PEMODELAN PERGERAKAN *NON-LINEAR* DENGAN MEMANFAATKAN FUNGSI GERAK *ANGULAR* PADA *QUADCOPTER*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muliyahati Sutejo
NIM: 145150300111113

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PEMODELAN PERGERAKAN *NON-LINEAR* DENGAN MEMANFAATKAN FUNGSI GERAK *ANGULAR* PADA *QUADCOPTER*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Muliyahati Sutejo
NIM: 145150300111113

Skripsi ini telah diuji dan dinyatakan lulus pada
6 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Gembong Edhi Setyawan, S.T, M.T	Wijaya Kurniawan, S.T, M.T
NIP: 201208 761201 1 001	NIP: 19820125 201504 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

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

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, 20 Juli 2018

Muliyahati Sutejo

NIM: 145150300111113

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas rahmat-Nya penulis dapat menyelesaikan laporan skripsi yang berjudul "PEMODELAN PERGERAKAN *NON-LINEAR* DENGAN MEMANFAATKAN FUNGSI GERAK *ANGULAR* PADA *QUADCOPTER*" dapat terselesaikan. Penulis menyadari bahwa penysunan dan pengerjaan skripsi ini tidak akan berhasil tanpa adanya bantuan dan dorongan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada:

1. Tuhan yang Maha Esa yang selalu memberikan petunjuk serta hikma dalam pengerjaan skripsi ini.
2. Kedua orang tua serta keluarga penulis yang telah memberikan doa serta dukungannya.
3. Bapak Gembong Edhi Set*yaw*an, S.T, M.T dan Bapak Wijaya Kurniawan, S.T, M.T yang telah bersedia dan dengan sabar membimbing serta mengarahkan penulis dalam pengerjaan skripsi ini.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
5. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
6. Bapak Heru Nurwasito, Ir., M.Kom. selaku Wakil Dekan I Bidang Akademik Fakultas Ilmu Komputer Universitas Brawijaya Malang.
7. Bapak Sabriansyah Rizqika Akbar, S.T., M.T., Ph.D selaku ketua jurusan Teknik Komputer Universitas Brawijaya Malang.
8. Seluruh civitas akademika Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak memberi bantuan selama penulis menempuh studi Informatika di Universitas Brawijaya dan selama penyelesaian skripsi ini.
9. Sabita Wildani, Yusril Dewantara, Ayang Setiyo Putri, Achmad Baecuni, Dimas Angger, Andyan Bina, Sabitha Wildani, Amroy Casro, *Cindy Lilian,* Mesra Diana, Enno Roscitra, Haqqi Rizki dan seluruh *Quadcopter team research* telah memberikan ilmu terkait penelitian yang pernah dilaku.kan dan memberikan motivasi dan do'a agar pengerjaan skripsi ini selesai
10. Riyyan Royhan, Yohana Kristina, Bunga Boru dan Olivia Rumisris yang sudah banyak membantu dan memberi dukungan dalam pengerjaan dan penulisan skripsi.
11. Faizal Andy Susilo, Hendriawan, Joniar Dimas dan seluruh dulur-dulur Teknik Komputer-E yang sudah banyak membantu dan memberi dukungan dalam pengerjaan dan penulisan skripsi.

12. Okke Rizki, Bramantyo, Asfar Triyadi, Saudi Amran, Romario Siregar, Rizki Teguh, Anang Malik, Dimas Bagus dan semua teman-teman Efek Rumah Okke yang sudah memberi semangat dan dukungan.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih terdapat banya kekurangan, sehingga saran dan kritik yang membangun sangat diharapkan oleh penulis. Semoga skripsi ini dapat memberikan manfaat dan berguna bagi ilmu pengetahuan khususnya teknik komputer.

