. Modul komunikasi yang berada pada perangkat *transmitter* bertindak sebagai *slave* sedangkan yang berada pada perangkat *receiver* bertindak sebagai *master*. Berikut merupakan implementasi kode program pada kedua perangkat tersebut dalam mengirimkan data berupa pwm dari perangkat *transmitter* (TX) menuju perangkat (RX).

Tabel 5.19 *Library* perangkat TX

Kode program : Perangkat TX	
1	`#include <SPI.h>`
2	`#include <Mirf.h>`
3	`#include <nRF24L01.h>`
4	`#include <MirfHardwareSpiDriver.h>`

Pada Tabel 5.19 merupakan kode program untuk memanggil *library* pada arduino uno IDE. Fungsi *library* arduino adalah kumpulan subrutin atau *file-file* yang di dalamnya berisi kumpulan definisi/perintah, deklarasi konstanta & variabel, *class* dan fungsi-fungsi tertentu. *Library* yang digunakan dalam perangkat TX ada empat, yakni SPI.h, Mirf.h, nRF24L01.h dan MirfHardwareSpiDriver.h.

Masing-masing dari *library* diatas memiliki fungsi dan peran tertentu dalam mengirim data. Tabel 5.20 dibawah ini akan menjelaskan secara detail fungsi dari setiap *library* yang digunakan.

Tabel 5.20 Fungsi *library* perangkat TX

No	Nama *Library*	Fungsi
1	SPI.h	Menangani interface komunikasi serial sinkron SPI (Serial Pheripheral Interface) seperti ditujukan untuk tipe protokol komunikasi serial sinkron yang memiliki tiga jalur kabel yakni MISO,MOSI dan SCLK.
2	Mirf.h	Berisi kumpulan kode perintah yang digunakan untuk mengatur komunikasi NRF24L01. Misal pada kode program : Mirf.payload = 32; Mirf.channel = 11;
3	nRF24L01	Berisi memory map, instruksi mnemonics dan bit mnemonics.
4	MirfHardware SpiDriver.h	Untuk mengaktifkan driver komunikasi pada SPI pada saat komunikasi data

Setelah mendeklarasikan *library* yang digunakan untuk modul komunikasi maka Tabel 5.21 berikut akan mengimplementasikan potongan kode program pada perangkat *transmitter* untuk mengirimkan nilai ke perangkat *receiver*.

Tabel 5.21 Kode program mengirim nilai dari TX menuju RX

Kode program : Komunikasi TX menuju RX

```
1  #include <SPI.h>
2  #include <Mirf.h>
3  #include <nRF24L01.h>
4  #include <MirfHardwareSpiDriver.h>
5
6  char data[32];
7  int pwmKiri = 0;
8  int pwmKanan = 0;
9
10 void setup(){
11   Mirf.spi = &MirfHardwareSpi;// menggunakan port perangkat
12 keras SPI
13   Mirf.init();
14   Mirf.setRADDR((byte *)"clie1");
15   Mirf.payload = 32; // max 32 union
16   Mirf.channel = 11; // any channel:  1..120
17   Mirf.config();
18   Serial.begin(9600);
19 }
20
21 void loop(){
22 sprintf(data,"%3d,%3d",pwmKiri, pwmKanan);
23
24   Mirf.setTADDR((byte *)"serv1");
```

```
25
26    Mirf.send((byte *)&data);
27
28    while(Mirf.isSending()){
29    }
30
31    delay(2);
32  }
```

Pada Tabel 5.21 baris 1-4 merupakan deklarasi *library* yang digunakan untuk proses komunikasi. Selanjutnya merupakan deklarasi variabel yang digunakan yakni *array* dengan nama "data" dengan tipe data *char*, pwmKiri dan pwmKanan dengan tipe data integer bernilai sama dengan nol.

Pada fungsi *setup* () berisi pengaturan komunikasi data. Pada baris ke-11 langkah utama adalah mengaktifkan *interface* SPI serta menggunakan *port* perangkat keras SPI kemudian mengatur *init* agar dapat mempersiapkan konfigurasi yang ada di dalam kode program supaya bisa dijalankan secara penuh dan optimal. Kemudian pada baris ke-14 Mengkonfigurasi alamat (5 *byte*) yang digunakan untuk mengidentifikasi modul "clie1" selama siaran. Mengatur *payload* dengan kapasitas muatan 32 *byte*, fungsi *payload* sendiri adalah data aktual yang diterima oleh sistem tujuan karena pada saat proses pengiriman data *overhead*, informasi *header* hanya akan dilucuti dari paket ketika mencapai tujuan. *Payload* pada klien dan server harus sama agar pengiriman data tidak terjadi *packet loss*. Dalam komunikasi SPI terdapat 128 saluran dari 0-127 dalam kode program diatas menggunakan *channel* 11 sebagai kanal atau saluran pengiriman data. Pada baris ke-17 konfigurasi NRF24L01 dengan nilai yang dipilih untuk saluran dan *payload*, aktifkan bagian penerimaan modul dan kosongkan *buffer* penerimaan. Kemudian untuk menentukan kecepatan pengiriman dan penerimaan data melalui *port* serial sebesar 9600 (*boudrate*).

Pada fungsi perulangan (*Void loop*) pada baris ke-22 melakukan format dan menyimpan serangkaian karakter dan nilai dalam *buffer array*. Daftar argumen apa pun dikonversi dan dikeluarkan sesuai dengan spesifikasi format yang sesuai dalam *format-string*. Sederhanannya adalah mengembalikan jumlah elemen yang berhasil dicetak. Dalam kode program tersebut menggunakan parameter data untuk menyimpan nilai pwmKiri dan pwmKanan. Peletakkan data juga diatur dengan menggunakan *printf special character* yakni "%3d" yang artinya mengontrol lebar integer dengan bilangan bulat dan lebar minimum tiga ruang. Kemudian pada baris ke-24 mengkonfigurasi alamat (5 *byte*) yang digunakan untuk mengidentifikasi modul (serv1) selama penerimaan. Kemudian baris selanjutnya Mengirim pesan ukuran "*payload*" dari data tabel *byte* "data" yang diberikan dalam konfigurasi sebelumnya. Pada bari ke-28 mengulang (menunggu) sampai pesan terkirim.

Selain implementasi kode program pada *transmitter*, perangkat *receiver* juga perlu melakukan pengaturan dan implementasi kode program. Tabel 5.22 berikut merupakan implementasi kode program pada perangkat *receiver* (RX).

Tabel 5.22 Kode program menerima nilai dari perangkat TX

Kode program : Komunikasi RX

```
1  #include <SPI.h>
2  #include <Mirf.h>
3  #include <nRF24L01.h>
4  #include <MirfHardwareSpiDriver.h>
5  char data[32];
6  void setup(){
7    Mirf.spi = &MirfHardwareSpi;
8    Mirf.init();
9    Mirf.setRADDR((byte *)"serv1");
10   Mirf.payload = 32; // max 32 union
11   Mirf.channel = 11; // any channel:  1..120
12   Mirf.config();
13   Serial.begin(9600);
14 }
15
16 void loop(){
17   if(!Mirf.isSending() && Mirf.dataReady()){
18   Mirf.getData((byte *)&data);
19   Serial.println(data);
20 }
21
```

Kode program diatas tidak jauh beda dengan kode program pada perangkat *transmitter*. Pada fungsi *setup* pada dasarnya sama yang membedakan hanya pada baris ke-9 yakni mengkonfigurasi alamat (5 *byte*) yang digunakan untuk mengidentifikasi modul (serv1) selama penerimaan. Pengaturan *SPI, init, boudrate, payload dan channel* harus sama agar terjadi sinkronisasi komunikasi antar perangkat. Pada fungsi perulangan terdapat kondisi apabila pesan telah diterima dan data telah siap maka mengambil data *payload* pada *array byte*.

5.2.5 Implementasi Indikator

Implementasi indikator merupakan penerapan seluruh komponen dan kode program yang telah disusun pada perancangan indikator. Komponen yang disusun pada indikator antara lain adalah lampu LED , resistor dan mikrokontroler arduino nano V3. Terdapat empat lampu LED berwarna hijau yang masing-masing mewakili gerakan pada Robot beroda dan gestur tangan pada pengguna. Berikut merupakan implementasi kode program dan *output* yang dihasilkan oleh lampu LED pada indikator.

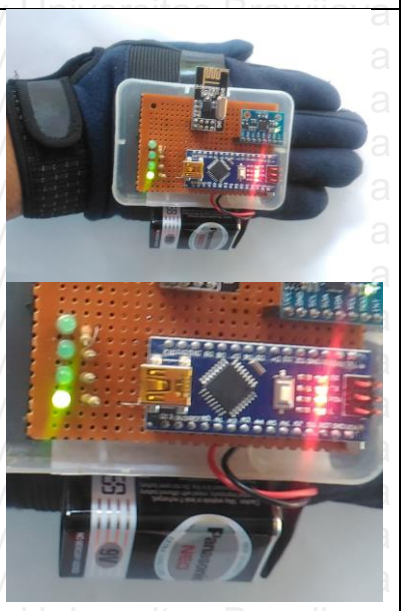
1. Indikator Robot beroda bergerak maju

   Pada indikator dibawah ini merupakan kondisi nyala lampu LED menurut *input* nilai pada pwm yang akan dikirimkan pada perangkat RX. Oleh karena itu tingkat akurasi antara gerakan Robot beroda dan nyala lampu LED tinggi. Lampu LED terbagi menjadi 4 yakni ; myLed 0, myLed 1, myLed 2 dan myLed 3.

Repository Universitas Brawijaya

Tabel 5.23 Indikator maju

No	Kode Program	*Output* LED

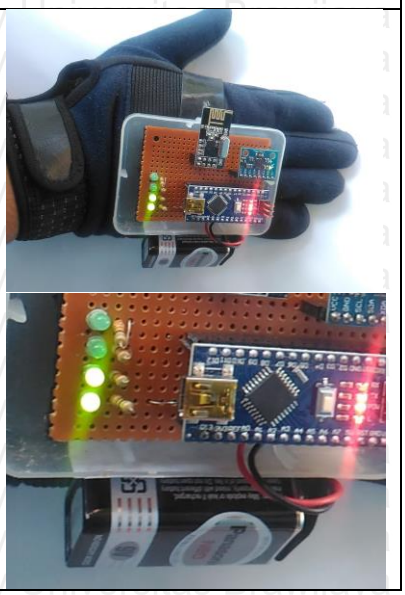
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
	`if(kalAngleX >= 0)`
	`{`
	`pwmRef = int(kalAngleX) + defaultPwm;`
	`pwmKiri = pwmRef - int(kalAngleY);`
	`pwmKanan = pwmRef + int(kalAngleY);`
	
	`if(pwmKiri < 0 ){ pwmKiri = 0; }`
	`if(pwmKiri > 255) { pwmKiri = 255; }`
	
	`if(pwmKanan < 0 ){ pwmKanan = 0; }`
	`if(pwmKanan > 255) { pwmKanan = 255; }`
	
	`digitalWrite(myLed[0],LOW);`
	`digitalWrite(myLed[3],HIGH);`
		

Pada Tabel 5.23 pada baris 1-19 merupakan kondisi apabila terdapat nilai PWM kanan dan kiri sama bernilai 255 maka gerakan Robot beroda akan maju mengakibatkan kondisi myLed 3 yang menyala sedangkan myLed lainnya dalam kondisi padam.

2. Indikator Robot beroda bergerak ke kanan

Tabel 5.24 Indikator ke kanan

No	Kode Program	*Output* LED

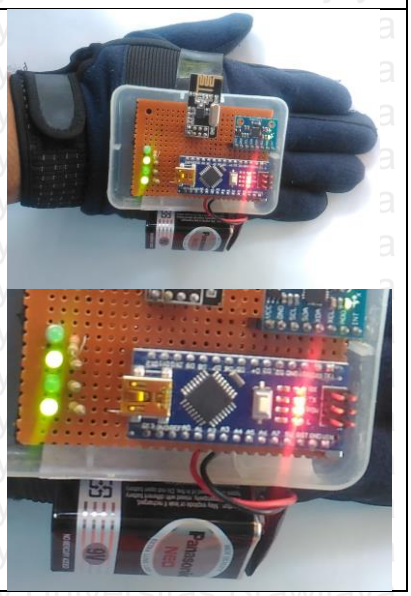
1
2
3
4
5
	`if(pwmKanan > pwmKiri)`
	`{`
	`digitalWrite(myLed[1],LOW);`
	`digitalWrite(myLed[2],HIGH);`
	`}`
		

Pada Tabel 5.24 diatas, saat indikator kekanan terjadi apabila pwmKanan lebih besar dari pwmKiri maka myLed yang nyala nomor 2 dan myLed nomor 1 padam. Pada dasarnya myLed yang menunujukkan kondisi maju dan mundur adalah myLed 0 dan 3 sedangkan untuk kondisi kanan dan kiri adalah myLed 1 dan 2. Pada kondisi diatas myLed yang menyala adalah nomor 2 dan 3 artinya gerakan Robot beroda ke kanan namun ada gerakan maju yang tidak terlalu signifikan.

3. Indikator Robot beroda bergerak ke kiri

Tabel 5.25 Indikator ke kiri

No	Kode Program	*Output* LED

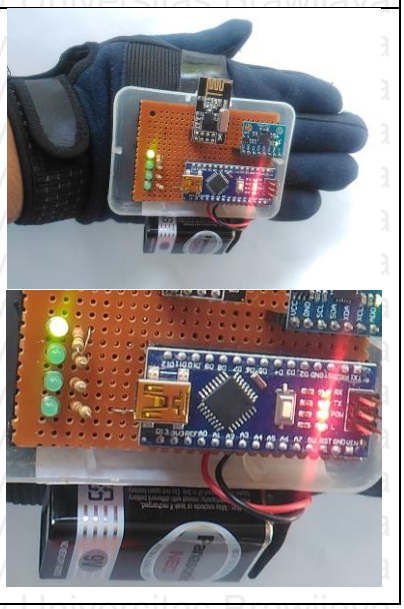
1
2
4
5
6
	`else if(pwmKanan < pwmKiri)`
	`{`
	`digitalWrite(myLed[1], HIGH);`
	`digitalWrite(myLed[2], LOW);`
	`}`
		

Pada Tabel 5.25 diatas, saat indikator kekanan terjadi apabila pwmKanan lebih kecil dari pwmKiri maka myLed yang nyala nomor 1 dan myLed nomor 2 padam. Karena terdapat kondisi dalam kondisi pada kode program diatas yang menyebabkan kondisi tersebut menjadi kebalikan dari kondisi saat gerak ke kanan. Seperti halnya kondisi pada saat bergerak kenan myLed yang menyala adalah nomor 1 dan 3 artinya gerakan Robot beroda ke kiri namun ada gerakan maju yang tidak terlalu signifikan.

4. Indikator Robot beroda bergerak mundur

Tabel 5.26 Indikator mundur

No	Kode Program	*Output* LED

1
2
4
5
6
7
8
9
	else
	{
	digitalWrite(myLed[0],
	HIGH);
	digitalWrite(myLed[3],
	LOW);
	
	}
		

Pada Tabel 5.26 merupakan kondisi apabila terdapat nilai pwm kanan dan kiri sama bernilai negatif atau selain dari kondisi yang telah dipaparkan diatas. Maka gerakan Robot beroda akan berlawanan dengan kondisi yang telah didefinisikan diatas. mengakibatkan kondisi myLed 0 yang menyala sedangkan myLed lainnya dalam kondisi padam dan gerakan pada Robot beroda mundur.

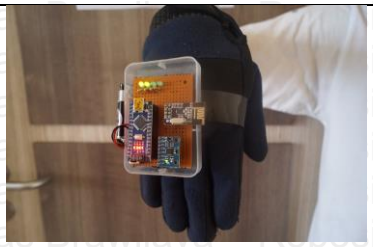
5.2.6 Implementasi Kendali

Implementasi pengendalian dibagi menjadi dua bagian yakni gestur tangan dan gerak robot. Gestur tangan adalah implementasi gerakan pada pengguna untuk mengendalikan Robot beroda. Sedangkan implementasi gerak robot merupakan implementasi gerakan motor DC terhadap masukkan dari perangkat TX atau gestur tangan pada pengguna.

5.2.6.1 Implementasi Gestur Tangan

Implementasi gerak tangan merupakan penerapan *gloves* pada tangan pengguna kemudian menggerakkan telapak tangan sesuai dengan gestur yang telah ditentukan. Gestur terbagi menjadi lima bagian menurut perancangan sebelumnya. Oleh karena itu pada Tabel 5.27 merupakan penerapan gerak tangan terhadap perangkat *transmitter* pada *gloves*.

Tabel 5.27 Implementasi gerak tangan

No	Gerak tangan	Gambaran gestur tangan	Keterangan
1	Sejajar dengan tanah		Dalam kondisi tersebut sudut *pitch* dan *roll* bernilai 0°. Sesuai dengan percangan kendali gestur maka dalam kondisi tersebut robot beroda dalam keadaan *idle* atau diam
2	Tangan ditekuk ke depan		Pada gambar tersebut pengguna mencoba menggerakkan gestur tangan sebesar -90° sehingga mengakibatkan laju robot beroda dalam keadaan maju serta dengan kecepatan maksimal yakni nilai PWM sebesar 255 dan tegangan maksimum 5V.
3	Tangan ditekuk belakang		Pada gambar tersebut pengguna mencoba menggerakkan gestur tangan sebesar 90° sehingga mengakibatkan laju robot beroda dalam keadaan mundur dengan kecepatan maksimal yakni nilai PWM sebesar 255 dan tegangan maksimum 5V.
4	Tangan miring ke kanan		Pada gambar tersebut pengguna mencoba menggerakkan gestur tangan sebesar 90° sehingga mengakibatkan laju robot beroda monuver ke kanan dengan kecepatan maksimal yakni nilai PWM sebesar 255 dan tegangan maksimum 5V.
5	Tangan miring ke kiri		Pada gambar tersebut pengguna mencoba menggerakkan gestur tangan sebesar -90° sehingga mengakibatkan laju robot beroda monuver ke kiri dengan kecepatan maksimal yakni nilai PWM sebesar 255 dan tegangan maksimum 5V.

REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

5.2.6.2 Implementasi Gerak Robot

Impelementasi gerak motor merupakan penerapan perangkat receiver atau Robot beroda untuk mampu bergerak sesuai dengan gestur tangan pengguna. Seperti yang telah di jelaskan pada implementasi gerak tangan bahwa gerakan yang dianjurkan dalam mengendalikan Robot beroda adalah lima gerakan. Secara garis besar gerak motor merupakan aktuator pada sistem. Dimana salah satu tujuan dari penelitian ini adalah mengendalikan navigasi pada Robot beroda. Oleh karena itu pada tahap ini akan mengimplementasikan gerakan Robot beroda seperti pada Tabel 5.28.

Tabel 5.28 Implementasi gerak motor

No	Gerak tangan	Gambaran gestur tangan
1	Sejajar dengan tanah	
2	Tangan ditekuk ke depan	

3	Tangan miring ke belakang	
4	Tangan miring ke kanan	
5	Tangan ditekuk ke kiri	

Berdasarkan implementasi tersebut gerakan robot beroda dapat ditentukan berdasarkan besar kecilnya sudut yang dihasilkan oleh perangkat pengedali (*gloves*). Pengguna mencoba mengimplementasikan lima gestur tangan yang telah disusun pada perancangan gestur.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

1. **Gestur tangan sejajar dengan tanah**

   Berhasil diimplementasikan pada robot beroda serta hasilnya adalah robot tidak bergerak sama sekali alias dalam keadaan diam. Namun apabila pengguna bergerak sedikit saja atau terjadi perubahan sudut dari 0° ke 1° maka robot beroda akan bergerak. Hal ini dikarenakan tingkat sensitivitas robot beroda yang tinggi yang diakibatkan nilai PWM acuan atau minimum yang diterapkan kode program bernilai 85 atau 1/3 dari tenaga maksimal robot beroda.

2. **Gestur tangan maju**

   Berhasil diimplementasikan pada robot beroda dan hasilnya adalah robot dapat bergerak maju sesuai dengan *input* pada *gloves*. Robot beroda dirancang dapat bergerak dua arah, yakni maju dan belok sehingga dapat tercipta gerakan serong atau kombinasi gerakan maju dan belok.

3. **Gestur tangan mundur**

   Berhasil diimplementasikan pada robot beroda dan hasilnya adalah robot dapat bergerak mundur sesuai dengan *input* pada *gloves*. Robot beroda dirancang dapat bergerak dua arah, yakni mundur dan belok sehingga dapat tercipta gerakan serong atau kombinasi gerakan maju dan belok.

4. **Gestur tangan miring ke kiri**

   Berhasil diimplementasikan pada robot beroda dan hasilnya adalah robot dapat bergerak berputar ke kiri sesuai dengan *input* pada *gloves*. Robot beroda dirancang dapat bergerak dua arah, yakni maju dan belok kiri atau sebaliknya sehingga dapat tercipta gerakan serong atau kombinasi gerakan maju dan belok.

5. **Gestur tangan miring ke kanan**

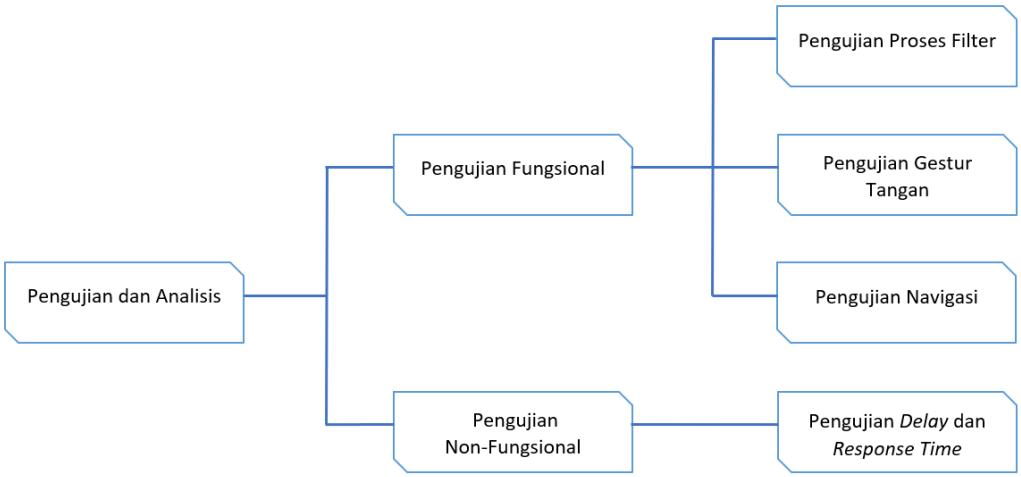
   Berhasil diimplementasikan pada robot beroda dan hasilnya adalah robot dapat bergerak berputar ke kanan sesuai dengan *input* pada *gloves*. Robot beroda dirancang dapat bergerak dua arah, yakni maju dan belok kanan atau sebaliknya sehingga dapat tercipta gerakan serong atau kombinasi gerakan maju dan belok.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

BAB 6 PENGUJIAN DAN ANALISIS

Tujuan pengujian ini adalah untuk membuktikan apakah sistem yang diimplementasikan telah memenuhi spesifikasi yang telah direncanakan sebelumnya. Hasil pengujian yang akan dimanfaatkan untuk menyempurnakan kinerja sistem dan sekaligus digunakan dalam pengembangan lebih lanjut.

Berdasarkan spesifikasi sistem yang telah dijalankan sebelumnya, maka dilakukan pengujian terhadap sistem menggunakan beberapa metode pengujian. Metode pengujian dipilih berdasarkan fungsi operasional dan beberapa parameter yang ingin diketahui dari sistem tersebut.