Malang, 20 Juli 2018

Penulis

Muliyahati97@gmail.com

ABSTRAK

Parrot AR.Drone 2.0 merupakan salah satu jenis *quadcopter* atau pesawat tanpa awak *Unmanned Aerial Vehicle* (UAV) yang memiliki empat baling baling pada setiap sisinya. Kelebihan yang dimiliki oleh *quadcopter* terletak pada mobilitas dan fleksibilitas pergerakannya. Hal itu dapat dilihat dari banyaknya penelitian yang dilakukan untuk mengembangkan sistem navigasi gerak dari *quadcopter* untuk melewati suatu wilayah. Wilayah lintasan yang dilalui *quadcopter* tidak hanya sepenuhnya lurus namun lintasan berbentuk persimpangan juga bisa ditemukan. Hal tersebut mengharuskan *quadcopter* untuk beralih arah dalam bergerak. Untuk beralih arah dibutuhkan pengendalian pada pergerakan *quadcopter*. Pada penelitian ini akan membangun sebuah sistem *quadcopter* untuk mengendalikan pergerakan *quadcopter* agar dapat bergerak secara *angular* atau beralih arah ke kiri atau ke kanan secara optimal dalam segi waktu maupun ketepatan pergerakan *quadcopter. Quadcopter* dapat bergerak secara *angular* karena adanya nilai dari parameter $angular_z$, $linear_x$ atau $linear_y$ dan nilai iterasi (i) yang kemudian diolah dengan persamaan fungsi *angular* untuk menentukan seberapa besar dan seberapa jauh lintasan yang akan ditempuh. Kecepatan laju quadcopter dapat diatur dengan memberikan nilai pada parameter *linear velocity* pada *quadcopter* dengan range nilai 0 hingga 1. Berdasarkan hasil dari perngujian pada sistem ini, gerak *angular* dapat dilakukan dengan nilai linear velocity v_{xy} terendah adalah 0,7 hingga v_{xy} dan tertinggi yaitu 1. Untuk estimasi waktu dari pergerakan *quadcopter* dalam berbagai arah didapatkan rata – rata waktu 0,018225 detik untuk setiap pergerakan *angular* yang di lakukan dan untuk akurasi jarak (r) yang ditempuh oleh *quadcopter* memiliki tingkat kurasi keberhasilan ketepatan jarak (r) adalah 99,99%.

Kata kunci: *Gerak Quadcopter, Quadcopter, Sistem Navigasi, Gerak* $Angular$

ABSTRACT

Parrot AR.Drone 2.0 is one type of quadcopter or unmanned aerial vehicle (UAV) which has four propellers on each side. The advantage possessed by the quadcopter lies in its mobility and movement. Things that can be seen from anywhere are used to develop navigation systems from quadcopter to call for regions. The trajectory traversed by the quadcopter is not only valid but can also be found. This requires the quadcopter to move direction in motion. To change the direction needed for the quadcopter movement. In this study, a quadcopter system will be built to control the movement of the quadcopter so that it can move angularly or move direction left or right optimally in terms of time and accuracy of quadcopter movement. The quadcopter can move in general because of the values of the $angular_z$, $angular_x$ *or* $angular_y$ *parameters and the iteration value (i) which are then processed with angular functions to determine the size and number of paths to be taken. Quadcopter speed can be given by giving values to linear velocity parameters on a quadcopter with a range of 0 to 1. Based on the results of testing on this system, the motion angle can be done with the lowest* v_{xy} *linear velocity value is 0.7 to* v_{xy} *and the highest is 1. To determine the time of the quadcopter movement in various ways to detect an average time of 0.018225 seconds for each angle of motion carried out and to find out the distance (r) achieved by the quadcopter has a curating rate of success the accuracy of distance (r) is 99.99%.*

Kata kunci: Quadcopter Motion, Quadcopter, Navigation System, $Angular$ *Motion*

DAFTAR ISI

PENGESAHAN ..... **Error! Bookmark not defined.**
PERNYATAAN ORISINALITAS ii
KATA PENGANTAR iv
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR TABEL xi
DAFTAR GAMBAR xiii
DAFTAR LAMPIRAN xiv
BAB 1 .PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 2
1.3 Tujuan 2
1.4 Manfaat 2
1.5 Batasan masalah 2
1.6 Sistematika pembahasan 3
BAB 2 *LAND*ASAN KEPUSTAKAAN 4
2.1 Tinjauan Pustaka 4
2.2 Dasar Teori 5
2.2.1 Quadcopter 5
2.2.2 Gerak *Angular* 7
2.2.3 Sending Command 8
2.2.4 *Robot Operating System* (ROS) 9
BAB 3 METODOLOGI 11
3.1 Metode Penelitian 11
3.2 Studi Literatur 11
3.3 Analisis Kebutuhan 12
3.4 Perancangan Sistem dan Implementasi 12
3.5 Pengujian dan Analisis 12
3.6 Kesimpulan dan Saran 12

BAB 4 Analisis kebutuhan 14
4.1 Kebutuhan Pengguna 14
4.2 Kebutuhan Sistem 14
4.2.1 Kebutuhan Perangkat Keras 14
4.2.2 Kebutuhan Perangat Lunak 16
4.3 Kebutuhan Fungsional 17
4.4 Kebutuhan Non-Fungsional 17
4.4.1 Karateristik Pengguna 17
4.4.2 Lingkungan Operasi 17
4.4.3 Asumsi dan Ketergantungan 17
4.4.4 Batasan Perancangan dan Implementasi 17
BAB 5 PErancangan dan implementasi 19
5.1 Komunikasi Sistem 19
5.1.1 Perancangan Komunikasi Sistem 19
5.1.2 Implementasi Komunikasi Sistem 20
5.2 Fungsi *Angular* 21
5.2.1 Perancangan Fungsi *Angular* 21
5.2.2 Implementasi Fungsi *Angular* 22
5.3 Gerak *Quadcopter* 23
5.3.1 Perancangan Gerak *Quadcopter* 23
5.3.2 Implementasi Gerak *Quadcopter* 24
BAB 6 Pengujian dan analisis 29
6.1 Pengujian Gerak *Angular* 29
6.1.1 Tujuan Pengujian 29
6.1.2 Pelaksanaan Pengujian 29
6.1.3 Prosedur Pengujian 29
6.1.4 Hasil Pengujian 30
6.1.5 Analisis Pengujian 30
6.2 Pengujian Ketepatan 30
6.2.1 Tujuan Pengujian 30
6.2.2 Pelaksanaan Pengujian 30
6.2.3 Prosedur Pengujian 31