Gambar 6.1 Diagram alir pengujian dan analisis

Dalam Gambar 6.1 diatas dipilih dua macam metode pengujian, yaitu pengujian fungsional dan non-fungsional. Pengujian fungsional digunakan untuk membuktikan apakah sistem yang diimplementasikan dapat memenuhi persyaratan fungsi operasional yang direncanakan sebelumnya. Terdiri dari pengujian proses filter, pengujian gestur tangan dan pengujian navigasi. Sedangkan pengujian non-fungsional merupakan pengujian terhadap kinerja sistem secara keseluruhan bertujuan untuk memperoleh beberapa parameter yang dapat menunjukkan kemampuan dan kehandalan sistem dalam menjalankan fungsi operasionalnya. Pengujian non-fungsional terdiri dari pengujian *delay* dan *response time*. Dengan menggunakan dua metode penelitian di atas, diharapkan pada sistem dapat ditemukan kelebihan dan kekurangan dari artefak yang dibuat sehingga memudahkan jika dilakukan pengembangan lebih lanjut.

6.1 Pengujian Filter

Pengujian filter merupakan pengujian *kalman filter* sebagai metode filter sinyal pada sistem. Dengan tujuan untuk memperoleh sinyal yang konstan dan stabil sehingga dapat mengoptimalkan kinerja dari sistem. Filter kalman dapat diatur dengan menentukan variabel *Q_angle*, *Q_bias* dan *R_measure* dengan metode *try and error*. Nilai *Q_angle* merupakan nilai dari *accelerometer* dan *Q_bias* merupakan nilai dari *gyroscope* yang dapat diatur atau *tunning* berdasarkan seberapa besar kesalahan yang mungkin terjadi. Namun sebelum itu harus menentukan konstanta pada *complementary filter* untuk proses *sensor fusion* pada *accelerometer* dan *gyroscope* sehingga tidak menghambat kinerja pada *kalman filter*.

Complementary filter dapat diatur dengan mengubah nilai konstanta pada kode program arduino IDE. Nilai konstanta berada antara range 0 – 1 sehingga pada pengujian ini adalah mencari nilai konstanta dari range tersebut. Namun tidak semua konstanta dari range tersebut akan di uji, tahap pengujian ini meneliti kontanta jarak yang mendekati nilai optimal dalam kinerja *kalman filter* adalah dari range 0.93 – 0.97. Oleh karena itu pada tahap selanjutnya adalah mendapatkan nilai konstanta dengan menguji nilai konstanta yang mendekati nilai optimal untuk meredam gangguan sinyal dan membuat sinyal lebih stabil.

6.1.1 Pengujian Konstanta *Complementary Filter*

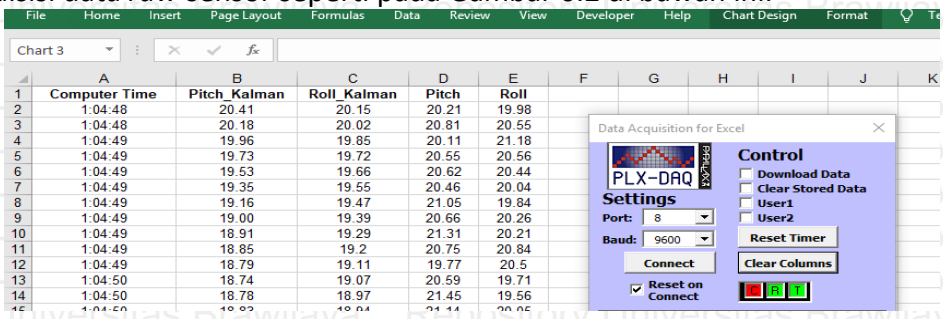
6.1.1.1 Tujuan Pengujian

Pengujian ini dilakukan untuk mendapatkan nilai konstanta yang sesuai dan tepat bagi proses filter. Sehingga pemilihan konstanta yang tepat dapat mengoptimalkan sistem.

6.1.1.2 Presedur Pengujian

Adapun prosedur yang perlu diperhatikan dalam melakukan pengujian tersebut, diantaranya adalah :

1. Menyiapkan perangkat *transmitter* (TX), kemudian hubungkan dengan PC.
2. Gunakan arduino IDE untuk *user interface*.
3. Buka program PLX-DAQ (*Parallax Data Acquisition*) dan Microsoft Excel untuk akuisisi data *raw* sensor seperti pada Gambar 6.2 di bawah ini.



Gambar 6.2 Tampilan program PLX-DAQ dan microsoft excel

4. Jalankan kode program yang telah dibuat sesuai dengan pengujian
5. Lakukan pengujian dengan melakukan *input* konstanta sebagai berikut; 0.93, 0.94, 0.95, 0.96, dan 0.97.
6. Hasil pengujian berupa data *raw* nilai *pitch* dan *roll* akan diakuisisi pada microsoft excel seperti pada Gambar 6.2 di atas.
7. Pada microsoft Excel akan ditampilkan data dalam bentuk tabel dan grafik perbandingan antara data menggunakan *kalman filter* dan tidak menggunakan *kalman filter*.
8. Menggunakan Nilai *Q_angle* = 0.001, *Q_bias* = 0.003 dan Nilai *R_measure* = 0.03 pada *library kalman filter.*

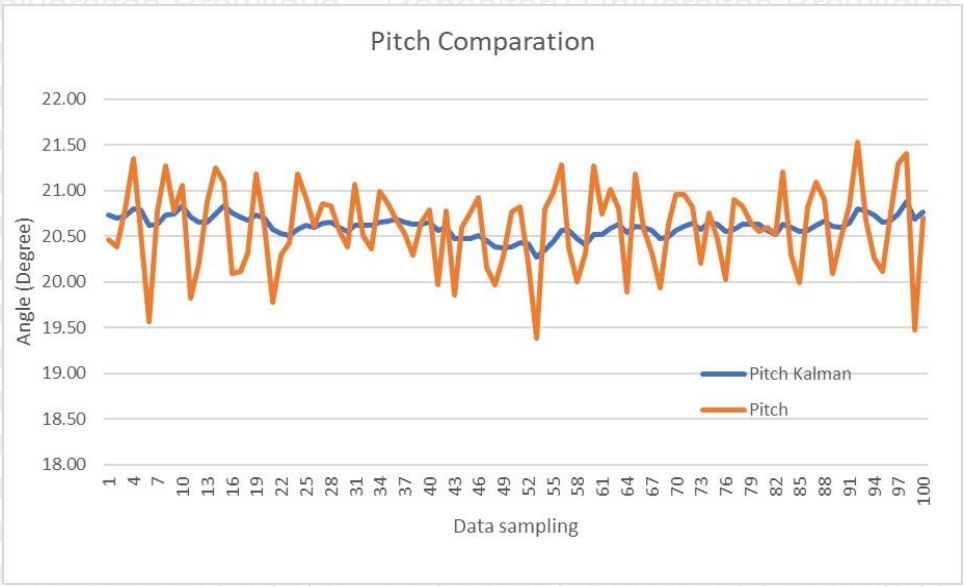
6.1.1.3 Pelaksanaan Pengujian

Pengujian ini dilakukan apabila seluruh prosedur pengujian telah dipersiapkan. Pengujian untuk mendapatkan nilai konstanta akan diuji berdasarkan gerakan statis pada perangkat *gloves*. Gerakan statis maksudnya adalah pada sudut tertentu misal pada pengujian ini menggunakan 20.50 derajat. Hal ini berlaku pada masing-masing konstanta serta sudut *pitch* dan *roll*-nya. Total data yang diakuisisi adalah 100 dengan kecepatan pengiriman data sebesar 0.1 ms. Kemudian dari data tersebut diambil secara acak lima data sebagai sampel dalam pengujian filter pada masing-masing konstanta.

6.1.1.4 Hasil Pengujian

Hasil yang didapatkan pada saat melakukan pengujian niai konstanta adalah sebagai berikut:

1. Pengujian dengan menggunakan konstanta 0.93
   a. Pengujian pada sudut *pitch*



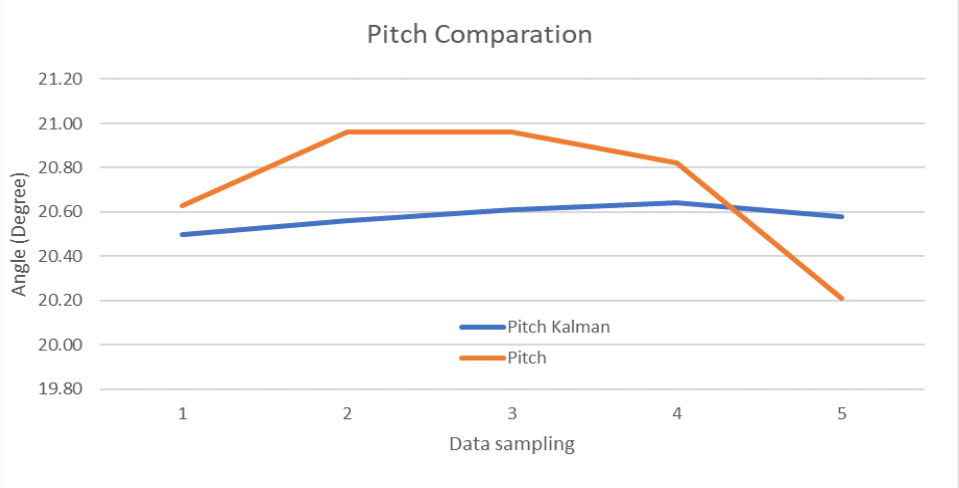
Gambar 6.3 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.93

Pada Gambar 6.3 merupakan perbandingan sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.1 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.1 Perbandingan lima data sampel sudut *pitch* k = 0.93

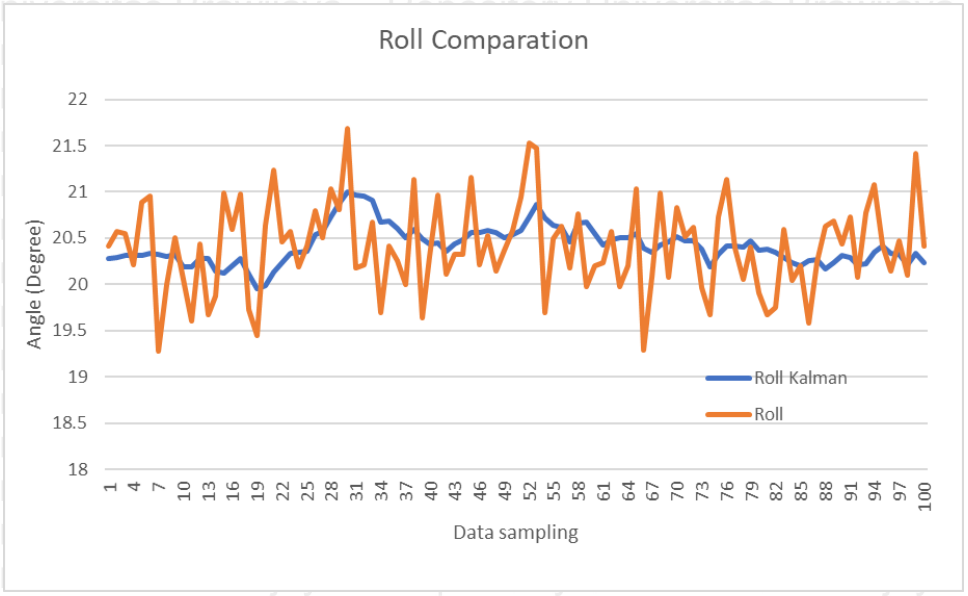
Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.50°	20.46°	20.63°	20.08°
2	20.56°	20.52°	20.96°	20.83°
3	20.61°	20.47°	20.96°	20.52°
4	20.64°	20.47°	20.82°	20.62°
5	20.58°	20.38°	20.21°	19.96°
Total	**102.89°**	**102.30°**	**103.58°**	**102.01°**
Rata - rata	**20.58°**	**20.46°**	**20.72°**	**20.40°**
Selisih sudut acuan (20.50)	**0.08°**	**0.04°**	**0.22°**	**0.10°**

Berdasarkan Tabel 6.1 dapat diperhatikan bahwa selisih sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.08° dan 0.22° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.4 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.4Grafik sudut *pitch* dengan 5 data sampel pada k = 0.93

b. Pengujian pada sudut *roll*



Gambar 6.5 Grafik sudut *roll* dengan 100 data sampel pada k = 0.93

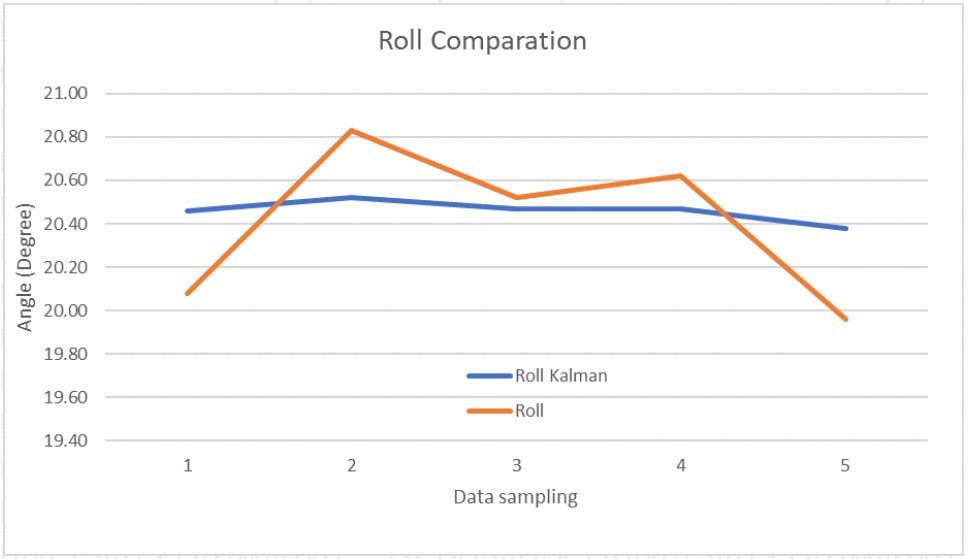
Pada Gambar 6.5 merupakan perbandingan sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.2 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.2 Perbandingan lima data sampel sudut *roll* k = 0.93

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.50°	20.46°	20.63°	20.08°
2	20.56°	20.52°	20.96°	20.83°
3	20.61°	20.47°	20.96°	20.52°
4	20.64°	20.47°	20.82°	20.62°
5	20.58°	20.38°	20.21°	19.96°
Total	**102.89°**	**102.30°**	**103.58°**	**102.01°**
Rata - rata	**20.58°**	**20.46°**	**20.72°**	**20.40°**
Selisih sudut acuan (20.50)	**0.08°**	**0.04°**	**0.22°**	**0.10°**

Berdasarkan Tabel 6.2 dapat diperhatikan bahwa selisih sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.04° dan 0.10° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara

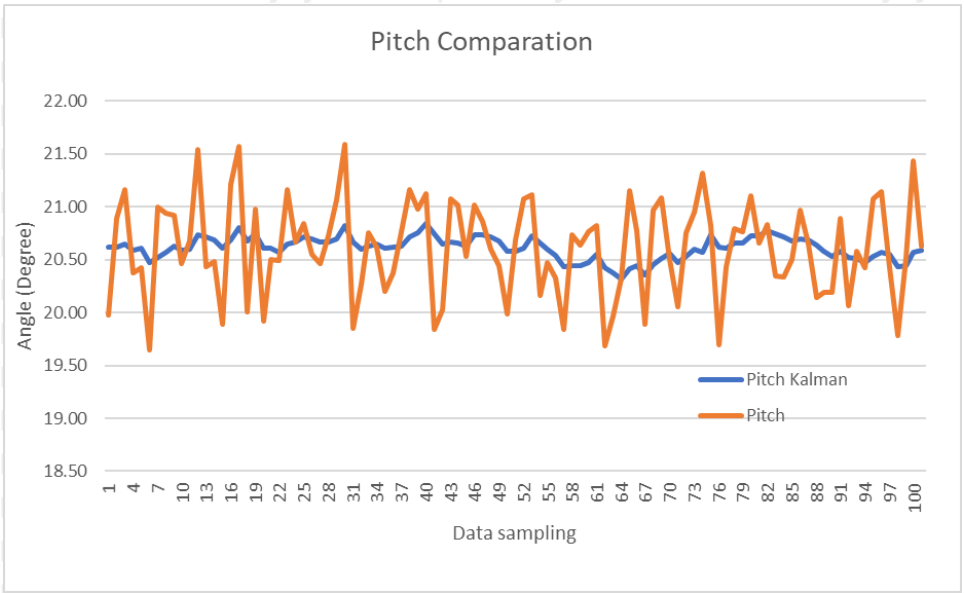
sudut *roll* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.6 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.6 Grafik sudut *roll* dengan 5 data sampel pada k = 0.93

2. Pengujian dengan menggunakan konstanta 0.94

a. Pengujian pada sudut *pitch*



Gambar 6.7 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.94

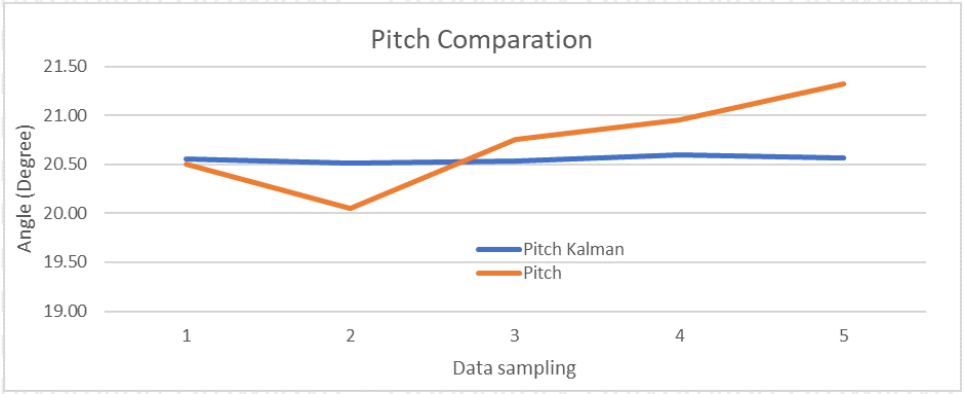
Pada Gambar 6.7 merupakan perbandingan sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui

perbandingan kedua gelombang tersebut maka pada Tabel 6.3 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.3 Perbandingan lima data sampel sudut *pitch* k = 0.94

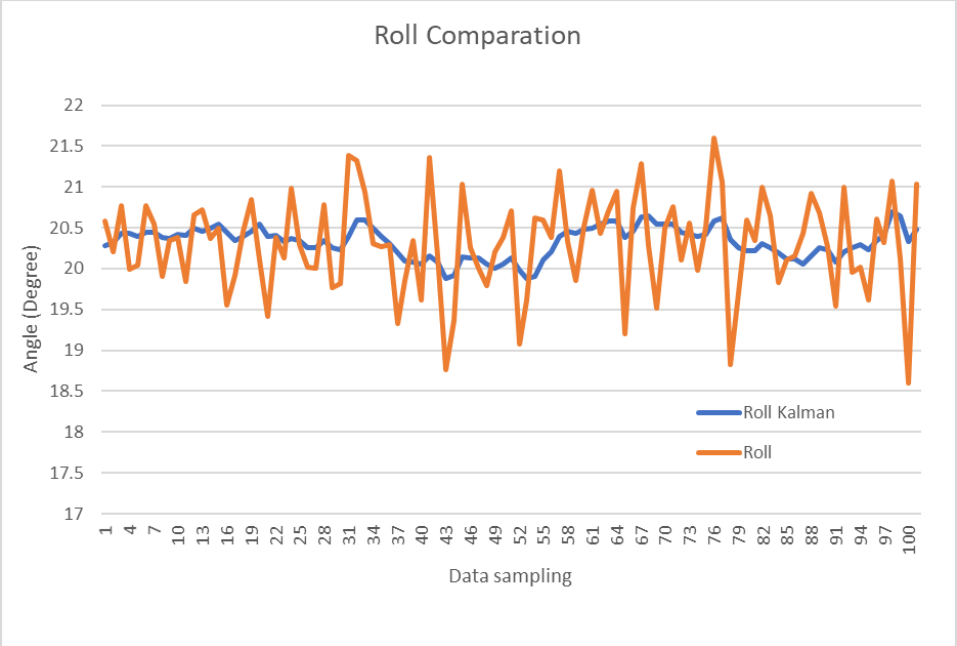
Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.56°	20.62°	20.50°	20.49°
2	20.51°	20.36°	20.05°	20.76°
3	20.53°	20.45°	20.75°	20.10°
4	20.60°	20.42°	20.95°	20.56°
5	20.57°	20.39°	21.32°	19.98°
Total	**102.77°**	**102.24°**	**103.57°**	**101.89°**
Rata - rata	**20.55°**	**20.45°**	**20.71°**	**20.38°**
Selisih sudut acuan (20.50)	**0.05°**	**0.05°**	**0.21°**	**0.12°**

Berdasarkan Tabel 6.3 dapat diperhatikan bahwa selisih sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.05° dan 0.21° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.8 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.8 Grafik sudut *pitch* dengan 5 data sampel pada k = 0.94

b. Pengujian pada sudut *roll*



Gambar 6.9 Grafik sudut *roll* dengan 100 data sampel pada k = 0.94

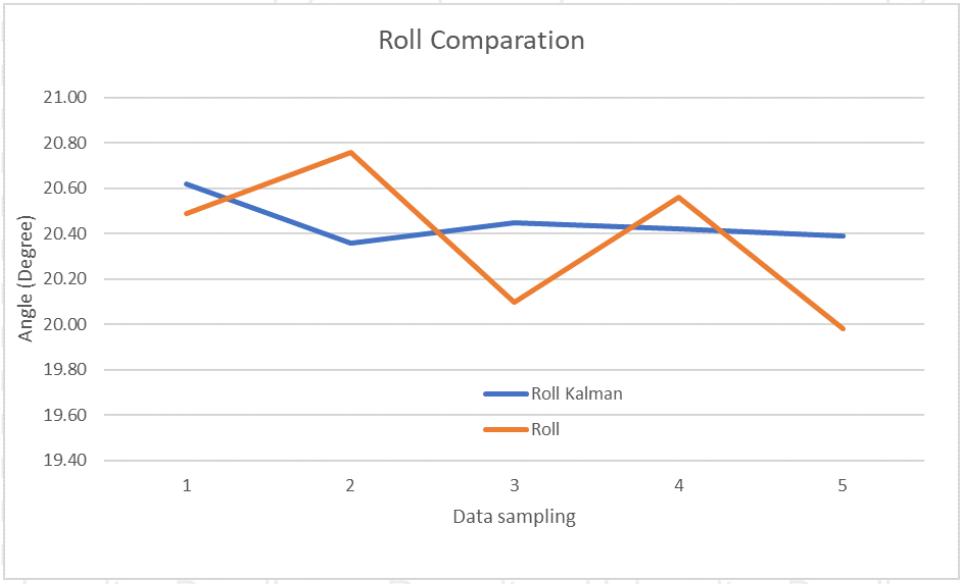
Pada Gambar 6.9 merupakan perbandingan sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.4 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.4 Perbandingan lima data sampel sudut *roll* k = 0.94

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.56°	20.62°	20.50°	20.49°
2	20.51°	20.36°	20.05°	20.76°
3	20.53°	20.45°	20.75°	20.10°
4	20.60°	20.42°	20.95°	20.56°
5	20.57°	20.39°	21.32°	19.98°
Total	**102.77°**	**102.24°**	**103.57°**	**101.89°**
Rata - rata	**20.55°**	**20.45°**	**20.71°**	**20.38°**
Selisih sudut acuan (20.50)	**0.05°**	**0.05°**	**0.21°**	**0.12°**

Berdasarkan Tabel 6.4 dapat diperhatikan bahwa selisih sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.05° dan 0.21° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara

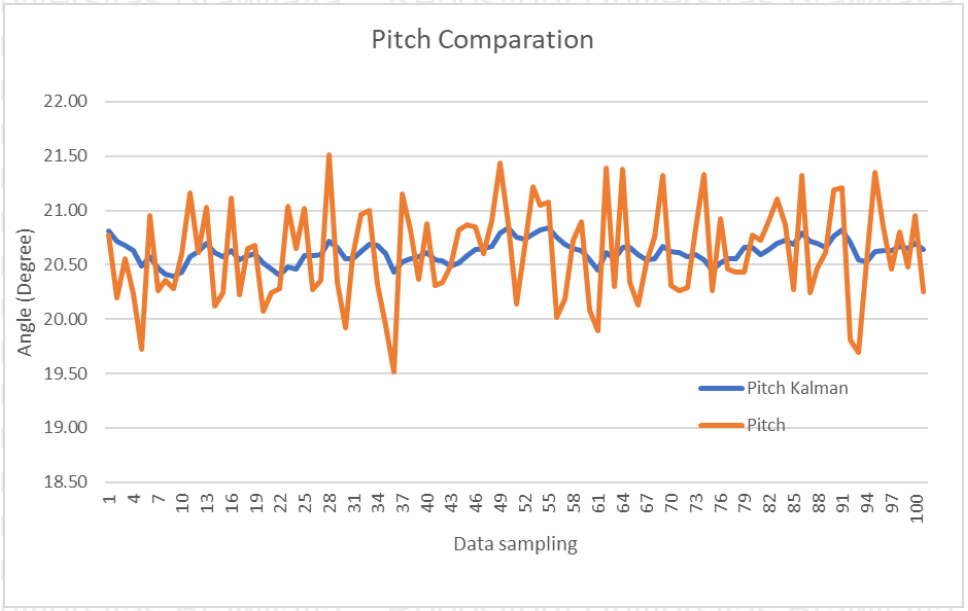
sudut *roll* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.10 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.10 Grafik sudut *roll* dengan 5 data sampel pada k = 0.94

3. Pengujian dengan menggunakan konstanta 0.95

   a. Pengujian pada sudut *pitch*



Gambar 6.11 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.95

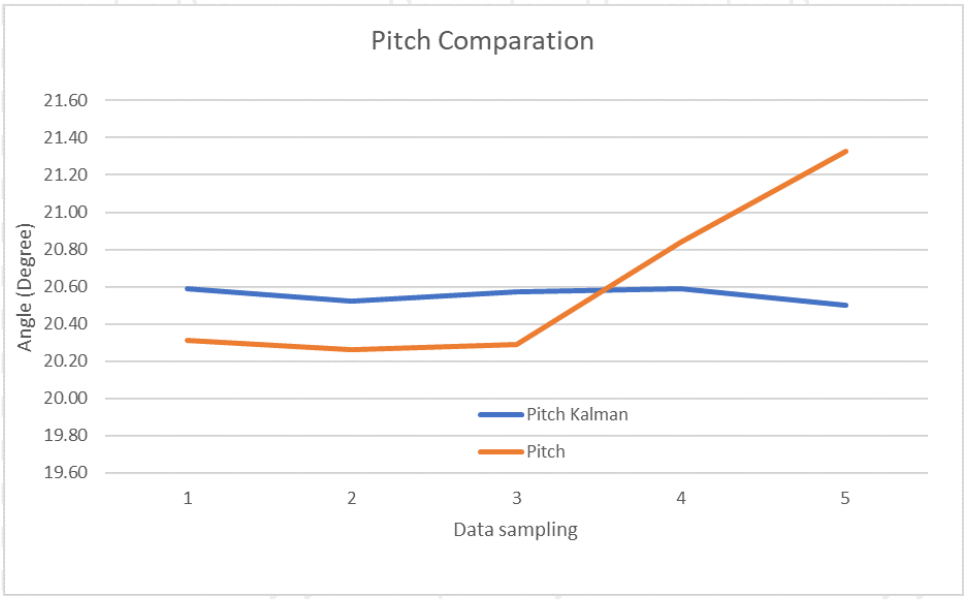
Pada Gambar 6.11 merupakan perbandingan sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada

gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.5 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.5 Perbandingan lima data sampel sudut *pitch* k = 0.95

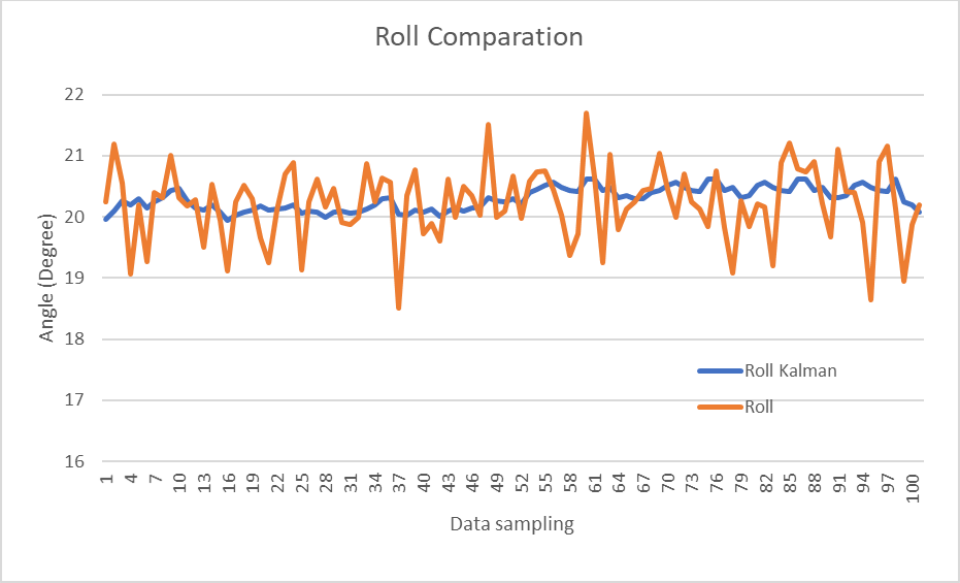
Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.59°	20.51°	20.31°	20.45°
2	20.52°	20.56°	20.26°	19.99°
3	20.57°	20.48°	20.29°	20.70°
4	20.59°	20.43°	20.84°	20.24°
5	20.50°	20.42°	21.33°	20.13°
Total	**102.77°**	**102.40°**	**103.03°**	**101.51°**
Rata - rata	**20.55°**	**20.48°**	**20.61°**	**20.30°**
Selisih sudut acuan (20.50)	**0.05°**	**0.02°**	**0.11°**	**0.20°**

Berdasarkan Tabel 6.5 dapat diperhatikan bahwa selisih sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.05° dan 0.11° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.12 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.12 Grafik sudut *pitch* dengan 5 data sampel pada k = 0.95

b. Pengujian pada sudut *roll*



Gambar 6.13 Grafik sudut *roll* dengan 100 data sampel pada k = 0.95

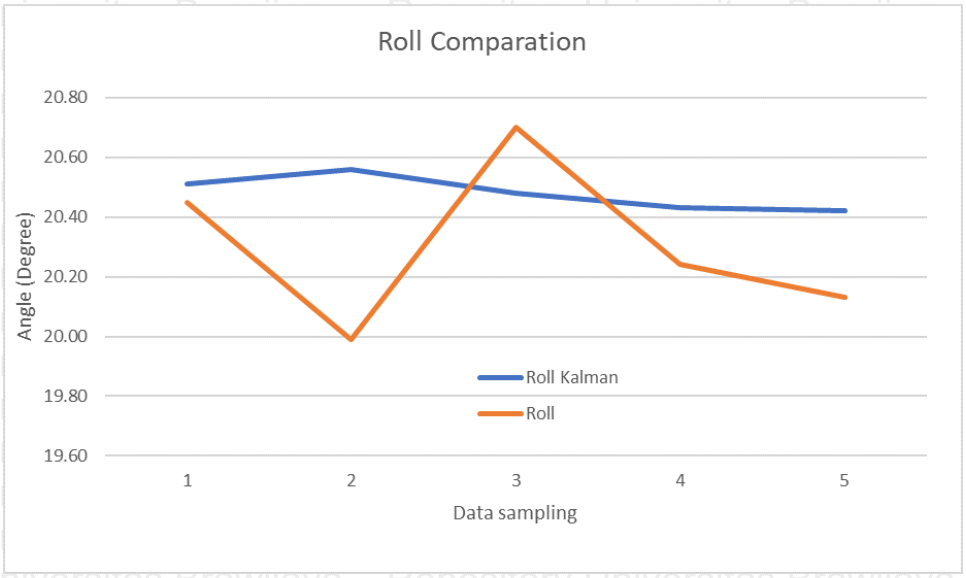
Pada Gambar 6.13 merupakan perbandingan sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.6 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.6 Perbandingan lima data sampel sudut *roll* k = 0.95

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.59°	20.51°	20.31°	20.45°
2	20.52°	20.56°	20.26°	19.99°
3	20.57°	20.48°	20.29°	20.70°
4	20.59°	20.43°	20.84°	20.24°
5	20.50°	20.42°	21.33°	20.13°
Total	**102.77°**	**102.40°**	**103.03°**	**101.51°**
Rata - rata	**20.55°**	**20.48°**	**20.61°**	**20.30°**
Selisih sudut acuan (20.50)	**0.05°**	**0.02°**	**0.11°**	**0.20°**

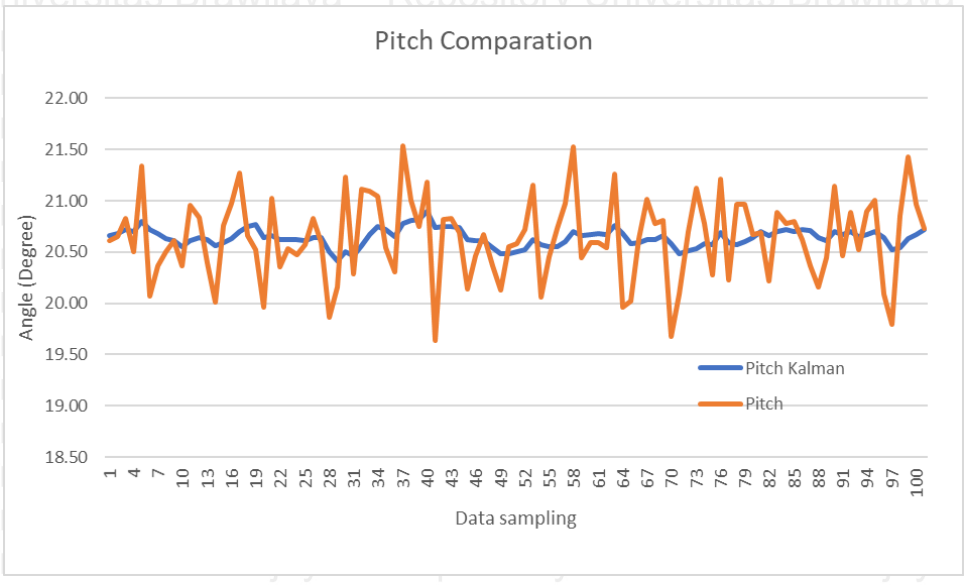
Berdasarkan Tabel 6.6 dapat diperhatikan bahwa selisih sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.02° dan 0.20° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara sudut *roll* dan sudut acuan dapat merepresentasikan kesalahan deteksi

sudut. Berikut Gambar 6.14 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.14 Grafik sudut *roll* dengan 5 data sampel pada k = 0.95

4. Pengujian dengan menggunakan konstanta 0.96
   a. Pengujian pada sudut *pitch*



Gambar 6.15 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.96

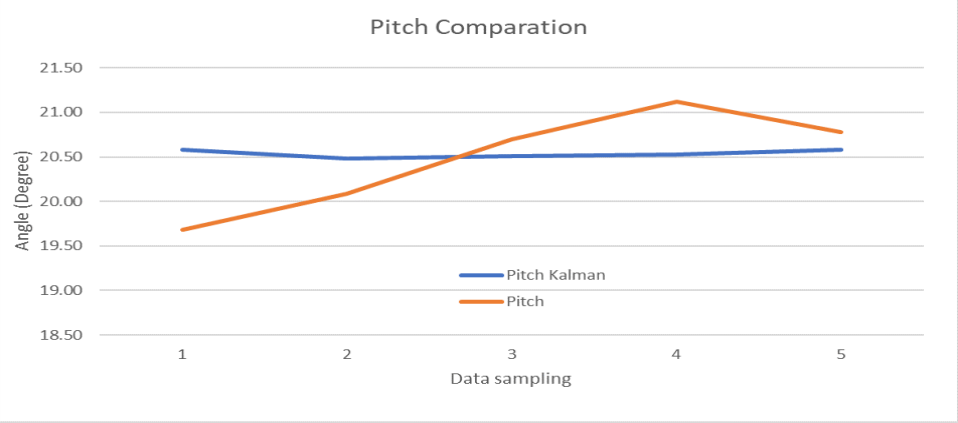
Pada Gambar 6.15 merupakan perbandingan sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui

perbandingan kedua gelombang tersebut maka pada Tabel 6.7 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.7 Perbandingan lima data sampel sudut *pitch* k = 0.96

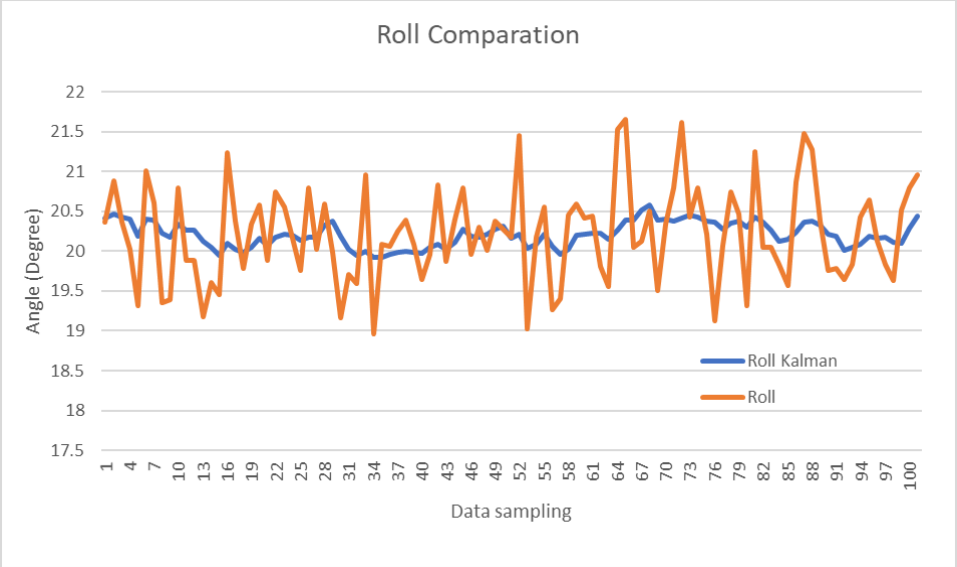
Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.58°	20.40°	19.68°	20.34°
2	20.48°	20.38°	20.09°	20.80°
3	20.51°	20.41°	20.58°	21.62°
4	20.53°	20.45°	21.12°	20.43°
5	20.58°	20.43°	20.78°	20.80°
Total	**102.68°**	**102.07°**	**102.37°**	**103.99°**
Rata - rata	**20.54°**	**20.41°**	**20.47°**	**20.80°**
Selisih sudut acuan (20.50)	**0.04°**	**0.09°**	**0.05°**	**0.30°**

Berdasarkan Tabel 6.7 dapat diperhatikan bahwa selisih sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.04° dan 0.05° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.16 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.16Grafik sudut *pitch* dengan 5 data sampel pada k = 0.96

b. Pengujian pada sudut *roll*



Gambar 6.17 Grafik sudut *roll* dengan 100 data sampel pada k = 0.96

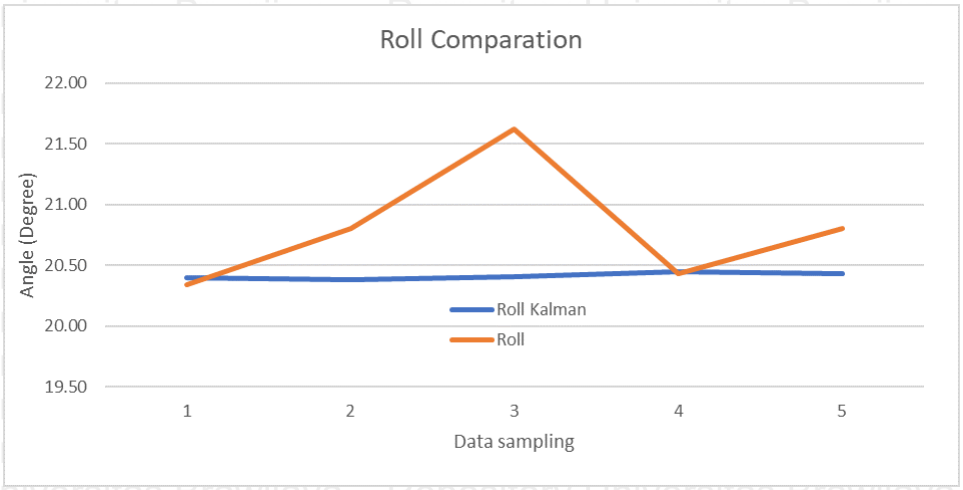
Pada Gambar 6.17 merupakan perbandingan sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.8 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.8 Perbandingan lima data sampel sudut *roll* k = 0.96

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.58°	20.40°	19.68°	20.34°
2	20.48°	20.38°	20.09°	20.80°
3	20.51°	20.41°	20.58°	21.62°
4	20.53°	20.45°	21.12°	20.43°
5	20.58°	20.43°	20.78°	20.80°
Total	**102.68°**	**102.07°**	**102.37°**	**103.99°**
Rata - rata	**20.54°**	**20.41°**	**20.47°**	**20.80°**
Selisih sudut acuan (20.50)	**0.04°**	**0.09°**	**0.05°**	**0.30°**

Berdasarkan Tabel 6.8 dapat diperhatikan bahwa selisih sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.09° dan 0.30° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara sudut *roll* dan sudut acuan dapat merepresentasikan kesalahan deteksi

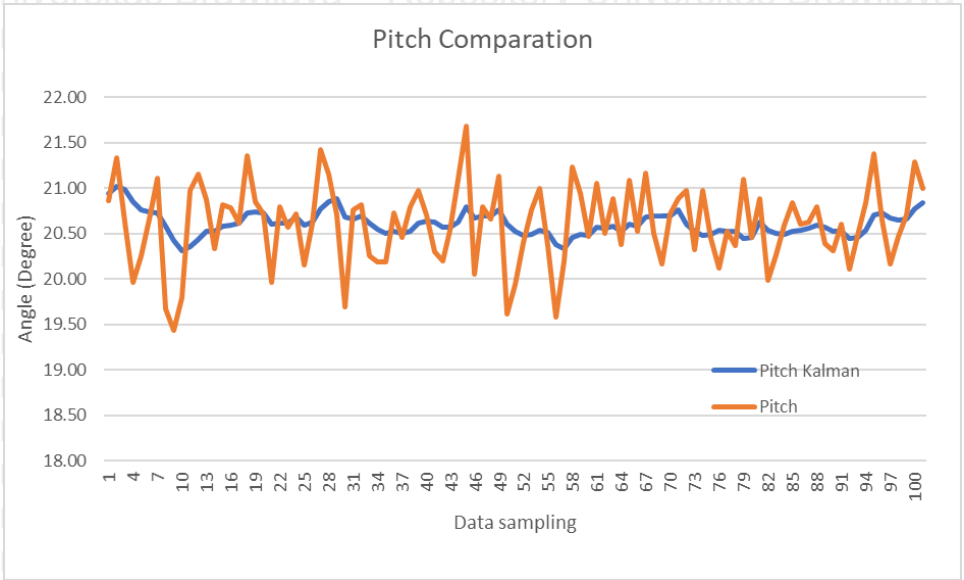
sudut. Berikut Gambar 6.18 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.18 Grafik sudut *roll* dengan 5 data sampel pada k = 0.96

5. Pengujian dengan menggunakan konstanta 0.97

a. Pengujian pada sudut *pitch*



Gambar 6.19 Grafik sudut *pitch* dengan 100 data sampel pada k = 0.97

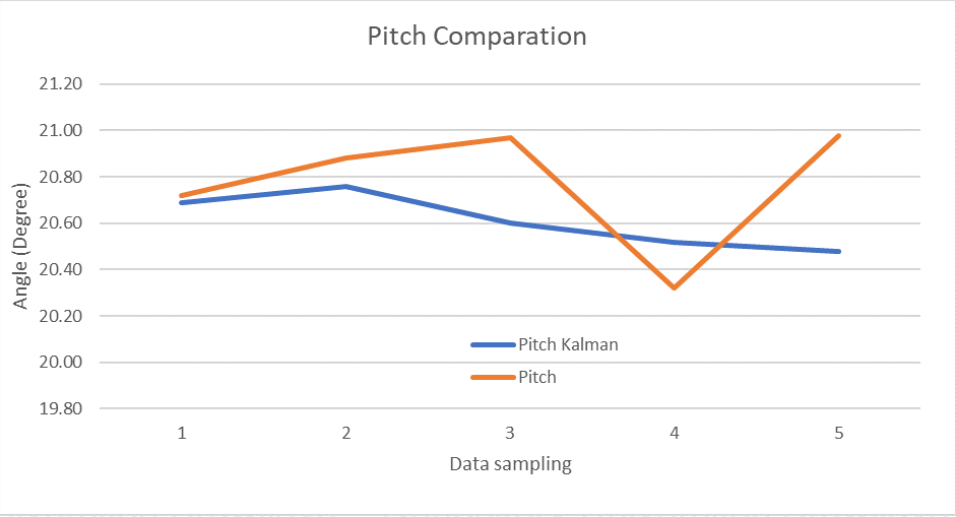
Pada Gambar 6.19 merupakan perbandingan sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.9 dibawah ini

merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.9 Perbandingan lima data sampel sudut *pitch* k = 0.97

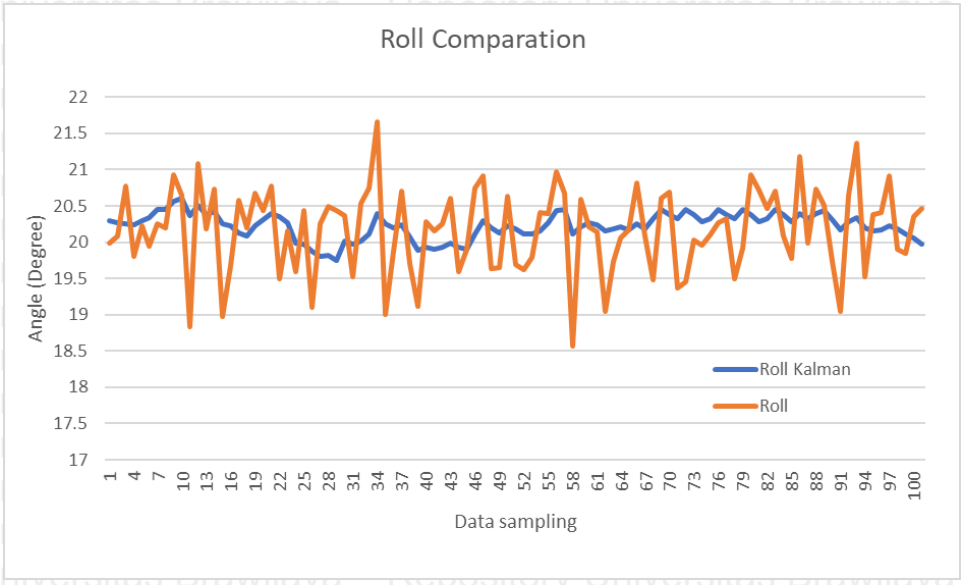
Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.69°	20.38°	20.72°	20.69°
2	20.76°	20.32°	20.88°	19.37°
3	20.60°	20.45°	20.97°	19.45°
4	20.52°	20.38°	20.32°	20.03°
5	20.48°	20.28°	20.98°	19.96°
Total	**103.05°**	**101.81°**	**103.87°**	**99.50°**
Rata - rata	**20.61°**	**20.36°**	**20.77°**	**19.90°**
Selisih sudut acuan (20.50)	**0.11°**	**0.14°**	**0.27°**	**0.60°**

Berdasarkan Tabel 6.9 dapat diperhatikan bahwa selisih sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.11° dan 0.27° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.20 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.20 Grafik sudut *pitch* dengan 5 data sampel pada k = 0.97

b. Pengujian pada sudut *roll*



Gambar 6.21 Grafik sudut *roll* dengan 100 data sampel pada k = 0.97

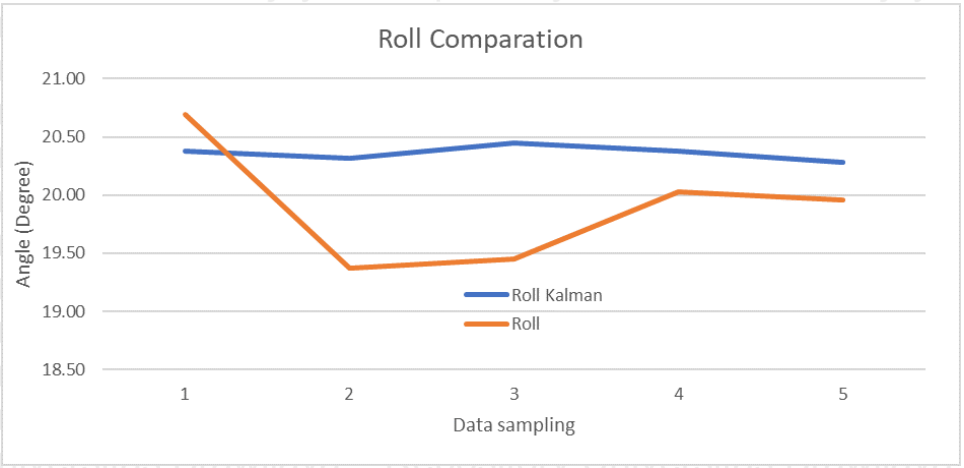
Pada Gambar 6.21 merupakan perbandingan sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.10 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.10 Perbandingan lima data sampel sudut *roll* k = 0.97

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	20.69°	20.38°	20.72°	20.69°
2	20.76°	20.32°	20.88°	19.37°
3	20.60°	20.45°	20.97°	19.45°
4	20.52°	20.38°	20.32°	20.03°
5	20.48°	20.28°	20.98°	19.96°
Total	**103.05°**	**101.81°**	**103.87°**	**99.50°**
Rata - rata	**20.61°**	**20.36°**	**20.77°**	**19.90°**
Selisih sudut acuan (20.50)	**0.11°**	**0.14°**	**0.27°**	**0.60°**

Berdasarkan Tabel 6.10 dapat diperhatikan bahwa selisih sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.14° dan 0.60° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 20.50°. Selisih antara sudut *roll* dan sudut acuan dapat merepresentasikan kesalahan deteksi

sudut. Berikut Gambar 6.22 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.22Grafik sudut *roll* dengan 5 data sampel pada k = 0.97

6.1.1.5 Analisis Hasil Pengujian

Berdasarkan hasil pengujian konstanta yang dilakukan diperoleh selisih sudut acuan seperti pada tabel dibawah ini.

Tabel 6.11 Perbandingan selisih sudut acuan kelima konstanta

Label	A	B	C	D	E	F
Konstanta	***Pitch*_Kalman**	***Roll*_Kalman**	***Pitch***	***Roll***	**A + B**	**C+D**
0.93	0.08°	0.04°	0.22°	0.10°	0.12°	0.31°
0.94	0.05°	0.05°	0.21°	0.12°	0.11°	0.34°
0.95	0.05°	0.02°	0.11°	0.20°	0.07°	0.30°
0.96	0.04°	0.09°	0.05°	0.30°	0.12°	0.35°
0.97	0.11°	0.14°	0.27°	0.60°	0.25°	0.87°

Analisis terhadap Tabel 6.11 diperoleh penjumlahan selisih terkecil sudut *pitch* dan *roll* menggunakan *kalman filter* adalah konstanta 0.94 dengan jumlah sudut sebesar 0.07° dan kontanta dengan selisih terbesar adalah konstanta 0.97 dengan jumlah sudut sebesar 0.25°. Beberapa faktor yang dapat mempengaruhi selisih sudut acuan adalah kemampuan sensor dalam mendeksi sudut acuan, kualitas rangkaian dan yang paling utama adalah proses filter. Oleh karena itu berdasarkan hasil pengujian diatas penulis menggunakan konstanta 0.95 pada sistem. Konstanta dengan selisih terkecil pada sudut acuan diharapkan mampu meminimalisir kesalahan pendeteksian sudut serta dapat mengoptimalkan redaman sinyal yang ditimbulkan oleh gangguan.

6.1.2 Pengujian Kalman Filter

6.1.2.1 Tujuan Pengujian

Menentukan variabel *Q_angle*, *Q_bias* dan *R_measure* dengan metode *try and error*. Nilai *Q_angle* merupakan nilai dari *accelerometer* dan *Q_bias* merupakan nilai dari *gyroscope* yang dapat diatur atau *tunning* berdasarkan seberapa besar kesalahan yang mungkin terjadi.

6.1.2.2 Prosedur Pengujian

Adapun prosedur yang perlu diperhatikan dalam melakukan pengujian tersebut, diantaranya adalah :

1. Menyiapkan perangkat *gloves*(TX), kemudian hubungkan dengan PC.
2. Gunakan arduino IDE untuk *user interface*.
3. Buka program PLX-DAQ (*Parallax Data Acquisition*) dan Microsoft Excel untuk akuisisi data *raw* sensor.
4. Jalankan kode program yang telah dibuat sesuai dengan pengujian
5. Lakukan pengujian variabel *process noise covariance matrix* pada *library* kalman.cpp berdasarkan :
   A. Skenario 1
      Q_angle = 0.001, *Q_bias* = 0.003 dan Nilai *R_measure* = 0.03
   B. Skenario 2
      Q_angle = 0.001, *Q_bias* = 0.050 dan Nilai *R_measure* = 0.03
   C. Skenario 3
      Q_angle = 0.001, *Q_bias* = 0.0001 dan Nilai *R_measure* = 0.01
6. Hasil pengujian berupa data *raw* nilai *pitch* dan *roll* akan diakuisisi pada microsoft excel.
7. Pada microsoft Excel akan ditampilkan data dalam bentuk tabel dan grafik perbandingan antara data menggunakan *kalman filter* dan tidak menggunakan *kalman filter*.
8. Menggunakan Nilai konstanta 0.95 sesuai dengan hasil uji perolehan pada *complementary filter.*

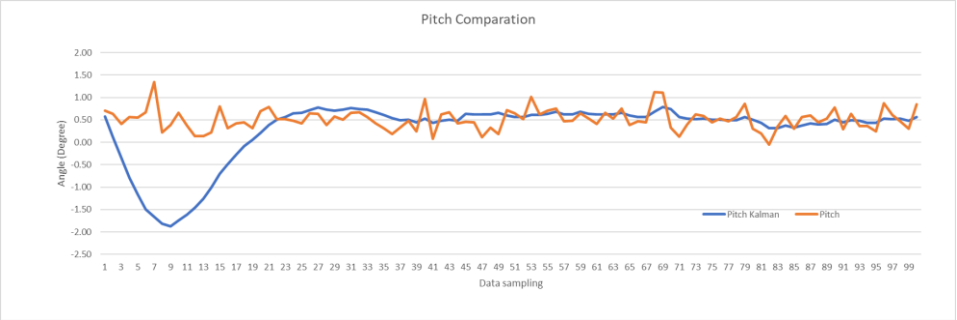
6.1.2.3 Pelaksanaan Pengujian

Pengujian ini dilakukan apabila kelengkapan seluruh prosedur pengujian telah terpenuhi. Pengujian dilaksanakan berdasarkan gerakan statis pada perangkat *gloves.* Gerakan statis maksudnya adalah pada sudut tertentu misal pada pengujian ini menggunakan 0 derajat. Hal ini berlaku pada masing-masing skenario. Total data yang diakuisisi adalah 100 dengan kecepatan pengiriman data sebesar 0.1 ms. Kemudian dari data tersebut diambil secara acak 10 data sebagai sampel dalam pengujian variabel *kalman filter*. Pengujian dilakukan dengan membandingkan jumlah standar deviasi dari sampel tersebut.

6.1.2.4 Hasil Pengujian

Hasil pengujian untuk menentukan variabel *Q_angle*, *Q_bias* dan *R_measure* adalah sebagai berikut.

1. Pengujian dengan menggunakan Skenario 1
   a. Pengujian pada sudut *pitch*



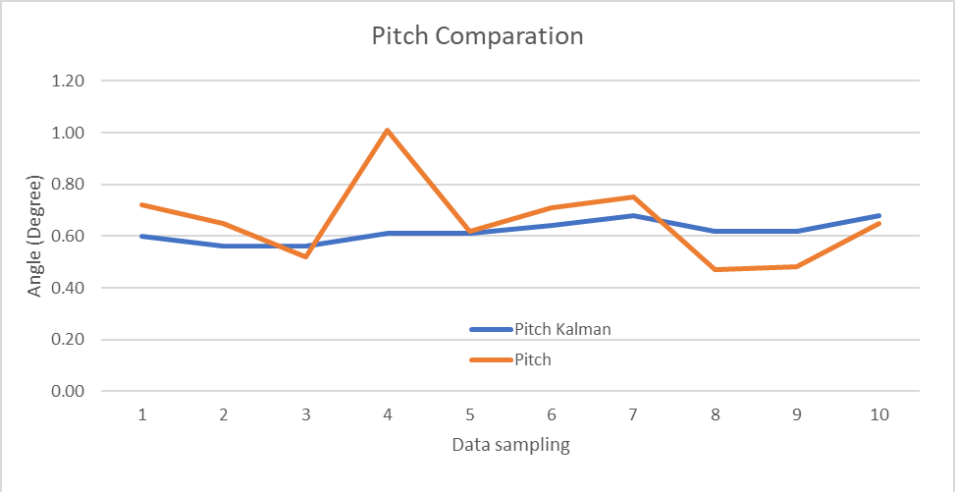
Gambar 6.23 Grafik sudut *pitch* dengan 100 data sampel skenario 1

Pada Gambar 6.23 merupakan perbandingan sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Menggunakan skenario 1 dengan nilai *Q_angle* = 0.001, *Q_bias* = 0.003 dan Nilai *R_measure* = 0.03. Dari percobaan tersebut sinyal data kalman filter (warna biru) dari 1 sampai 25 mengalami *rise time signal* hal tersebut wajar dialami dikarenakan kalman membutuhkan data sebelumnya (*Covariance noises process*) untuk dibandingkan. *Overshoot* sinyal dapat diminimalisir dengan mengimplementasikan variable tersebut. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.12 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.12 Perbandingan 10 data sampel sudut *pitch* skenario 1

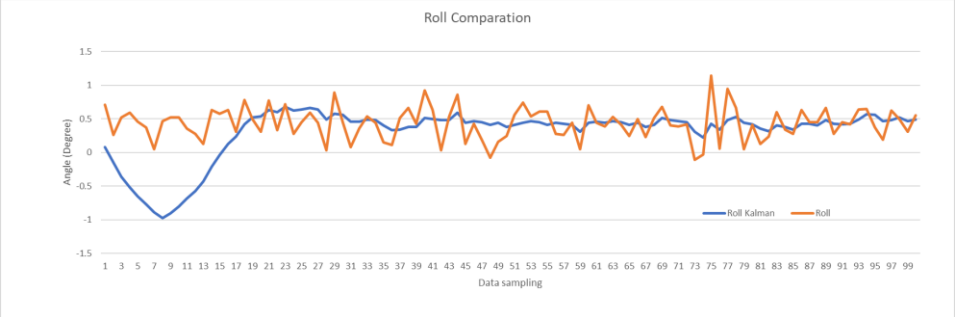
Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	0.60°	0.38°	0.72°	0.25°
2	0.56°	0.41°	0.65°	0.56°
3	0.56°	0.44°	0.52°	0.74°
4	0.61°	0.47°	1.01°	0.54°
5	0.61°	0.45°	0.62°	0.61°
6	0.64°	0.41°	0.71°	0.61°
7	0.68°	0.44°	0.75°	0.28°
8	0.62°	0.43°	0.47°	0.26°
9	0.62°	0.41°	0.48°	0.44°
10	0.68°	0.31°	0.65°	0.05°
Standar Deviasi	0.041°	0.045°	0.159°	0.216°

Berdasarkan Tabel 6.12 dapat diperhatikan bahwa jumlah standar deviasi sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.041° dan 0.159° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 0°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.24 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.24 Grafik sudut *pitch* dengan 10 data sampel skenario 1

b. Pengujian pada sudut *roll*



Gambar 6.25 Grafik sudut *roll* dengan 100 data sampel skenario 1

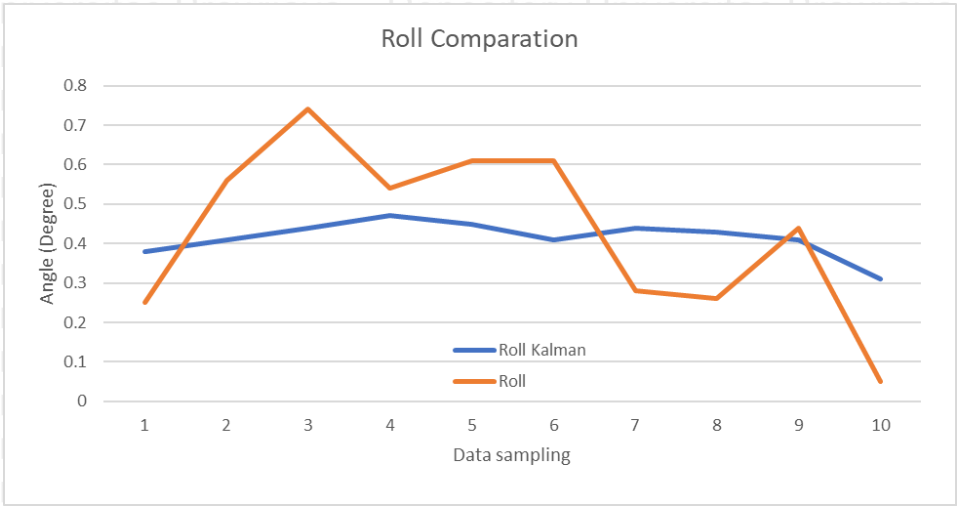
Pada Gambar 6.25 merupakan perbandingan sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Menggunakan skenario 1 dengan nilai *Q_angle* = 0.001, *Q_bias* = 0.003 dan Nilai *R_measure* = 0.03. Dari percobaan tersebut sinyal data kalman filter (warna biru) dari 1 sampai 21 mengalami *rise time signal* hal tersebut wajar dialami dikarenakan kalman membutuhkan data sebelumnya (*Covariance noises process*) untuk dibandingkan. *Overshoot* sinyal dapat diminimalisir dengan mengimplementasikan variable tersebut. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.5 dibawah ini

merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.13 Perbandingan 10 data sampel sudut *roll* skenario 1

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	0.60°	0.38°	0.72°	0.25°
2	0.56°	0.41°	0.65°	0.56°
3	0.56°	0.44°	0.52°	0.74°
4	0.61°	0.47°	1.01°	0.54°
5	0.61°	0.45°	0.62°	0.61°
6	0.64°	0.41°	0.71°	0.61°
7	0.68°	0.44°	0.75°	0.28°
8	0.62°	0.43°	0.47°	0.26°
9	0.62°	0.41°	0.48°	0.44°
10	0.68°	0.31°	0.65°	0.05°
Standar Deviasi	0.041°	0.045°	0.159°	0.216°

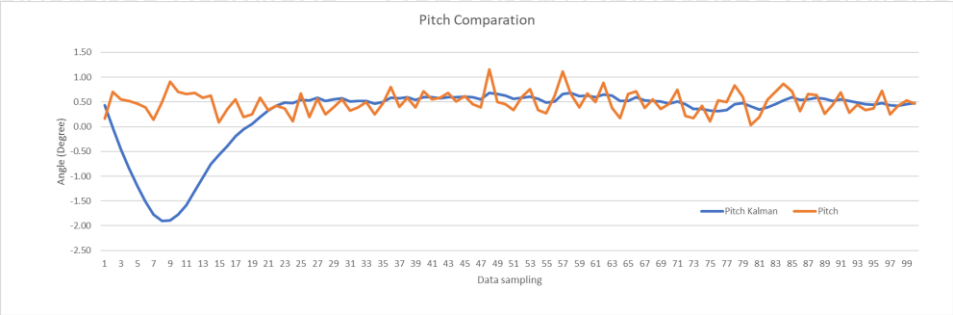
Berdasarkan Tabel 6.12 dapat diperhatikan bahwa jumlah standar deviasi sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.045° dan 0.216° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 0°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.12 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.26 Grafik sudut *roll* dengan 10 data sampel skenario 1

2. Pengujian dengan menggunakan Skenario 2

a. Pengujian pada sudut *pitch*



Gambar 6.27 Grafik sudut *pitch* dengan 100 data sampel skenario 2

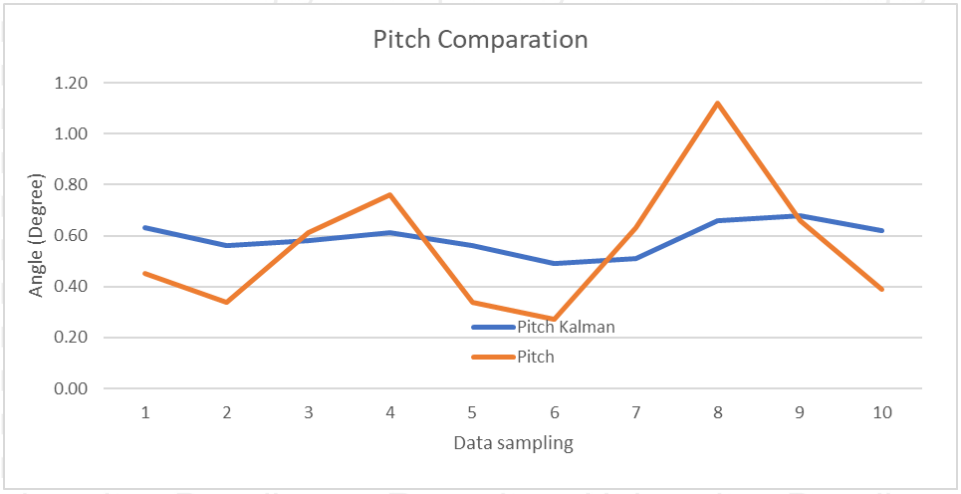
Pada Gambar 6.23 merupakan perbandingan sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Menggunakan skenario 2 dengan nilai *Q_angle* = 0.001, *Q_bias* = 0.050 dan Nilai *R_measure* = 0.03. Dari percobaan tersebut sinyal data kalman filter (warna biru) dari 1 sampai 25 mengalami *rise time signal* hal tersebut wajar dialami dikarenakan kalman membutuhkan data sebelumnya (*Covariance noises process*) untuk dibandingkan. *Overshoot* sinyal dapat diminimalisir dengan mengimplementasikan variable tersebut. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.14 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.14 Perbandingan 10 data sampel sudut *pitch* skenario 2

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	0.63°	0.13°	0.45°	0.14°
2	0.56°	0.25°	0.34°	0.73°
3	0.58°	0.09°	0.61°	0.10°
4	0.61°	0.12°	0.76°	0.31°
5	0.56°	0.20°	0.34°	0.50°
6	0.49°	0.35°	0.27°	0.85°
7	0.51°	0.30°	0.63°	0.20°
8	0.66°	0.33°	1.12°	0.46°
9	0.68°	0.33°	0.66°	0.36°
10	0.62°	0.26°	0.39°	-0.03°
Standar Deviasi	0.062°	0.096°	0.257°	0.279°

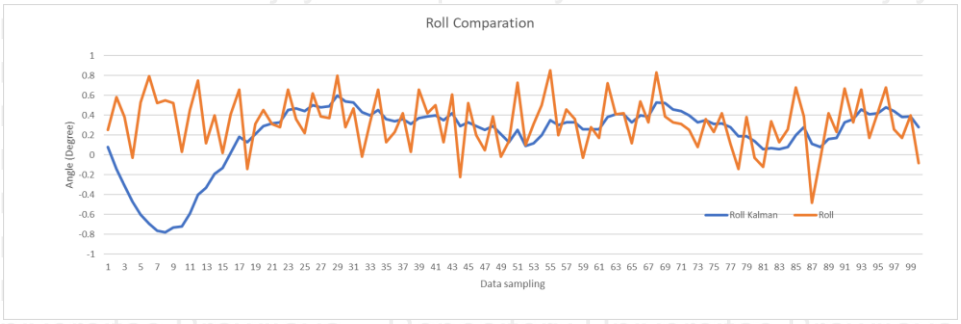
Berdasarkan Tabel 6.14 dapat diperhatikan bahwa jumlah standar deviasi sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.062° dan 0.257° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar

0°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.28 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.28 Grafik sudut *pitch* dengan 10 data sampel skenario 2

b. Pengujian pada sudut *roll*



Gambar 6.29 Grafik sudut *roll* dengan 100 data sampel skenario 2

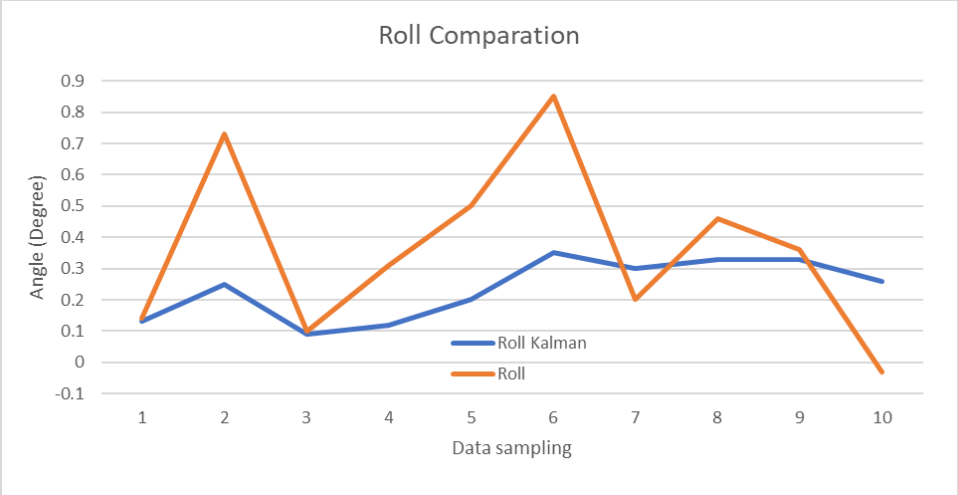
Pada Gambar 6.29 merupakan perbandingan sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Menggunakan skenario 2 dengan nilai *Q_angle* = 0.001, *Q_bias* = 0.050 dan Nilai *R_measure* = 0.03. Dari percobaan tersebut sinyal data kalman filter (warna biru) dari 1 sampai 21 mengalami *rise time signal* hal tersebut wajar dialami dikarenakan kalman membutuhkan data sebelumnya (*Covariance noises process*) untuk dibandingkan. *Overshoot* sinyal dapat diminimalisir dengan mengimplementasikan variable tersebut namun masih lebih baik hasil dari skenario 1 dari pada skenario 2. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.15 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Tabel 6.15 Perbandingan 10 data sampel sudut *roll* skenario 2

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	0.63°	0.13°	0.45°	0.14°
2	0.56°	0.25°	0.34°	0.73°
3	0.58°	0.09°	0.61°	0.10°
4	0.61°	0.12°	0.76°	0.31°
5	0.56°	0.20°	0.34°	0.50°
6	0.49°	0.35°	0.27°	0.85°
7	0.51°	0.30°	0.63°	0.20°
8	0.66°	0.33°	1.12°	0.46°
9	0.68°	0.33°	0.66°	0.36°
10	0.62°	0.26°	0.39°	-0.03°
Standar Deviasi	0.062°	0.096°	0.257°	0.279°

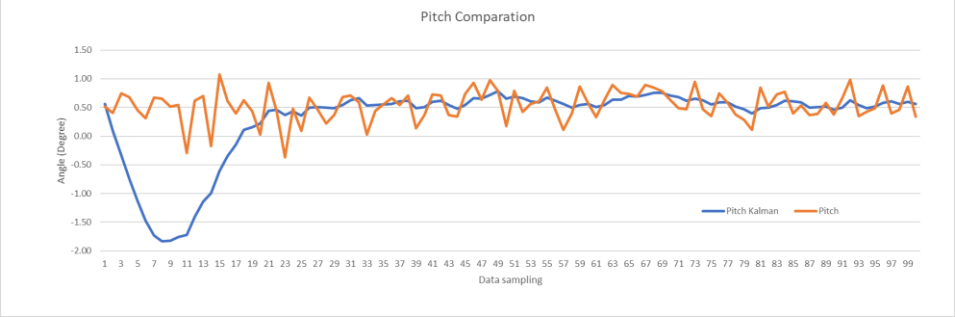
Berdasarkan Tabel 6.15 dapat diperhatikan bahwa jumlah standar deviasi sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.096° dan 0.279° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 0°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.12 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.30 Grafik sudut *roll* dengan 10 data sampel skenario 2

3. Pengujian dengan menggunakan Skenario 3

a. Pengujian pada sudut *pitch*



Gambar 6.31 Grafik sudut *pitch* dengan 100 data sampel skenario 3

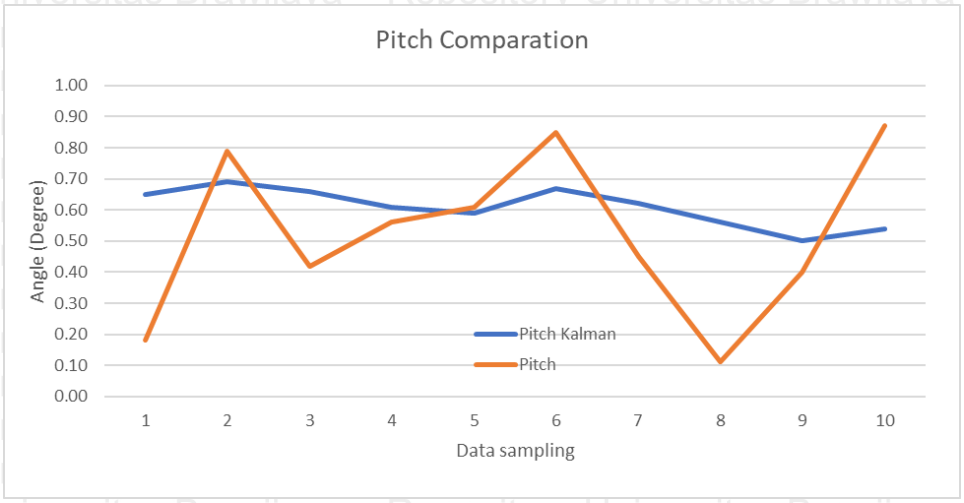
Pada Gambar 6.23 merupakan perbandingan sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Menggunakan skenario 3 dengan nilai *Q_angle* = 0.001, *Q_bias* = 0.0001 dan Nilai *R_measure* = 0.01. Dari percobaan tersebut sinyal data kalman filter (warna biru) dari 1 sampai 21 mengalami *rise time signal* hal tersebut wajar dialami dikarenakan kalman membutuhkan data sebelumnya (*Covariance noises process*) untuk dibandingkan. *Overshoot* sinyal dapat diminimalisir dengan mengimplementasikan variable tersebut. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.16 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.16 Perbandingan 10 data sampel sudut *pitch* skenario 3

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	0.65°	-0.58°	0.18°	-0.35°
2	0.69°	-0.37°	0.79°	0.32°
3	0.66°	-0.52°	0.42°	-1.02°
4	0.61°	-0.51°	0.56°	-0.48°
5	0.59°	-0.52°	0.61°	-0.45°
6	0.67°	-0.51°	0.85°	-0.50°
7	0.62°	-0.39°	0.45°	-0.03°
8	0.56°	-0.40°	0.11°	-0.55°
9	0.50°	-0.42°	0.40°	-0.53°
10	0.54°	-0.47°	0.87°	-0.55°
Standar Deviasi	0.062°	0.070°	0.264°	0.353°

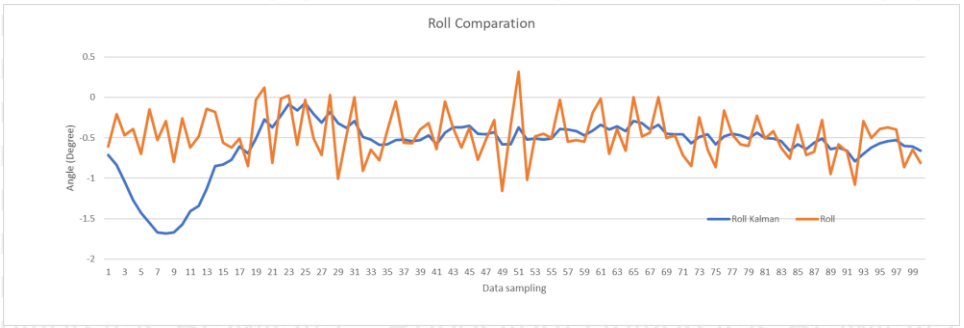
Berdasarkan Tabel 6.12 dapat diperhatikan bahwa jumlah standar deviasi sudut *pitch* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.062°dan 0.353° (yang ditandai dengan huruf

berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 0°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.32 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.32 Grafik sudut *pitch* dengan 10 data sampel skenario 3

b. Pengujian pada sudut *roll*



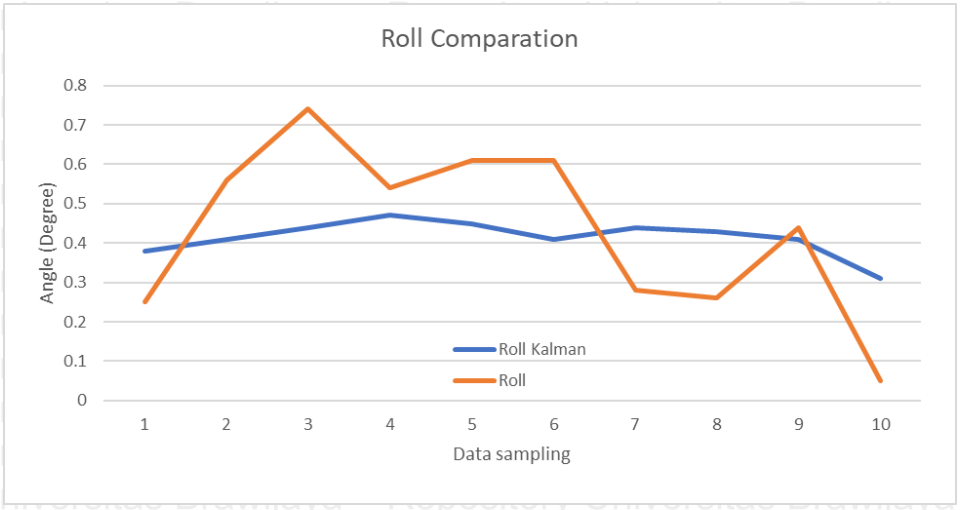
Gambar 6.33 Grafik sudut *roll* dengan 100 data sampel skenario 3

Pada Gambar 6.33 merupakan perbandingan sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter*. Menggunakan skenario 3 dengan nilai *Q_angle* = 0.001, *Q_bias* = 0.0001 dan Nilai *R_measure* = 0.01. Dari percobaan tersebut sinyal data kalman filter (warna biru) dari 1 sampai 19 mengalami *rise time signal* hal tersebut wajar dialami dikarenakan kalman membutuhkan data sebelumnya (*Covariance noises process*) untuk dibandingkan. *Overshoot* sinyal dapat diminimalisir dengan mengimplementasikan variable tersebut. Terdapat perbedaan yang signifikan antara kedua gelombang sinyal tersebut. Gelombang sinyal berwarna biru cenderung lebih stabil dari pada gelombang berwarna oranye. Namun untuk lebih rinci mengetahui perbandingan kedua gelombang tersebut maka pada Tabel 6.17 dibawah ini merupakan data sampel yang diambil berdasarkan total seratus data yang diakusisi di atas.

Tabel 6.17 Perbandingan 10 data sampel sudut *roll* skenario 3

Data ke	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	0.65°	-0.58°	0.18°	-0.35°
2	0.69°	-0.37°	0.79°	0.32°
3	0.66°	-0.52°	0.42°	-1.02°
4	0.61°	-0.51°	0.56°	-0.48°
5	0.59°	-0.52°	0.61°	-0.45°
6	0.67°	-0.51°	0.85°	-0.50°
7	0.62°	-0.39°	0.45°	-0.03°
8	0.56°	-0.40°	0.11°	-0.55°
9	0.50°	-0.42°	0.40°	-0.53°
10	0.54°	-0.47°	0.87°	-0.55°
Standar Deviasi	0.062°	0.070°	0.264°	0.353°

Berdasarkan Tabel 6.17 dapat diperhatikan bahwa jumlah standar deviasi sudut *roll* menggunakan *kalman filter* dan tanpa menggunakan *kalman filter* adalah 0.070° dan 0.353° (yang ditandai dengan huruf berwarna merah). Sudut acuan yang digunakan dalam pengujian sebesar 0°. Selisih antara sudut *pitch* dan sudut acuan dapat merepresentasikan kesalahan deteksi sudut. Berikut Gambar 6.34 merupakan grafik pengujian untuk data sampel di atas.



Gambar 6.34 Grafik sudut *roll* dengan 10 data sampel skenario 3

6.1.2.5 Analisis Hasil Pengujian

Berdasarkan hasil pengujian ketiga skenario tersebut diperoleh jumlah standar deviasi seperti pada tabel dibawah ini.

Tabel 6.18 Perbandingan standar deviasi

Skenario	*Pitch*_Kalman	*Roll*_Kalman	*Pitch*	*Roll*
1	0.041°	0.045°	0.159°	0.216°
2	0.062°	0.096°	0.257°	0.279°
3	0.062°	0.070°	0.264°	0.353°

Analisis terhadap Tabel 6.11, Berdasarkan hasil perolehan standar deviasi pada masing-masing skenario maka perolehan standar deviasi terkecil mampu meminimalisir jumlah variasi atau sebaran sejumlah nilai pada masing-masing sumbu. Oleh karena itu, diperoleh standar deviasi terkecil sudut *pitch* dan *roll* dari ketiga skenario tersebut adalah skenario 1 dengan nilai variabel *Q_angle* = 0.001, *Q_bias* = 0.003 dan Nilai *R_measure* = 0.03 . Skenario 1 memperoleh hasil standar deviasi *pitch* dan *roll* dengan jumlah 0.041° dan 0.045° menggunakan kalman filter serta 0.159° dan 0.216° tanpa menggunakan kalman filter. Sedangkan untuk perolehan standar deviasi terbesar diperoleh pada percobaan skenario 3 dengan nilai variabel *Q_angle* = 0.001, *Q_bias* = 0.0001 dan Nilai *R_measure* = 0.01. skenario 3 memperoleh hasil standar deviasi *pitch* dan *roll* dengan jumlah 0.062° dan 0.070° menggunakan kalman filter serta 0.264° dan 0.353° tanpa menggunakan kalman filter. Sedangkan untuk urutan kedua ditempati skenario 2 dengan variabel *Q_angle* = 0.001, *Q_bias* = 0.050 dan Nilai *R_measure* = 0.03. sehingga diperoleh hasil standar deviasi *pitch* dan *roll* dengan jumlah 0.062° dan 0.096° menggunakan kalman filter serta 0.257° dan 0.279° tanpa menggunakan kalman filter.

Beberapa faktor yang dapat mempengaruhi perolehan standar deviasi adalah ragam nilai atau sebaran sejumlah nilai pada masing-masing sumbu. Oleh karena itu berdasarkan hasil pengujian diatas penulis menggunakan skenario 1 pada sistem. Standar deviasi terkecil pada pengujian diatas diharapkan mampu meminimalisir kesalahan pendeteksian sudut serta dapat mengoptimalkan redaman sinyal yang ditimbulkan oleh gangguan.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

6.2 Pengujian Gestur Tangan

Pengujian gestur tangan merupakan pengujian perangkat pengendali robot beroda berdasarkan gestur tangan. Nilai sensor MPU-6050 yang telah mengalami pengujian proses filter juga diterapkan dalam pengujian.

6.2.1 Tujuan Pengujian

Pengujian ini diharapkan mampu untuk mengetahui pengaruh metode *kalman filter* terhadap gerakan gestur tangan pengguna berdasarkan sudut *pitch* dan *roll* untuk merepresentasikan gerakan maju,mundur,kiri dan kanan.

6.2.2 Prosedur Pengujian

Adapun prosedur yang perlu diperhatikan dalam melakukan pengujian tersebut, diantaranya adalah :

1. Menyiapkan perangkat *transmitter* (TX), kemudian hubungkan dengan PC
2. Gunakan arduino IDE untuk *user* interface
3. Buka program PLX-DAQ (*Parallax Data Acquisition*) dan Microsoft Excel untuk akuisisi 40 data *raw* sensor
4. Jalankan kode program yang telah dibuat sesuai dengan pengujian
5. Lakukan pengujian gestur tangan sesuai yang ditentukan pada tahap perancangan dan implementasi (dimulai pada sudut 0°) pada masing-masing sudut *pitch* dan *roll* kemudian bergerak naik sampai menuju 90°
6. Hasil pengujian berupa data *raw* nilai *pitch* dan *roll* akan diakuisisi berdasarkan gerakan maju,mundur,kiri dan kanan
7. Pada microsoft Excel akan ditampilkan data yang telah diakuisisi dalam bentuk tabel dan grafik perubahan sudut dari 0° sampai 90°
8. Amati data keluaran perubahan sudut pada saat melakukan gerakan pada setiap sudut *pitch* dan *roll*
9. Pengujian pada masing-masing bentuk gerakan dilakukan sebanyak 10 kali.

6.2.3 Pelaksanaan Pengujian

Pengujian ini dilakukan apabila kelengkapan seluruh prosedur pengujian telah terpenuhi. Pengujian dilakukan hanya menggunakan perangkat *transmitter* (TX) tanpa perlu menggunakan perangkat receiver (RX). Gerakan gestur tangan yang diujiakan adalah gerakan maju,mundur,kiri dan kanan.

6.2.4 -Hasil Pengujian

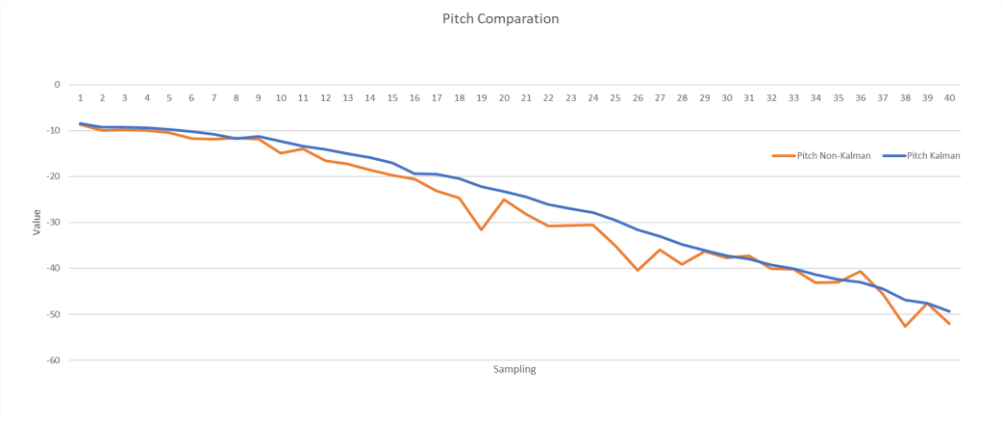
Hasil pengujian dilakukan dengan melihat rentang waktu tertentu pada saat melakukan gerakan gestur tangan. Grafik yang dihasilkan akan berbeda – beda berdasarkan data yang diakuisisi. Sehingga hasil yang didapatkan pada saat melakukan pengujian niai gestur tangan adalah sebagai berikut:

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

1. Pengujian dengan menggunakan gestur maju

Gambar 6.35 Gestur maju

Pada Gambar 6.35 Posisi tangan ketika melakukan gerakan gestur maju yakni ditekuk ke depan dengan jarak antara 0° hingga -90° derajat. Pada saat posisi 0° derajat tangan sejajar dengan tanah. Dalam tahap ini, ketika melakukan gerakan gestur maju data dari sudut *pitch* akan menjadi sudut acuan mendeteksi kemiringan tangan dan sudut *roll* dalam keadaan *idle*. Sedangkan Gambar 6.36 merupakan grafik yang dihasilkan adalah sebagai berikut.



Gambar 6.36 Grafik gestur maju

2. Pengujian dengan menggunakan gestur mundur

Gambar 6.37 Gestur mundur

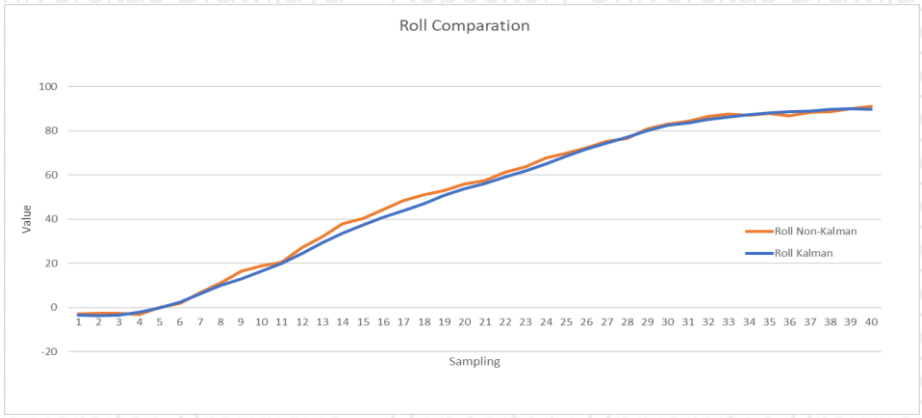
Pada Gambar 6.37 Posisi tangan ketika melakukan gerakan gestur mundur yakni ditekuk ke belakang dengan jarak antara 0° hingga 90°. Pada saat posisi 0° tangan sejajar dengan tanah. Dalam tahap ini, ketika melakukan gerakan gestur mundur data dari sudut *pitch* akan menjadi sudut acuan mendeteksi kemiringan tangan dan sudut *roll* dalam keadaan idle. Sedangkan Gambar 6.38 merupakan grafik yang dihasilkan adalah sebagai berikut.

Gambar 6.38 Grafik gestur mundur

3. Pengujian dengan menggunakan gestur kanan

Gambar 6.39 Gestur kanan

Pada Gambar 6.39 Posisi tangan ketika melakukan gerakan gestur kanan yakni ditekuk ke kanan dengan jarak antara 0° hingga 90°. Pada saat posisi 0° tangan sejajar dengan tanah. Dalam tahap ini, ketika melakukan gerakan gestur ke kanan data dari sudut *roll* akan menjadi sudut acuan mendeteksi kemiringan tangan dan sudut *pitch* dalam keadaan *idle*. Sedangkan Gambar 6.40 merupakan grafik yang dihasilkan adalah sebagai berikut.

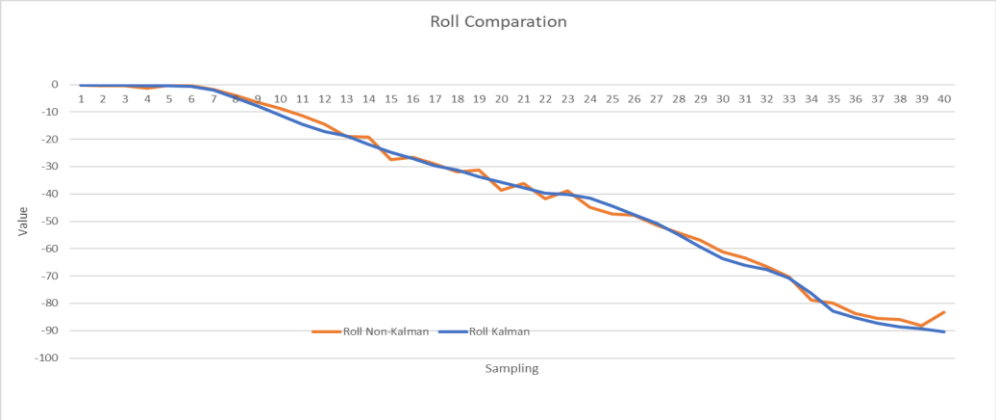


Gambar 6.40 Grafik Gestur kanan

4. Pengujian dengan menggunakan gestur kiri

Gambar 6.41 Gestur kiri

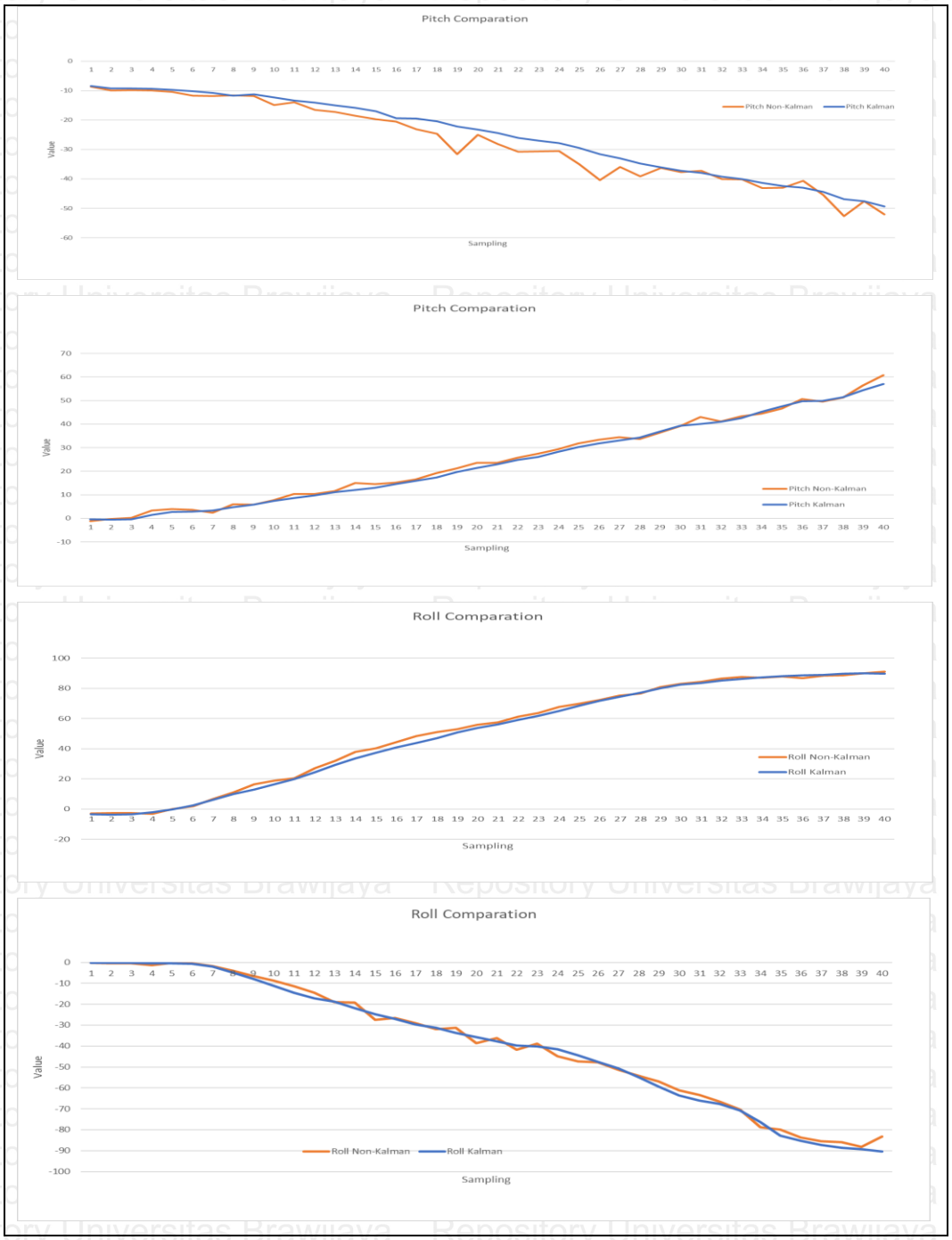
Pada Gambar 6.29 Posisi tangan ketika melakukan gerakan gestur kiri yakni ditekuk ke kiri dengan jarak antara 0° hingga -90°. Pada saat posisi 0° tangan sejajar dengan tanah. Dalam tahap ini, ketika melakukan gerakan gestur ke kiri data dari sudut *roll* akan menjadi sudut acuan mendeteksi kemiringan tangan dan sudut *pitch* dalam keadaan *idle*. Sedangkan Gambar 6.24 merupakan grafik yang dihasilkan adalah sebagai berikut.



Gambar 6.42 Grafik gestur kiri

6.2.5 Analisis Hasil Pengujian

Berdasarkan hasil pengujian gerakan gestur tangan terhadap empat bentuk gerakan diatas dapat dianalisis hasil perolehan sinyal berdasarkan perubahan sinyal ke-empat grafik pada Gambar 6.43. Perubahan yang terjadi adalah minimnya gangguan pada sinyal yang telah di filter menggunakan kalman. Serta penerapan konstanta sebesar 0.95 pada pengujian sebelumnya telah berhasil meredam *noise* yang terdapat pada sinyal.



Gambar 6.43 Grafik gestur maju,mundur,kiri dan kanan

Pada Gambar 6.43 gestur bekerja pada jarak antara 0° hingga 90° sesuai dengan prosedur pengujian. Namun pada saat kondisi gestur maju dan gestur kiri sinyal bernilai negatif. Hal tersebut akibat kondisi berlawan gerak antara gestur maju dengan gestur mundur serta gestur kiri dengan gestur kanan. Memudahkan dalam analisa sudut *pitch* dan *roll* pada sinyal. Serta berdasarkan analisis sinyal yang dihasilkan oleh masing-masing gestur tangan menunjukkan bahwa kalman filter berhasil memeperbaiki hasil pengukuran. Ditunjukkan dengan data hasil pengujian 10 kali uji gestur tangan.

Kemudian Berdasarkan hasil uji gestur tangan pada masing-masing gerakan sebanyak 10 kali diperoleh tabel keberhasilan mendeteksi gestur. Batas toleransi adalah 1 derajat diatas sudut acuan.

Tabel 6.19 Keberhasilan mendeteksi gestur

No	Gestur	Jumlah Pengujian	Jumlah Keberhasilan	Sudut Acuan	Presentase keberhasilan(%)
1	Gestur maju	10	10	0° - 90°	100 %
2	Gestur mundur	10	8	0° - 90°	80 %
3	Gestur kiri	10	10	0° - 90°	100 %
4	Gestur kanan	10	9	0° - 90°	90 %
Rata-rata					92.5 %

Berdasarkan Tabel 6.19, tingkat keberhasilan pengujian deteksi gestur tangan oleh perangkat pengendali (TX) sebesar 92.5%. Cukup tinggi untuk kendali gerakan gestur dari 0° hingga 90°. Beberapa faktor kegagalanya diantaranya diakibatkan *human error* (kesalahan mengendalikan), kesalahan sistem (pembacaan sensor) dan kerusakan pada alat. Selain itu terbukti bahwa kalman filter berhasil meminimalisir noise perhitungan oleh sensor dengan menganggap bahwa error 1 derajat pada sistem merupakan bentuk kegagalan perhitungan. Untuk data rinci perolehan keberhasilan terdapat pada Lampiran C.

6.3 Pengujian Navigasi

Pengujian navigasi merupakan pengujian gabungan antara perangkat pengendali dengan robot beroda.

6.3.1 Tujuan Pengujian

Pengujian ini dilakukan untuk mengetahui tingkat akurasi gestur tangan terhadap navigasi pada robot beroda.

6.3.2 Prosedur Pengujian

Adapun prosedur pengujian ketepatan navigasi yang harus dilakukan adalah sebagai berikut.

1. Persiapkan lintasan/sirkuit lurus dengan panjang 5 meter dengan masing-masing jarak permeter terdapat indikator panjang lintasan

2. Permukaan jalur sirkuit yang digunakan harus rata dan tidak kasar atau bergelombang.
3. Siapkan perangkat pengendali kemudian hubungkan dengan PC
4. Menyiapkan perangkat *transmitter* (TX), kemudian hubungkan dengan PC untuk mendeteksi sudut acuan serta memperoleh rekaman grafik.
5. Gunakan arduino IDE untuk *user interface*.
6. Jalankan kode program yang telah dibuat sesuai dengan pengujian
7. Lakukan skenario pengujian sebagai berikut :

a. Tes kecepatan robot beroda maju

- Tes dengan nilai (Sudut = 90°)
- Tes dengan nilai (Sudut = 45°)
- Tes dengan nilai (Sudut = 20°)

b. Tes kecepatan mobil robot mundur

- Tes dengan nilai (Sudut = 90°)
- Tes dengan nilai (Sudut = 45°)
- Tes dengan nilai (Sudut = 20°)

c. Tes kecepatan mobil robot belok atau berputar ke kiri selama 10 detik

- Tes dengan nilai (Sudut = 90°)
- Tes dengan nilai (Sudut = 45°)
- Tes dengan nilai (Sudut = 20°)

d. Tes kecepatan mobil robot belok atau berputar ke kanan selama 10 detik

- Tes dengan nilai (Sudut = 90°)
- Tes dengan nilai (Sudut = 45°)
- Tes dengan nilai (Sudut = 20°)

8. Untuk mendapatkan tingkat akurasi yang baik maka pada pengujian ini dilakukan percobaan sebanyak 5 kali pada skenario pengujian
9. Amati serta dokumentasikan kendali navigasi pada robot beroda.

6.3.3 Pelaksanaan Pengujian

Pengujian ini dilakukan apabila kelengkapan seluruh prosedur pengujian telah terpenuhi. Pengujian ini dilakukan pada ruangan dengan dimensi lebih dari 5x5 meter dengan tujuan agar sirkuit pengujian bisa diterapkan di ruangan tersebut. Pengujian kendali navigasi tidak terbatas hanya gerakan gestur maju,mundur,kiri dan kanan namun, kendali kecepatan robot beroda juga didasarkan pada sudut acuan pada skenario pengujian. Langkah awal pengujian

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

dengan meletakkan robot beroda pada gapura "*Start*" kemudian menguji kecepatan robot beroda sampai garis/gapura "*finish*".

6.3.4 Hasil Pengujian

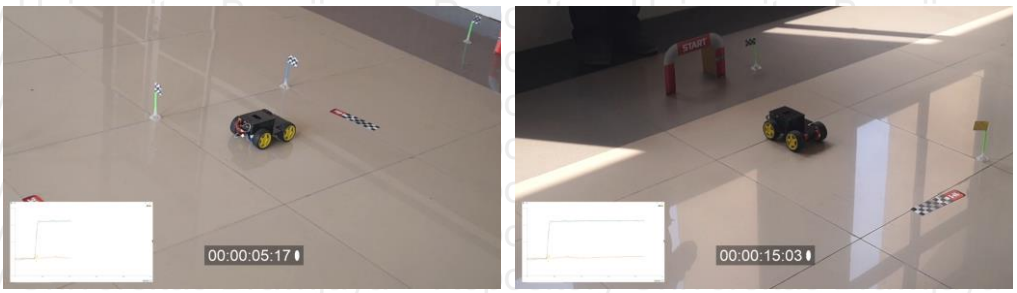
Berdasakan skenario pengujian tersebut diperoleh hasil pengujian sebagai berikut.

1. Tes kecepatan robot beroda maju

Gambar 6.44 Pengujian kecepatan navigasi robot maju

Pada Gambar 6.44 merupakan pengujian kecepatan navigasi pada robot beroda berdasarkan skenario pengujian. Skenario pengujian pada tahap ini adalah tes kecepatan bergerak maju pada robot beroda sesuai dengan *input* ketiga sudut tersebut sehingga akan diperolah tabel pertandingan kecepatan bergerak robot beroda. Sudut yang digunakan untuk mengendalikan navigasi robot maju adalah sudut *pitch*.

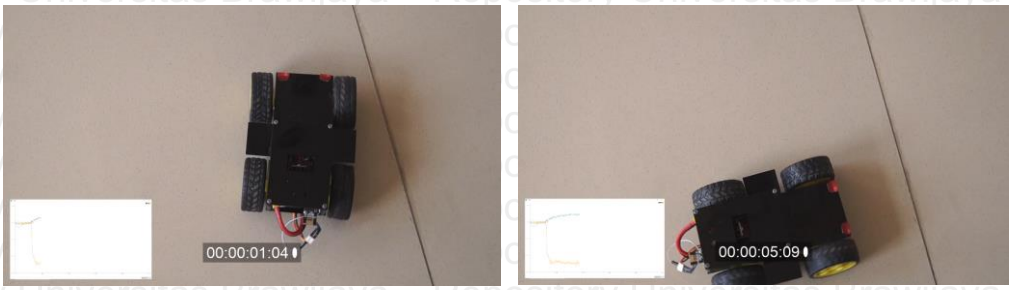
2. Tes kecepatan robot beroda mundur



Gambar 6.45 Pengujian kecepatan navigasi robot mundur

Pada Gambar 6.45 merupakan pengujian kecepatan navigasi pada robot beroda berdasarkan skenario pengujian. Skenario pengujian pada tahap ini adalah tes kecepatan bergerak mundur pada robot beroda sesuai dengan *input* ketiga sudut tersebut sehingga akan diperolah tabel pertandingan kecepatan bergerak robot beroda. Sudut yang digunakan untuk mengendalikan navigasi robot mundur adalah sudut *pitch*.

3. Tes kecepatan robot beroda belok atau berputar ke kiri selama 10 detik



Gambar 6.46 Pengujian kecepatan manuver ke kiri

Pada Gambar 6.46 merupakan pengujian kecepatan navigasi pada robot beroda berdasarkan skenario pengujian. Skenario pengujian pada tahap ini adalah tes kecepatan manuver ke kiri selama 10 detik pada robot beroda sesuai dengan *input* ketiga sudut tersebut sehingga akan diperoleh tabel pertandingan kecepatan bergerak robot beroda. Sudut yang digunakan untuk mengendalikan navigasi robot manuver ke kiri adalah sudut *roll*.

4. Tes kecepatan robot beroda belok atau berputar kekanan selama 10 detik

Gambar 6.47 Pengujian kecepatan manuver ke kanan

Pada Gambar 6.47 merupakan pengujian kecepatan navigasi pada robot beroda berdasarkan skenario pengujian. Skenario pengujian pada tahap ini adalah tes kecepatan manuver ke kanan selama 10 detik pada robot beroda sesuai dengan *input* ketiga sudut tersebut sehingga akan diperoleh tabel pertandingan kecepatan bergerak robot beroda. Sudut yang digunakan untuk mengendalikan navigasi robot manuver ke kanani adalah sudut *roll*.

6.3.5 Analisis Hasil Pengujian

Berdasarkan hasil pengujian tersebut diperoleh tabel perbandingan gerakan robot beroda sesuasi *input* yang dihasilkan dari sudut acuan pengguna serta perhitungan kecepatan dan radian berdasarkan persamaan (6.1) dan (6.2).

$$Kecepatan = \frac{jarak}{waktu} \quad (6.1)$$

$$Rad = derajat \frac{\pi}{180} \quad (6.2)$$

Tabel 6.20 Perbandingan kecepatan navigasi robot maju

No	Sudut Acuan	Panjang Lintasan (m)	Waktu Tempuh percobaan (s)					Kecepatan rata-rata (m/s)
			1	2	3	4	5	
1	90°	5	11.32	11.49	12.01	11.84	12.32	0.42
2	45°	5	14.21	14.90	15.13	14.50	15.04	0.34
3	20°	5	19.14	19.10	19.73	20.24	20.11	0.25

Pada Tabel 6.20, Dengan panjang lintasan 5 meter dengan waktu tempuh dan sudut yang berbeda sesuai dengan *input* maka dilakukan perhitungan untuk memperoleh kecepatan robot beroda dengan persamaan (6.2). Sehingga diperoleh kecepatan robot beroda rata- rata 0.42 m/s pada saat gestur 90°, 0.34 m/s pada saat gestur 45° dan 0.25 m/s pada saat gestur 20°.

Tabel 6.21 Perbandingan kecepatan navigasi robot mundur

No	Sudut Acuan	Panjang Lintasan (m)	Waktu Tempuh percobaan (s)					Kecepatan rata-rata (m/s)
			1	2	3	4	5	
1	90°	5	11.49	11.71	11.92	12.21	12.28	0.42
2	45°	5	15.06	15.01	14.80	14.79	15.23	0.33
3	20°	5	20.11	19.56	20.03	20.56	19.89	0.25

Pada Tabel 6.21, Sama hal-nya dengan perbandingan kecepatan maju, kecepatan mundur juga menguji robot beroda melintasi sirkuit dengan panjang lintasan 5 meter dengan waktu tempuh dan sudut yang berbeda sesuai dengan *input* maka dilakukan perhitungan untuk memperoleh kecepatan robot beroda berdasarkan persamaan (6.1). Sehingga diperoleh kecepatan robot beroda rata-rata 0.42 m/s pada saat gestur 90°, 0.33 m/s pada saat gestur 45° dan 0.25 m/s pada saat gestur 20°.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Tabel 6.22 Perbandingan kecepatan manuver ke kiri

No	Sudut Acuan	Waktu Tempuh (s)	Rata- rata rotasi 5 percobaan (rad)	Kecepatan rotasi (Rad/s)
1	90°	10	250	4.17
2	45°	10	234	3.91
3	20°	10	178	2.97

Pada Tabel 6.22, Waktu tempuh lintasan 5 meter dengan rotasi perputaran manuver robot beroda berdasarkan sudut yang berbeda sesuai dengan *input* maka dilakukan perhitungan untuk memperoleh kecepatan radian per detik dengan persamaan (6.2). Sehingga diperoleh kecepatan rotasi robot beroda 4.17 rad/s pada saat sudut 90°, 3.91 rad/s pada saat sudut 45° dan 2.97 pada saat sudut 20°.

Tabel 6.23 Perbandingan kecepatan manuver ke kanan

No	Sudut Acuan	Waktu Tempuh (s)	Rata- rata rotasi 5 percobaan (rad)	Kecepatan rotasi (Rad/s)
1	90°	10	251	4.18
2	45°	10	231	3.85
3	20°	10	177	2.96

Pada Tabel 6.23, Waktu tempuh lintasan 5 meter dengan rotasi perputaran manuver robot beroda berdasarkan sudut yang berbeda sesuai dengan *input* maka dilakukan perhitungan untuk memperoleh kecepatan radian per detik dengan persamaan (6.2). Sehingga diperoleh kecepatan rotasi robot beroda 4.18 rad/s pada saat sudut 90°, 3.85 rad/s pada saat sudut 45° dan 2.96 pada saat sudut 20°.

Khusus untuk penerapan PWM pada robot beroda, ada point yang tidak kalah penting untuk diperhatikan, bahwa keluaran dari PWM tersebut tidaklah linear. Misalnya motor beroperasi pada 1200 rpm (tanpa beban). Jika diberikan ratio PWM sebasar 100%, maka motor tersebut akan berputar 1200 rpm, namun ketika kita ingin motor berputar pada 600 rpm. Maka kita memberikan ratio PWM sebesar 50%, meskipun pada kenyataannya 600 rpm dapat dicapai ketika ratio PWM mencapai 30%. Hal kedua adalah perhitungan *friction* dan besarnya beban pada motor. Dengan besar ratio PWM yang sama, bisa jadi memberikan kecepatan yang berbeda ketika motor berputar tanpa beban dengan motor yang telah dirakit bersama rangka robot yang tentunya menambah besar massa dan gaya netralnya.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

6.4 Pegujian *Delay* dan *Response time*

6.4.1 Tujuan Pengujian

Pengujian *delay* bertujuan untuk mengetahui waktu yang dibutuhkan oleh masing - masing perangkat untuk saling terhubung. Selanjutnya, tujuan dari *response time* adalah untuk mengetahui seberapa besar tingkat ketanggapan atau kepekaan robot beroda berdasarkan gestur tangan yang dihasilkan pada perangkat kendali atau *gloves*.

6.4.2 Prosedur Pengujian

Adapun prosedur yang perlu diperhatikan dalam melakukan pengujian tersebut, diantaranya adalah :

1. Menyiapkan perangkat pengendali atau *gloves* dan robot beroda
2. Sambungkan perangkat kendali dengan laptop untuk sinkronisasi antara *timer* pada perangkat dengan video saat direkam
3. Nyalakan sakelar pada perangkat robot beroda dan pengendali secara bergantian
4. Lakukan kendali robot beroda beroda berdasarkan gestur tangan maju, mundur,kiri dan kanan sesuai perancangan gestur tangan.
5. Amati dan dokumentasi segala aktifitas pengendalian dan pergerakan yang dilakukan perangkat kendali dan robot beroda.
6. Analisis hasil dokumentasi menggunakan aplikasi *editing* video adobe premiere pro dengan menambahkan *timer* fungsi millis pada kode program arduino yang untuk mendapatkan waktu proses pengendalian. *Timer* diatur dengan format 00:00:00:00 yaitu, jam:menit:detik:milidetik.
7. Manfaatkan *timer* fungsi millis untuk menghitung besar *delay* dan *response time*.
8. Pengujian pada masing-masing bentuk gerakan dilakukan sebanyak lima kali untuk memperoleh hasil *delay* dan *response time* yang akurat.

6.4.3 Pelaksanaan Pengujian

Pengujian ini dilakukan apabila kelengkapan seluruh prosedur pengujian telah terpenuhi. Pengujian ini dilakukan pada ruangan dengan dimensi lebih dari 5x5 meter dengan tujuan agar mobilisasi robot beroda untuk proses pengujian bisa diterapkan di ruangan tersebut. Pengujian kendali navigasi gerakan gestur tangan maju,mundur,kiri dan kanan. Persiapkan alat dokumentasi sebelum melakukan pengujian untuk analisis hasil pengujian *delay* dan *response time*.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

6.4.4 Hasil Pengujian

Hasil yang didapatkan pada saat melakukan pengujian adalah sebagai berikut:

1. Menghitung *Delay* Sistem

(A) (B)

Gambar 6.48 *Delay* sistem

Berdasarkan Gambar 6.48, pengguna mencoba menyalakan sakelar pada perangkat pengendali untuk proses pengendalian robot beroda. Pada gambar bagian (A) menunjukkan *timer* pada saat sakelar dinyalakan dengan waktu 00:00:09:25. Kemudian pada gambar bagian (B) menunjukkan *timer* pada saat sistem secara keseluruhan dapat beroperasi dengan waktu 00:00:11:12 baik itu dalam pergerakan robot beroda, pembacaan gestur tangan oleh sensor, nyala indikator dan lain-lain. Dengan menghitung perbedaan selisih waktu pada gambar (A) dan (B) diperoleh *delay* sebesar 1.87 detik dalam proses menyalakan perangkat kendali hingga terjadi proses.

2. Menghitung *Response time* Gestur Maju

(A) (B)

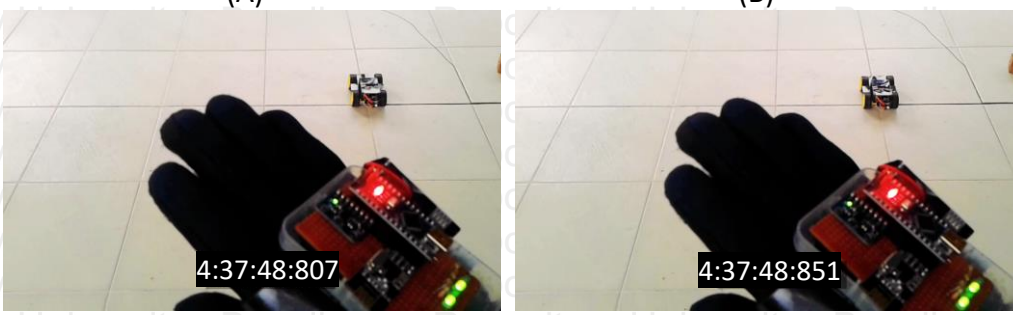
Gambar 6.49 *Response time* gestur maju

Berdasarkan Gambar 6.49, pengguna mengendalikan robot beroda dari kondisi diam kemudian bergerak maju. Gambar bagian (A) menunjukkan *timer* pada saat robot beroda dalam kondisi diam dengan waktu 4:37:47:187. Kemudian pada gambar bagian (B) menunjukkan *timer* pada saat robot beroda

mulai bergerak maju dengan waktu 4:37:47:227. Dengan menghitung perbedaan selisih waktu pada gambar (A) dan (B) diperoleh *response time* sebesar 0.040 detik.

3. Menghitung *Response time* Gestur Mundur

(A) (B)



Gambar 6.50 *Response time* gestur mundur

Berdasarkan Gambar 6.50, pengguna mengendalikan robot beroda dari kondisi diam kemudian bergerak mundur. Gambar bagian (A) menunjukkan *timer* pada saat robot beroda dalam kondisi diam dengan waktu 4:37:48:807. Kemudian pada gambar bagian (B) menunjukkan *timer* pada saat robot beroda mulai bergerak mundur dengan waktu 4:37:48:851. Dengan menghitung perbedaan selisih waktu pada gambar (A) dan (B) diperoleh *response time* sebesar 0.044 detik.

4. Menghitung *Response time* Gestur Kiri

(A) (B)

Gambar 6.51 *Response time* gestur kiri

Berdasarkan Gambar 6.51, pengguna mengendalikan robot beroda dari kondisi diam kemudian bergerak ke kiri. Gambar bagian (A) menunjukkan *timer* pada saat robot beroda dalam kondisi diam dengan waktu 4:37:51:458. Kemudian pada gambar bagian (B) menunjukkan *timer* pada saat robot beroda mulai bergerak ke kiri dengan waktu 4:37:51:505. Dengan menghitung perbedaan selisih waktu pada gambar (A) dan (B) diperoleh *response time* sebesar 0.047 detik.

5. Menghitung *Response time* Gestur Kanan

(A) (B)

Gambar 6.52 *Response time* gestur kanan

Berdasarkan Gambar 6.52, pengguna mengendalikan robot beroda dari kondisi diam kemudian bergerak ke kanan. Gambar bagian (A) menunjukkan *timer* pada saat robot beroda dalam kondisi diam dengan waktu 4:37:52:005. Kemudian pada gambar bagian (B) menunjukkan *timer* pada saat robot beroda mulai bergerak ke kanan dengan waktu 4:37:52:052. Dengan menghitung perbedaan selisih waktu pada gambar (A) dan (B) diperoleh *response time* sebesar 0.047 detik.

6.4.5 Analisis Hasil Pengujian

Berdasarkan hasil pengujian *delay* dan *response time* maka diperoleh nilai pada masing-masing gerakan gestur tangan yang diuji sebanyak lima kali. Kemudian menghitung perbedaan selisih waktu pada saat pengguna melakukan gerakan gestur tangan sampai terjadinya gerakan pada robot beroda dengan persamaan (6.1).

$$\begin{aligned} Response\ time/\ Delay &= (Waktu\ akhir\ (t_2)\ -\ waktu\ awal\ (t_1)) \\ &= t_2\ -\ t_1 \\ &= t_{21} \end{aligned} \quad \textbf{(6.3)}$$

Dengan hasil perhitungan tersebut diperoleh *delay* dan *response time* sebagai berikut.

Tabel 6.24 Hasil uji *delay* hubungan antar perangkat

Percobaan	t_2	t_1	***Delay*** **(detik)**
1	00:00:09:25	00:00:11:12	1.87
2	00:00:37:47	00:00:39:40	1.93
3	00:00:52:51	00:00:54:41	1.90
4	00:01:02:05	00:01:04:89	1.84
5	00:01:24:81	00:01:26:83	2.02
Rata-rata *delay* sistem			1.91

Berdasarkan Tabel 6.24, rata – rata hasil pengujian *delay* pada sistem sebesar 1.91 detik. Menunjukkan bahwa jeda waktu antara proses menyalakan sakelar pada sistem kendali memerlukan waktu sekitar 1.91 detik agar dapat terhubung ke robot beroda. Serta 1.91 detik yang dibutuhkan untuk sistem dapat berjalan seluruhnya.

Tabel 6.25 Hasil uji *response time* pada masing-masing gestur

Gestur tangan	*Response time* pada percobaan (detik)					*Rata-rata* (detik)
	1	2	3	4	5	
Maju	0.040	0.047	0.042	0.045	0.042	0.043
Mundur	0.046	0.044	0.043	0.040	0.046	0.044
Kanan	0.047	0.048	0.042	0.041	0.044	0.044
kiri	0.044	0.041	0.047	0.045	0.046	0.045
Rata-rata *response time* pada seluruh gestur (detik)						0.044

Tabel 6.25 menunjukkan bahwa rata-rata *response time* pada ke-empat gestur tangan sebesar 0.044. Dapat diasumsikan bahwa *response time* sangat cepat sehingga sistem kendali robot beroda menggunakan gestur tangan menjadi sangat akurat, responsif dan optimal. Tingkat kepekaan sensor serta kecepatan transmisi nilai melalui modul NRF24L01 sangat menentukan *response time*.

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan dari penelitian Sistem Kendali Navigasi Robot Beroda dengan Gestur Tangan Menggunakan Metode *Kalman filter* adalah sebagai berikut.

1. Berdasarkan pengujian proses filter yang didapatkan dapat diasumsikan bahwa nilai yang diakuisisi oleh sensor tidak selalu stabil. Banyak faktor yang mempengaruhi sehingga sinyal banyak mengandung *noise* dan ketidakakuratan lainnya. Hasil pengujian menunjukkan penentuan variabel *Q_angle* = 0.001, *Q_bias* = 0.003 dan Nilai *R_measure* = 0.03 dengan metode *try and error* sangat efektif dalam meminimalisir akibat yang ditimbulkan dari *noise*. Namun hal tersebut juga terjadi akibat peran dari pada *complementary filter* untuk proses *sensor fusion* pada *accelerometer* dan *gyroscope* sehingga tidak menghambat kinerja pada *kalman filter*. Hasil pengujian kalman filter menunjukkan bahwa nilai konstanta yang tepat untuk digunakan pada *complementary filter* adalah 0.93. Proses filter keduanya sangat penting untuk terciptanya tingkat akurasi yang tinggi terhadap sinyal yang dihasilkan. Keakuratan dari sinyal juga akan berdapak terhadap gerakan yang dihasilkan oleh robot beroda.
2. Berdasarkan pengujian gestur tangan dapat disimpulkan bahwa implementasi kalman filter terhadap hasil uji gestur tangan pengguna mampu untuk untuk meminimalisir atau meredam *noise* pada sudut pitch dan roll pada saat melakukan gerakan gestur tangan maju, mundur, kiri dan kanan. Hal itu ditunjukkan dengan menganalisis sinyal hasil uji gestur. Hasil pengujian menunjukkan bahwa tingkat akurasi ketepatan gerakan gestur tangan terhadap pergerakan dari robot beroda sesuai gerakan navigasi yang telah ditentukan sebesar 92.5%. Sehingga dapat diasumsikan bahwa perangkat *gloves* memiliki akurasi yang tinggi pada saat mengendalikan navigasi robot beroda.
3. Robot beroda berdasarkan pengujian mampu bergerak sesuai dengan gestur tangan manusia. Pengujian navigasi menjelaskan secara rinci mengenai kendali kecepatan pada robot beroda berdasarkan *input* sudut 20°, 45° dan 90° pada perangkat pengendali. Hasil pengujian menunjukkan bahwa kendali kecepatan navigasi robot beroda berdasarkan *input* sudut berhasil di implementasikan serta menunjukkan kecepatan pada masing-masing robot beroda berdasarkan gestur tangan dan sudut yang berbeda.
4. Pengujian *delay* dan *response time* menunjukkan bahwa delay pada saat sistem dinyalakan sampai dengan robot beroda mengalami pergerakan adalah sebesar 1.91 detik. Sementara untuk *response time* yang dibutuhkan perangkat *gloves* untuk mengendalikan robot beroda adalah sebesar 0.044 detik. Dapat diasumsikan bahwa *response time* sangat cepat sehingga sistem kendali robot beroda menggunakan gestur tangan menjadi sangat akurat, responsif dan optimal.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

Hasil penelitian dapat menunjukan bahwa sebuah sistem pengontrolan navigasi robot berdasarkan gerakan tangan dapat diimplementasikan dan diaplikasikan secara riil, sehingga dapat memberikan pengalaman baru dalam berinteraksi dengan robot.

7.2 Saran

Saran untuk pengembangan Sistem Kendali Navigasi Robot Beroda dengan Gestur Tangan Menggunakan Metode Kalman filter terhadap penelitian selanjutnya adalah sebagai berikut.

1. Penerapan komunikasi dua arah pada sistem kendali dan robot beroda sehingga dapat memantau sinyal yang dikirimkan oleh perangkat komunikasi NRF24L01.
2. Penerapan fitur serta algoritma sistem kendali lainnya untuk mengoptimalkan kinerja sistem seperti *PID (Proportional–Integral–Derivative controller).*
3. Penerapan strategi dalam hal manajemen sumber daya baterai.

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

DAFTAR PUSTAKA

Al Khatib, E. I., Jaradat, M. A., Hafez, M. A. & Roigari, M., 2015. Multiple sensor fusion for mobile robot localization and navigation using the Extended Kalman Filter. *10th International Symposium on Mechatronics and its Applications (ISMA),* 8-10 December.p. 1.

ALSrobot, 2016. *www.alsrobot.com.* [Online] Tersedia di: <http://www.alsrobot.com/index.php?route=product/product&product_id=535> [*Acce*ssed 1 9 2018].

Article, D. E., 2008. *AN-0193 Dasar Robotika 1 – Motor DC.* [Online] Tersedia di: <http://delta-electronic.com/article/> [*Acce*ssed 1 9 2018].

Assunta, P., Ottaviano, E. & Pierluigi, R., 2015. *Handbook of Research on Emerging Digital Tools for Architectural Surveying, Modeling, and Representation.* Lazio(Lazio): s.n.

Bhuyan, A. I. & Mallick, T. C., 2014. *Gyro-Acce*lerometer based control of a robotic Arm. *The 9th International Forum on Strate,* Issue *Gyro-Acce*lerometer based control of a robotic Arm, p. 2.

Bozinovski, S., 1994. *Parallel programming for mobile robot control.* Poznan: s.n.

Christina, M., Javier, V., Ramon, M. & Fransisco, P. J., 2015. Real-Time Hand Tracking and Gesture Recognition for Human-Computer Interaction. 5(3).

Components101, 2018. *Arduino Nano.* [Online] Tersedia di: <https://components101.com/microcontrollers/arduino-nano> [*Acce*ssed 5 4 2019].

Dejan, 2017. *Howtomechatronics.* [Online] Tersedia di: <https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/> [*Acce*ssed 2 4 2019].

Electronics, 2016. *Electronics.* [Online] Tersedia di: <https://microcontrollerelectronics.com/fs1000a-wireless-rf433-transmit-and-receive-module-pair/> [*Acce*ssed 10 12 2018].

Electronics, L. L., 2012. *Introduction to Encoding and Decoding. Making a simple Encoder-Decoder device.* [Online] Tersedia di: <http://letslearnelectronics.blogspot.com/2012/07/introduction-to-encoding-and-decoding_1610.html> [*Acce*ssed 1 9 2018].

ElectronicWings, 2018. *Sensors & Modules.* [Online] Tersedia di: <https://www.electronicwings.com/sensors-modules/mpu6050-*gyro*scope-*acce*lerometer-temperature-sensor-module> [*Acce*ssed 2 4 2019].

Glass, n.d. *Glass.* [Online] Tersedia di: <https://x.company/glass/> [*Acce*ssed 17 5 2019].

Grace, G. R., Dika, S. & Demi, S., 2015. Embedded sensor fusion and moving-average filter for Inertial Measurement Unit (IMU) on the microcontroller-based stabilized platform. *International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT),* 29-30 October.p. 1.

Hasan, U. Z., Asif, A. J., Khan, M. A. & Safwan, T. F., 2018. A simple and effective way of controlling a robot by hand gesture. *International Conference on Intelligent Computing and Control Systems (ICICCS),* 11 January.p. 1.

Instructable, 2016. *L293D MOTOR DRIVER.* [Online] Tersedia di: <https://www.instructables.com/id/L293D-Motor-Driver/> [*Acc*essed 1 9 2018].

James, D., 2008. *Analog Devices. Accelerometer and.* [Online] Tersedia di: <https://elpuig.xeill.net/Members/vcarceler/articulos/jugando-con-elwiimote-y-gnu-linux/sensor971.pdf/at_download/file> [*Acc*essed 1 9 2018].

Lab, I., 2018. *Immersia Lab.* [Online] Tersedia di: <http://www.immersa-lab.com/pengertian-*gyro*scope-dan-cara-kerjanya.htm> [*Acc*essed 20 4 2019].

Luo, Y., Tian, X. & Tan, J., 2013. Research on Vehicle Attitude Determination. *IEEE,* p. 14.

Martin, R., 2017. *Thesis Project : Integrated Sensor Orientation on Micro Aerial Vehicles.* Laussane: Federal Institute Of Technology In Laussane.

Matari´c, M. J., 2007. *The Robotics Primer.* London: The MIT Press.

Matari´c, M. J., 2017. *The Robotics Primer.* London, England: Massachusetts Institute of Technology.

Mathwork, 2018. *Estimating Position of an Aircraft using Kalman Filter.* [Online] Tersedia di: <https://www.mathworks.com/help/dsp/examples/estimating-position-of-an-aircraft-using-kalman-filter.html> [*Acc*essed 1 9 2018].

Mohinder, Andrews, A. & Grewal, 2001. *Kalman filtering: theory and practice using matlab.* s.l.:John Wiley & Sons ,Inc.

Mojeebi, T. & Tulo, S. K., 2016. *Acce*lerometer Gesture Controlled Robot. . *An International Journal of,* Volume 3, p. 3.

Paul, Z. & Howard, M., 2000. *Fundamentals of Kalman Filtering: A Practical Approach.* America: American Institute of Aeronautics and Astronautics, Incorporated.

Prasetio, B. H. & Kurniawan, W., 2015. Uji Performansi Ensemble Kalman Filter untuk Mengurangi Noise Pengukuran Sensor pada Robot. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* 11 Juni, Volume 2, p. 1.

Pro, M., 2018. *How to Interface Arduino and the MPU 6050 Sensor.* [Online] Tersedia di: <https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor> [*Acce*ssed 1 9 2018].

Qiang, L., Ranyang, L., Kaifan, J. & Wei, D., 2015. Kalman Filter and Its Application. *International Conference on Intelligent Networks and Intelligent Systems,* 1-3 November, Volume 8, p. 1.

Qiang, L., Ranyang, L., Kaifan, J. & Wei, D., 2015. Kalman Filter and Its Application. *International Conference on Intelligent Networks and Intelligent Systems (ICINIS),* 1-3 November.p. 1.

R., M. & G., 2014. Hand Gesture Based Mobile Robot Control Using PIC Microcontroller. *International Conference on Circuit, Power and Computing Technologies [ICCPCT],* Issue Hand Gesture Based Mobile Robot Control, p. 2.

S., R. P. & B. A., 2016. Sistem Pengenalan Gestur Tangan Untuk Kendali Gerak Mobile Robot Berbasis Pengolan Citra. Volume 4.

Takeshi, K., Takeshi, O., Masakatsu, K. & Katsuhiko, S., 2018. The hand mouse: GMM hand-color classification and mean shift tracking. 6(2).

Techopedia, 2016. *Wearable Device.* [Online] Tersedia di: <https://www.techopedia.com/definition/31206/*wearable-device*> [*Acce*ssed 6 Juni 2019].

Zaman, H. U., Joy, A. A. & Akash, K. M., 2017. A Simple and Effective Way of Controlling a Robot by Hand Gesture. *International Conference on Intelligent Computing and Control Systems ,* p. 3.

Repository Universitas Brawijaya

LAMPIRAN A KODE PROGRAM

A.1 Kode program I2c

Kode program : konfigurasi komunikasi dengan protokol I2c

```
1  const uint8_t IMUAddress = 0x68;
2  const uint16_t I2C_TIMEOUT = 1000;
3  uint8_t i2cWrite(uint8_t registerAddress, uint8_t data, bool
4  sendStop) {
5    return i2cWrite(registerAddress, &data, 1, sendStop);
6  }
7
8  uint8_t i2cWrite(uint8_t registerAddress, uint8_t *data,
9  uint8_t length, bool sendStop) {
10   Wire.beginTransmission(IMUAddress);
11   Wire.write(registerAddress);
12   Wire.write(data, length);
13   uint8_t rcode = Wire.endTransmission(sendStop);
14   if (rcode) {
15     Serial.print(F("i2cWrite failed: "));
16     Serial.println(rcode);
17   }
18   return rcode;
19 }
20
21 uint8_t i2cRead(uint8_t registerAddress, uint8_t *data,
22 uint8_t nbytes) {
24   uint32_t timeOutTimer;
25   Wire.beginTransmission(IMUAddress);
26   Wire.write(registerAddress);
27   uint8_t rcode = Wire.endTransmission(false); // Don't
28 release the bus
29   if (rcode) {
30     Serial.print(F("i2cRead failed: "));
31     Serial.println(rcode);
32     return rcode;
33   }
34   Wire.requestFrom(IMUAddress, nbytes, (uint8_t)true); //
35 Send a repeated start and then release the bus after reading
36   for (uint8_t i = 0; i < nbytes; i++) {
37     if (Wire.available())
38       data[i] = Wire.read();
39     else {
40       timeOutTimer = micros();
41       while (((micros() - timeOutTimer) < I2C_TIMEOUT) &&
42 !Wire.available());
43       if (Wire.available())
44         data[i] = Wire.read();
45       else {
46         Serial.println(F("i2cRead timeout"));
47         return 5; // This error value is not already taken
48 by endTransmission
49       }
50     }
51   }
52   return 0; // Success
53 }
```

A.2 Kode program *gloves* (*Transmitter*)

Kode program : program *transmitter*

```
1  #include <SPI.h>
2  #include <Mirf.h>
3  #include <nRF24L01.h>
4  #include <MirfHardwareSpiDriver.h>
5  #include <Wire.h>
6  #include <Kalman.h>
7  #define RESTRICT_PITCH
8
9  Kalman kalmanX;
10 Kalman kalmanY;
11 int myLed[] = {A3,A1,A0,A2};
12 double accX, accY, accZ;
13 double gyroX, gyroY, gyroZ;
14 int16_t tempRaw;
15 double gyroXangle, gyroYangle;
16 double compAngleX, compAngleY;
17 double kalAngleX, kalAngleY;
18 uint32_t timer;
19 uint8_t i2cData[14];
20 char data[32];
21
22 void setup(){
23   Serial.begin(9600);
24
25   for(int i=0; i<=3; i++)
26   {
27     pinMode(myLed[i], OUTPUT);
28     digitalWrite(myLed[i], HIGH);
29   }
30
31   Mirf.spi = &MirfHardwareSpi;
32   Mirf.init();
33   Mirf.setRADDR((byte *)"clie1");
34   Mirf.payload = 32;
35   Mirf.channel = 11;
36   Mirf.config();
37   Wire.begin();
38   #if ARDUINO >= 157
39     Wire.setClock(400000UL); // Set I2C frequency to 400kHz
40   #else
41     TWBR = ((F_CPU / 400000UL) - 16) / 2; // Set I2C
42 frequency to 400kHz
43   #endif
44
45   i2cData[0] = 7; // Set the sample rate to 1000Hz -
46 8kHz/(7+1) = 1000Hz
47   i2cData[1] = 0x00; // Disable FSYNC and set 260 Hz Acc
48 filtering, 256 Hz Gyro filtering, 8 KHz sampling
49   i2cData[2] = 0x00; // Set Gyro Full Scale Range to
50 ±250deg/s
51   i2cData[3] = 0x00; // Set Accelerometer Full Scale Range
52 to ±2g
53
```


Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

```
  while (i2cWrite(0x19, i2cData, 4, false)); // Write to
all four registers at once
  while (i2cWrite(0x6B, 0x01, true)); // PLL with X axis
gyroscope reference and disable sleep mode
  while (i2cRead(0x75, i2cData, 1));
  if (i2cData[0] != 0x68) { // Read "WHO_AM_I" register
    Serial.print(F("Error reading sensor"));
    while (1);
  }

  delay(100); // Wait for sensor to stabilize

  /* Set kalman and gyro starting angle */
  while (i2cRead(0x3B, i2cData, 6));
  accX = (int16_t)((i2cData[0] << 8) | i2cData[1]);
  accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
  accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);

  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ *
accZ)) * RAD_TO_DEG;

  kalmanX.setAngle(roll);
  kalmanY.setAngle(pitch);
  gyroXangle = roll;
  gyroYangle = pitch;
  compAngleX = roll;
  compAngleY = pitch;

  timer = micros();

  Serial.println("Beginning ... ");
}

void loop(){
  while (i2cRead(0x3B, i2cData, 14));
  accX = (int16_t)((i2cData[0] << 8) | i2cData[1]); // 8
bit merupakan banyaknya data yang dikirim per detak,
dikarenakan arduino hanya mampum menampung data 8 bit
  accY = (int16_t)((i2cData[2] << 8) | i2cData[3]);
  accZ = (int16_t)((i2cData[4] << 8) | i2cData[5]);
  tempRaw = (int16_t)((i2cData[6] << 8) | i2cData[7]);
  gyroX = (int16_t)((i2cData[8] << 8) | i2cData[9]);
  gyroY = (int16_t)((i2cData[10] << 8) | i2cData[11]);
  gyroZ = (int16_t)((i2cData[12] << 8) | i2cData[13]);;

  double dt = (double)(micros() - timer) / 1000000; //
Calculate delta time
  timer = micros();

  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch = atan(-accX / sqrt(accY * accY + accZ *
accZ)) * RAD_TO_DEG;

  double gyroXrate = gyroX / 131.0; // Convert to deg/s
  double gyroYrate = gyroY / 131.0; // Convert to deg/s

  if ((roll < -90 && kalAngleX > 90) || (roll > 90 &&
kalAngleX < -90)) {
```

```
113     kalmanX.setAngle(roll);
114     compAngleX = roll;
115     kalAngleX = roll;
116     gyroXangle = roll;
117   } else
118     kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt); //
119 Calculate the angle using a Kalman filter
120
121   if (abs(kalAngleX) > 90)
122     gyroYrate = -gyroYrate; // Invert rate, so it fits the
123 restriced accelerometer reading
124   kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
125
126   gyroXangle += gyroXrate * dt; // Calculate gyro angle
127 without any filter
128   gyroYangle += gyroYrate * dt;
129
130   compAngleX = 0.95 * (compAngleX + gyroXrate * dt) + 0.05
131 * roll; // Calculate the angle using a Complimentary filter
132   compAngleY = 0.95 * (compAngleY + gyroYrate * dt) + 0.05
133 * pitch;
134
135   // Reset the gyro angle when it has drifted too much
136   if (gyroXangle < -180 || gyroXangle > 180)
137     gyroXangle = kalAngleX;
138   if (gyroYangle < -180 || gyroYangle > 180)
139     gyroYangle = kalAngleY;
140
141   int defaultPwm = 80;
142   int pwmRef;
143
144   int pwmKiri = 0;
145   int pwmKanan = 0;
146
147   if(kalAngleX >= 0)
148   {
149      pwmRef = int(kalAngleX) + defaultPwm;
150      pwmKiri = pwmRef - int(kalAngleY);
151      pwmKanan = pwmRef + int(kalAngleY);
152
153     if(pwmKiri < 0 ){ pwmKiri = 0; }
154     if(pwmKiri > 255) { pwmKiri = 255; }
155
156     if(pwmKanan < 0 ){ pwmKanan = 0; }
157     if(pwmKanan > 255) { pwmKanan = 255; }
158
159     digitalWrite(myLed[0], LOW);
160     digitalWrite(myLed[3], HIGH);
161     if(pwmKanan > pwmKiri)
162     {
163       digitalWrite(myLed[1], LOW);
164       digitalWrite(myLed[2], HIGH);
165     }
166     else if(pwmKanan < pwmKiri)
167     {
168       digitalWrite(myLed[1], HIGH);
169       digitalWrite(myLed[2], LOW);
170     }
171     else
```

```
172       {
173         digitalWrite(myLed[1], HIGH);
174         digitalWrite(myLed[2], HIGH);
175       }
176
177     }
178     if(kalAngleX < 0)
179     {
180       pwmRef = int(kalAngleX) - defaultPwm;
181       pwmKiri = pwmRef + int(kalAngleY);
182       pwmKanan = pwmRef - int(kalAngleY);
183
184       if(pwmKiri > 0 ){ pwmKiri = 0; }
185       if(pwmKiri < -255) { pwmKiri = -255; }
186
187       if(pwmKanan > 0 ){ pwmKanan = 0; }
188       if(pwmKanan < -255) { pwmKanan = -255; }
189
190       digitalWrite(myLed[0], HIGH);
191       digitalWrite(myLed[3], LOW);
192       if(pwmKanan > pwmKiri)
193       {
194         digitalWrite(myLed[1], HIGH);
195         digitalWrite(myLed[2], LOW);
196       }
197       else if(pwmKanan < pwmKiri)
198       {
199         digitalWrite(myLed[1], LOW);
200         digitalWrite(myLed[2], HIGH);
201       }
202       else
203       {
204         digitalWrite(myLed[1], HIGH);
205         digitalWrite(myLed[2], HIGH);
206       }
207     }
208
209     Serial.print(kalAngleX); Serial.print("\t");
210     Serial.print(kalAngleY); Serial.print("\t");
211
212     Serial.print(pwmRef); Serial.print("\t");
213     Serial.print(pwmKiri); Serial.print("\t");
214     Serial.print(pwmKanan); Serial.print("\t");
215
216
217   //Edit
218   //  Serial.print(roll); Serial.print("\t");
219   //  Serial.print(pitch); Serial.print("\t");
220
221
222     Serial.print("\r\n");
223
224     sprintf(data,"%3d,%3d",pwmKiri, pwmKanan);
225
226     Mirf.setTADDR((byte *)"serv1");
227
228     Mirf.send((byte *)&data);
229
230     while(Mirf.isSending()){
```

```
231     }
232
233     delay(2);
234   }
```

A.3 Kode program robot beroda (*Receiver*)

Kode program : *receiver* (TX) Robot beroda

```
1   #include <SPI.h>
2   #include <Mirf.h>
3   #include <nRF24L01.h>
4   #include <MirfHardwareSpiDriver.h>
5
6   #define pwmKanan  9
7   #define in1       8
8   #define in2       7
9   #define in3       3
10  #define in4       2
11  #define pwmKiri   6
12
13  char data[32];
14  String valSplit[5];
15
16  void setup(){
17    Serial.begin(9600);
18
19    pinMode(in1, OUTPUT);
20    pinMode(in2, OUTPUT);
21    pinMode(in3, OUTPUT);
22    pinMode(in4, OUTPUT);
23
24    Mirf.spi = &MirfHardwareSpi;
25    Mirf.init();
26    Mirf.setRADDR((byte *)"serv1");
27    Mirf.payload = 32;
28    Mirf.channel = 11;
29    Mirf.config();
30    Serial.println("Listening...");
31  }
32
33  void loop(){
34
35    if(!Mirf.isSending() && Mirf.dataReady())
36    {
37
38      Mirf.getData((byte *)&data);
39      parseData(String(data), ",");
40
41      if(valSplit[0].toInt() > 0 && valSplit[1].toInt() > 0)
42      {
43        digitalWrite(in1, LOW);
44        digitalWrite(in2, HIGH);
45        digitalWrite(in3, LOW);
46        digitalWrite(in4, HIGH);
47        analogWrite(pwmKiri, valSplit[0].toInt());
48        analogWrite(pwmKanan, valSplit[1].toInt());
49
50      }
```

```
51      if(valSplit[0].toInt() < 0 && valSplit[1].toInt() < 0)
52      {
53        digitalWrite(in1, HIGH);
54        digitalWrite(in2, LOW);
55        digitalWrite(in3, HIGH);
56        digitalWrite(in4, LOW);
57        analogWrite(pwmKiri, abs(valSplit[0].toInt()));
58        analogWrite(pwmKanan, abs(valSplit[1].toInt()));
59      }
60      if(valSplit[0].toInt() == 0 && valSplit[1].toInt() == 0)
61      {
62        digitalWrite(in1, HIGH);
63        digitalWrite(in2, LOW);
64        digitalWrite(in3, HIGH);
65        digitalWrite(in4, LOW);
66        analogWrite(pwmKiri, 0);
67        analogWrite(pwmKanan, 0);
68      }
69
70      Serial.print(valSplit[0].toInt()); Serial.print("\t");
71      Serial.print(valSplit[1].toInt()); Serial.print("\t");
72      Serial.println();
73      //Serial.println(data);
74    }
75  }
76
77  void parseData(String text, String key)
78  {
79    int countSplitSecond=0;
80    int lastIndexSecond=0;
81    text += key;
82
83    //Serial.println(text);
84    for(int j = 0; j < text.length(); j++)
85    {
86      if(text.substring(j, j+1) == key)
87      {
88        valSplit[countSplitSecond] =
89  text.substring(lastIndexSecond,j);
90        lastIndexSecond = j + 1;
91        //Serial.print(countSplitSecond);
92        //Serial.print(":");
93        //Serial.println(valSplit[countSplitSecond]);
94        countSplitSecond++;
95      }
96    }
97
98  }
```

Repository Universitas Brawijaya

LAMPIRAN B AKUISISI DATA UJI *KALMAN FILTER*

B.1 Akuisisi data skenario 1

No	*Computer Time*	*Pitch_Kalman*	*Roll_Kalman*	*Pitch*	*Roll*
1	3:58:40	0.58	0.08	0.71	0.71
2	3:58:40	0.12	-0.15	0.63	0.26
3	3:58:40	-0.35	-0.36	0.41	0.52
4	3:58:40	-0.78	-0.52	0.56	0.59
5	3:58:40	-1.16	-0.65	0.55	0.46
6	3:58:40	-1.50	-0.77	0.67	0.37
7	3:58:40	-1.66	-0.89	1.34	0.05
8	3:58:40	-1.81	-0.97	0.22	0.47
9	3:58:41	-1.87	-0.9	0.39	0.52
10	3:58:41	-1.74	-0.8	0.66	0.52
11	3:58:41	-1.61	-0.68	0.38	0.36
12	3:58:41	-1.46	-0.57	0.14	0.28
13	3:58:42	-1.26	-0.43	0.14	0.13
14	3:58:42	-1.01	-0.21	0.22	0.63
15	3:58:42	-0.70	-0.04	0.8	0.58
16	3:58:42	-0.49	0.13	0.31	0.63
17	3:58:42	-0.28	0.24	0.42	0.31
18	3:58:43	-0.09	0.42	0.44	0.78
19	3:58:43	0.05	0.52	0.31	0.5
20	3:58:43	0.21	0.54	0.7	0.31
21	3:58:43	0.39	0.63	0.79	0.77
22	3:58:43	0.51	0.6	0.52	0.33
23	3:58:44	0.57	0.68	0.52	0.72
24	3:58:44	0.65	0.62	0.48	0.28
25	3:58:44	0.66	0.64	0.42	0.45
26	3:58:44	0.72	0.66	0.65	0.59
27	3:58:44	0.78	0.64	0.64	0.44
28	3:58:45	0.73	0.49	0.39	0.03
29	3:58:45	0.71	0.58	0.58	0.89
30	3:58:45	0.73	0.56	0.5	0.45
31	3:58:45	0.77	0.46	0.66	0.08
32	3:58:45	0.74	0.46	0.67	0.36
33	3:58:46	0.73	0.49	0.56	0.54
34	3:58:46	0.67	0.48	0.42	0.44
35	3:58:46	0.61	0.4	0.31	0.15
36	3:58:46	0.54	0.33	0.19	0.11
37	3:58:47	0.49	0.34	0.33	0.51
38	3:58:47	0.50	0.38	0.48	0.66
39	3:58:47	0.45	0.38	0.25	0.43
40	3:58:47	0.53	0.51	0.97	0.92
41	3:58:47	0.43	0.5	0.08	0.63
42	3:58:48	0.48	0.48	0.62	0.03
43	3:58:48	0.51	0.48	0.67	0.53
44	3:58:48	0.48	0.59	0.42	0.86

45	3:58:48	0.63	0.44	0.46	0.13
46	3:58:48	0.62	0.47	0.45	0.43
47	3:58:49	0.62	0.45	0.11	0.19
48	3:58:49	0.62	0.41	0.33	-0.08
49	3:58:49	0.66	0.44	0.19	0.16
50	3:58:49	0.60	0.38	0.72	0.25
51	3:58:49	0.56	0.41	0.65	0.56
52	3:58:50	0.56	0.44	0.52	0.74
53	3:58:50	0.61	0.47	1.01	0.54
54	3:58:50	0.61	0.45	0.62	0.61
55	3:58:50	0.64	0.41	0.71	0.61
56	3:58:50	0.68	0.44	0.75	0.28
57	3:58:51	0.62	0.43	0.47	0.26
58	3:58:51	0.62	0.41	0.48	0.44
59	3:58:51	0.68	0.31	0.65	0.05
60	3:58:51	0.63	0.44	0.53	0.7
61	3:58:51	0.62	0.46	0.41	0.44
62	3:58:52	0.62	0.44	0.66	0.39
63	3:58:52	0.62	0.47	0.53	0.53
64	3:58:52	0.66	0.45	0.75	0.41
65	3:58:52	0.60	0.41	0.39	0.25
66	3:58:52	0.56	0.44	0.47	0.5
67	3:58:53	0.56	0.38	0.45	0.23
68	3:58:53	0.68	0.41	1.12	0.51
69	3:58:53	0.79	0.51	1.11	0.68
70	3:58:53	0.74	0.48	0.33	0.4
71	3:58:54	0.57	0.47	0.13	0.39
72	3:58:54	0.53	0.45	0.4	0.42
73	3:58:54	0.52	0.31	0.62	-0.11
74	3:58:54	0.53	0.22	0.59	-0.03
75	3:58:54	0.50	0.43	0.45	1.14
76	3:58:55	0.51	0.34	0.53	0.06
77	3:58:55	0.49	0.48	0.47	0.95
78	3:58:55	0.49	0.53	0.57	0.66
79	3:58:55	0.56	0.44	0.86	0.05
80	3:58:55	0.50	0.42	0.3	0.41
81	3:58:56	0.43	0.36	0.2	0.13
82	3:58:56	0.32	0.32	-0.05	0.24
83	3:58:56	0.31	0.4	0.35	0.6
84	3:58:56	0.38	0.38	0.59	0.34
85	3:58:56	0.33	0.34	0.3	0.28
86	3:58:57	0.37	0.43	0.56	0.63
87	3:58:57	0.42	0.43	0.6	0.45
88	3:58:57	0.40	0.4	0.45	0.45
89	3:58:57	0.41	0.48	0.53	0.66
90	3:58:57	0.50	0.43	0.78	0.28
91	3:58:58	0.45	0.42	0.29	0.45
92	3:58:58	0.49	0.43	0.64	0.42
93	3:58:58	0.48	0.49	0.36	0.64
94	3:58:58	0.43	0.57	0.36	0.65

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

95	3:58:58	0.43	0.56	0.25	0.37
96	3:58:59	0.53	0.47	0.87	0.19
97	3:58:59	0.52	0.48	0.61	0.62
98	3:58:59	0.53	0.52	0.47	0.5
99	3:58:59	0.48	0.47	0.3	0.31
100	3:59:00	0.56	0.49	0.85	0.55

B.2 Akuisisi data skenario 2

No	*Computer Time*	*Pitch_Kalman*	*Roll_Kalman*	*Pitch*	*Roll*
1	4:03:26	0.43	0.08	0.16	0.25
2	4:03:26	-0.02	-0.14	0.7	0.58
3	4:03:26	-0.45	-0.31	0.55	0.38
4	4:03:26	-0.83	-0.47	0.52	-0.03
5	4:03:26	-1.20	-0.6	0.47	0.53
6	4:03:27	-1.51	-0.69	0.39	0.79
7	4:03:27	-1.77	-0.76	0.14	0.52
8	4:03:27	-1.90	-0.78	0.5	0.55
9	4:03:27	-1.89	-0.73	0.91	0.52
10	4:03:28	-1.77	-0.72	0.7	0.03
11	4:03:28	-1.59	-0.59	0.66	0.45
12	4:03:28	-1.30	-0.4	0.68	0.75
13	4:03:28	-1.03	-0.33	0.59	0.12
14	4:03:28	-0.76	-0.19	0.63	0.4
15	4:03:29	-0.57	-0.13	0.09	0.02
16	4:03:29	-0.39	0.02	0.36	0.41
17	4:03:29	-0.19	0.18	0.55	0.66
18	4:03:29	-0.05	0.13	0.19	-0.14
19	4:03:29	0.05	0.21	0.25	0.31
20	4:03:30	0.19	0.29	0.59	0.45
21	4:03:30	0.33	0.32	0.34	0.31
22	4:03:30	0.42	0.33	0.42	0.28
23	4:03:30	0.49	0.45	0.37	0.66
24	4:03:30	0.48	0.47	0.11	0.36
25	4:03:31	0.54	0.44	0.67	0.22
26	4:03:31	0.53	0.5	0.19	0.62
27	4:03:31	0.59	0.48	0.56	0.39
28	4:03:31	0.52	0.49	0.25	0.37
29	4:03:31	0.55	0.6	0.39	0.8
30	4:03:32	0.57	0.54	0.55	0.28
31	4:03:32	0.51	0.53	0.33	0.47
32	4:03:32	0.52	0.43	0.39	-0.02
33	4:03:32	0.52	0.4	0.5	0.33
34	4:03:32	0.47	0.45	0.25	0.66
35	4:03:33	0.50	0.36	0.47	0.13
36	4:03:33	0.59	0.34	0.8	0.23
37	4:03:33	0.57	0.36	0.4	0.42
38	4:03:33	0.60	0.31	0.58	0.03
39	4:03:34	0.54	0.37	0.39	0.66

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

40	4:03:34	0.60	0.39	0.72	0.42
41	4:03:34	0.60	0.4	0.55	0.5
42	4:03:34	0.57	0.35	0.58	0.13
43	4:03:34	0.60	0.42	0.68	0.61
44	4:03:35	0.60	0.29	0.51	-0.22
45	4:03:35	0.61	0.33	0.62	0.52
46	4:03:35	0.60	0.29	0.45	0.2
47	4:03:35	0.55	0.25	0.39	0.05
48	4:03:35	0.68	0.29	1.16	0.39
49	4:03:36	0.66	0.21	0.5	-0.02
50	4:03:36	0.63	0.13	0.45	0.14
51	4:03:36	0.56	0.25	0.34	0.73
52	4:03:36	0.58	0.09	0.61	0.1
53	4:03:36	0.61	0.12	0.76	0.31
54	4:03:37	0.56	0.2	0.34	0.5
55	4:03:37	0.49	0.35	0.27	0.85
56	4:03:37	0.51	0.3	0.63	0.2
57	4:03:37	0.66	0.33	1.12	0.46
58	4:03:37	0.68	0.33	0.66	0.36
59	4:03:38	0.62	0.26	0.39	-0.03
60	4:03:38	0.63	0.26	0.67	0.28
61	4:03:38	0.60	0.26	0.5	0.17
62	4:03:38	0.65	0.38	0.89	0.72
63	4:03:38	0.63	0.41	0.38	0.41
64	4:03:39	0.52	0.41	0.17	0.42
65	4:03:39	0.52	0.33	0.66	0.12
66	4:03:39	0.60	0.4	0.72	0.54
67	4:03:39	0.53	0.39	0.38	0.33
68	4:03:40	0.52	0.53	0.55	0.83
69	4:03:40	0.51	0.52	0.36	0.39
70	4:03:40	0.47	0.46	0.47	0.33
71	4:03:40	0.51	0.44	0.75	0.31
72	4:03:40	0.45	0.4	0.22	0.25
73	4:03:41	0.36	0.33	0.17	0.08
74	4:03:41	0.36	0.35	0.42	0.36
75	4:03:41	0.32	0.31	0.11	0.23
76	4:03:41	0.31	0.32	0.53	0.42
77	4:03:41	0.34	0.28	0.5	0.11
78	4:03:42	0.45	0.19	0.83	-0.14
79	4:03:42	0.48	0.19	0.61	0.38
80	4:03:42	0.41	0.14	0.03	-0.03
81	4:03:42	0.35	0.06	0.2	-0.12
82	4:03:42	0.39	0.07	0.54	0.34
83	4:03:43	0.46	0.06	0.7	0.13
84	4:03:43	0.53	0.08	0.87	0.26
85	4:03:43	0.60	0.2	0.71	0.68
86	4:03:43	0.54	0.28	0.31	0.39
87	4:03:43	0.55	0.11	0.66	-0.48
88	4:03:44	0.59	0.08	0.64	-0.03
89	4:03:44	0.56	0.16	0.26	0.42

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

90	4:03:44	0.52	0.17	0.44	0.23
91	4:03:44	0.55	0.33	0.69	0.67
92	4:03:44	0.52	0.36	0.28	0.33
93	4:03:45	0.49	0.46	0.44	0.66
94	4:03:45	0.45	0.41	0.34	0.17
95	4:03:45	0.44	0.42	0.37	0.43
96	4:03:45	0.48	0.48	0.73	0.68
97	4:03:45	0.43	0.44	0.25	0.26
98	4:03:46	0.42	0.38	0.42	0.17
99	4:03:46	0.45	0.39	0.53	0.4
100	4:03:46	0.48	0.28	0.47	-0.08

B.3 Akuisisi data skenario 3

No	*Computer Time*	*Pitch_Kalman*	*Roll_Kalman*	*Pitch*	*Roll*
1	4:35:01	0.56	-0.71	0.52	-0.61
2	4:35:01	0.09	-0.84	0.41	-0.21
3	4:35:02	-0.33	-1.04	0.75	-0.47
4	4:35:02	-0.74	-1.27	0.68	-0.39
5	4:35:02	-1.13	-1.43	0.45	-0.7
6	4:35:02	-1.47	-1.55	0.31	-0.15
7	4:35:03	-1.73	-1.67	0.67	-0.53
8	4:35:03	-1.83	-1.68	0.65	-0.29
9	4:35:03	-1.82	-1.67	0.52	-0.8
10	4:35:03	-1.76	-1.57	0.54	-0.26
11	4:35:03	-1.72	-1.41	-0.29	-0.62
12	4:35:04	-1.41	-1.34	0.62	-0.48
13	4:35:04	-1.14	-1.13	0.7	-0.14
14	4:35:04	-0.99	-0.85	-0.17	-0.18
15	4:35:04	-0.61	-0.83	1.08	-0.56
16	4:35:04	-0.35	-0.77	0.62	-0.62
17	4:35:05	-0.15	-0.61	0.4	-0.51
18	4:35:05	0.11	-0.69	0.63	-0.85
19	4:35:05	0.16	-0.51	0.43	-0.03
20	4:35:05	0.22	-0.27	0.03	0.12
21	4:35:05	0.44	-0.37	0.93	-0.81
22	4:35:06	0.46	-0.23	0.44	-0.02
23	4:35:06	0.37	-0.08	-0.37	0.02
24	4:35:06	0.43	-0.16	0.48	-0.59
25	4:35:06	0.36	-0.07	0.09	-0.03
26	4:35:06	0.50	-0.21	0.67	-0.51
27	4:35:07	0.51	-0.31	0.46	-0.71
28	4:35:07	0.50	-0.18	0.22	0.03
29	4:35:07	0.49	-0.32	0.37	-1.01
30	4:35:07	0.54	-0.38	0.68	-0.47
31	4:35:07	0.63	-0.29	0.71	0
32	4:35:08	0.66	-0.49	0.59	-0.91
33	4:35:08	0.53	-0.52	0.03	-0.65
34	4:35:08	0.54	-0.59	0.44	-0.78

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

35	4:35:08	0.55	-0.58	0.55	-0.41
36	4:35:08	0.56	-0.53	0.66	-0.05
37	4:35:09	0.61	-0.52	0.54	-0.56
38	4:35:09	0.62	-0.54	0.71	-0.57
39	4:35:09	0.49	-0.53	0.14	-0.4
40	4:35:09	0.51	-0.47	0.37	-0.32
41	4:35:10	0.60	-0.58	0.73	-0.64
42	4:35:10	0.62	-0.44	0.71	-0.05
43	4:35:10	0.54	-0.37	0.37	-0.37
44	4:35:10	0.48	-0.37	0.34	-0.62
45	4:35:10	0.54	-0.35	0.74	-0.38
46	4:35:11	0.66	-0.45	0.93	-0.77
47	4:35:11	0.65	-0.46	0.64	-0.53
48	4:35:11	0.72	-0.43	0.98	-0.28
49	4:35:11	0.78	-0.58	0.79	-1.16
50	4:35:11	0.65	-0.58	0.18	-0.35
51	4:35:12	0.69	-0.37	0.79	0.32
52	4:35:12	0.66	-0.52	0.42	-1.02
53	4:35:12	0.61	-0.51	0.56	-0.48
54	4:35:12	0.59	-0.52	0.61	-0.45
55	4:35:12	0.67	-0.51	0.85	-0.5
56	4:35:13	0.62	-0.39	0.45	-0.03
57	4:35:13	0.56	-0.4	0.11	-0.55
58	4:35:13	0.50	-0.42	0.4	-0.53
59	4:35:13	0.54	-0.47	0.87	-0.55
60	4:35:13	0.56	-0.41	0.57	-0.19
61	4:35:14	0.51	-0.34	0.33	-0.02
62	4:35:14	0.54	-0.4	0.64	-0.7
63	4:35:14	0.64	-0.36	0.89	-0.39
64	4:35:14	0.64	-0.42	0.76	-0.66
65	4:35:14	0.70	-0.29	0.74	0
66	4:35:15	0.69	-0.32	0.69	-0.48
67	4:35:15	0.72	-0.4	0.89	-0.44
68	4:35:15	0.76	-0.34	0.85	0
69	4:35:15	0.76	-0.45	0.78	-0.5
70	4:35:16	0.71	-0.46	0.64	-0.47
71	4:35:16	0.68	-0.46	0.49	-0.71
72	4:35:16	0.62	-0.57	0.47	-0.85
73	4:35:16	0.65	-0.49	0.95	-0.25
74	4:35:16	0.63	-0.46	0.47	-0.65
75	4:35:17	0.55	-0.58	0.35	-0.86
76	4:35:17	0.59	-0.48	0.75	-0.16
77	4:35:17	0.59	-0.45	0.58	-0.46
78	4:35:17	0.52	-0.47	0.38	-0.58
79	4:35:17	0.47	-0.51	0.29	-0.6
80	4:35:18	0.40	-0.44	0.11	-0.23
81	4:35:18	0.49	-0.5	0.85	-0.51
82	4:35:18	0.50	-0.51	0.52	-0.42
83	4:35:18	0.54	-0.54	0.73	-0.63
84	4:35:18	0.62	-0.66	0.77	-0.76

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

85	4:35:19	0.61	-0.58	0.4	-0.34
86	4:35:19	0.59	-0.64	0.53	-0.71
87	4:35:19	0.50	-0.56	0.37	-0.67
88	4:35:19	0.51	-0.51	0.39	-0.28
89	4:35:19	0.52	-0.64	0.58	-0.95
90	4:35:20	0.46	-0.62	0.38	-0.58
91	4:35:20	0.50	-0.66	0.66	-0.67
92	4:35:20	0.63	-0.79	0.99	-1.08
93	4:35:20	0.54	-0.7	0.35	-0.29
94	4:35:20	0.49	-0.62	0.42	-0.5
95	4:35:21	0.52	-0.57	0.49	-0.39
96	4:35:21	0.58	-0.54	0.88	-0.37
97	4:35:21	0.61	-0.53	0.4	-0.4
98	4:35:21	0.56	-0.6	0.46	-0.86
99	4:35:21	0.60	-0.61	0.87	-0.65
100	4:35:22	0.56	-0.66	0.34	-0.81

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA

LAMPIRAN C HASIL UJI GESTUR TANGAN

C.1 Gestur Maju

Percobaan	Sudut Acuan	Sudut Awal	Sudut Akhir	Keterangan
1	0° - 90°	0.15°	90.34°	Berhasil
2	0° - 90°	0.32°	90.94°	Berhasil
3	0° - 90°	0.46°	90.84°	Berhasil
4	0° - 90°	0.41°	90.17°	Berhasil
5	0° - 90°	0.67°	90.19°	Berhasil
6	0° - 90°	0.97°	90.78°	Berhasil
7	0° - 90°	0.29°	90.81°	Berhasil
8	0° - 90°	0.26°	90.16°	Berhasil
9	0° - 90°	0.47°	90.24°	Berhasil
10	0° - 90°	0.17°	90.27°	Berhasil

C.2 Gestur Mundur

Percobaan	Sudut Acuan	Sudut Awal	Sudut Akhir	Keterangan
1	0° - 90°	0.60°	87.54°	Gagal
2	0° - 90°	0.43°	90.78°	Berhasil
3	0° - 90°	0.92°	90.11°	Berhasil
4	0° - 90°	1.21°	90.45°	Gagal
5	0° - 90°	0.12°	90.65°	Berhasil
6	0° - 90°	0.24°	90.94°	Berhasil
7	0° - 90°	0.56°	90.81°	Berhasil
8	0° - 90°	0.60°	90.62°	Berhasil
9	0° - 90°	0.24°	90.31°	Berhasil
10	0° - 90°	0.30°	90.78°	Berhasil

C.3 Gestur Kiri

Percobaan	Sudut Acuan	Sudut Awal	Sudut Akhir	Keterangan
1	0° - 90°	0.66°	90.81°	Berhasil
2	0° - 90°	0.27°	90.23°	Berhasil
3	0° - 90°	0.10°	90.14°	Berhasil

4	0° - 90°	0.34°	90.12°	Berhasil
5	0° - 90°	0.84°	90.66°	Berhasil
6	0° - 90°	0.99°	90.46°	Berhasil
7	0° - 90°	0.67°	90.49°	Berhasil
8	0° - 90°	0.51°	90.11°	Berhasil
9	0° - 90°	0.73°	90.26°	Berhasil
10	0° - 90°	0.11°	90.91°	Berhasil

C.4 Gestur Kanan

Percobaan	Sudut Acuan	Sudut Awal	Sudut Akhir	Keterangan
1	0° - 90°	0.42°	90.31°	Berhasil
2	0° - 90°	0.76°	90.12°	Berhasil
3	0° - 90°	0.33°	90.16°	Berhasil
4	0° - 90°	0.86°	90.67°	Berhasil
5	0° - 90°	0.74°	90.48°	Berhasil
6	0° - 90°	0.19°	90.91°	Berhasil
7	0° - 90°	0.79°	92.21°	Gagal
8	0° - 90°	0.67°	90.44°	Berhasil
9	0° - 90°	0.16°	90.13°	Berhasil
10	0° - 90°	0.51°	90.01°	Berhasil

Repository Universitas Brawijaya
REPOSITORY.UB.AC.ID
UNIVERSITAS BRAWIJAYA