6.2.4 Hasil Pengujian 31
6.2.5 Analisis Pengujian 37
Dari hasil perhitungan persentase eror yang telah dilakukan, dan dengan menggunakan rumus 6.2 didapatkan tingkat akurasi sistem berdasarkan jarak yang di tempuh sebesar 99,99% 41
6.3 Pengujian Estimasi Waktu Gerak sistem 41
6.3.1 Tujuan Pengujian 41
6.3.2 Pelaksanaan Pengujian 42
6.3.3 Prosedur Pengujian 42
6.3.4 Hasil Pengujian 42
6.3.5 Analisis Pengujian 46
BAB 7 Penutup 47
7.1 Kesimpulan 47
7.2 Saran 47
DAFTAR PUSTAKA 48
Lampiran 49
A. Kode Program Gerak *Angular* 49

DAFTAR TABEL

Kode Program 5. 1 Inisialisasi Variabel 22
Kode Program 5. 2 Perhitungan Fungsi *Angular* 23
Kode Program 5. 3 Inisialisasi Parameter Gerak *Quadcopter* 24
Kode Program 5. 4 Gerak *take-off* dan *landing* 25
Kode Program 5. 5 Gerak Manuver 26
Kode Program 5. 6 Gerak *Quadcopter* Pada Sumbu x 26
Kode Program 5. 7 Penentuan Arah Gerak *Quadcopter* pada Sumbu x 27
Kode Program 5. 8 Penentuan Arah Gerak *Qudcopter* Pada Sumbu y 27
Tabel 6. 1 Hasil Pengujian Gerak *Angular* Satu Lingkaran Penuh 30
Tabel 6. 2 Jarak Gerak *Angular* Maju - Kiri 32
Tabel 6. 3 Jarak Gerak *Angular* Maju - Kanan 32
Tabel 6. 4 Jarak Gerak *Angular* Mundur - Kanan 33
Tabel 6. 5 Jarak Gerak *Angular* Mundur - Kiri 34
Tabel 6. 6 Jarak Gerak *Angular* Kanan - Kanan 34
Tabel 6. 7 Jarak Gerak *Angular* Kanan - Kiri 35
Tabel 6. 8 Jarak Gerak *Angular* Kiri - Kanan 36
Tabel 6. 9 Jarak Gerak *Angular* Kiri - Kiri 36
Tabel 6. 10 Besar Eror Gerak Maju - Kiri 37
Tabel 6. 11 Besar Eror Gerak Maju - Kanan 38
Tabel 6. 12 Besar Eror Gerak Mundur - Kanan 38
Tabel 6. 13 Besar Eror Gerak Mundur - Kiri 39
Tabel 6. 14 Besar Eror Gerak Kanan - Kanan 39
Tabel 6. 15 Besar Eror Gerak Kanan - Kiri 40
Tabel 6. 16 Besar Eror Gerak Kiri - Kanan 40
Tabel 6. 17 Besar Eror Gerak Kiri - Kiri 41
Tabel 6. 18 Estimasi Waktu Gerak Maju - Kiri 42
Tabel 6. 19 Estimasi Waktu Gerak Maju - Kanan 43
Tabel 6. 20 Estimasi Waktu Gerak Mundur - Kanan 43
Tabel 6. 21 Estimasi Waktu Gerak Mundur - Kiri 44
Tabel 6. 22 Estimasi Waktu Gerak Kanan - Kanan 44

Tabel 6. 23 Estimasi Waktu Gerak Kanan - Kiri 45
Tabel 6. 24 Estimasi Waktu Gerak Kiri - Kanan 45
Tabel 6. 25 Estimasi Waktu Gerak Kiri - Kiri 45

DAFTAR GAMBAR

Gambar 2. 1 Pergerakan Motor *Quadcopter* 5
Gambar 2. 2 Gerakan *Throttle Quadcopter* 5
Gambar 2. 3 Gerakan *Pitch Quadcopter* 6
Gambar 2. 4 Gerakan Roll *Quadcopter* 6
Gambar 2. 5 Gerakan *Yaw Quadcopter* 6
Gambar 2. 6 Gerak *Angular* 7
Gambar 2. 7 Komunikasi Antar Node Pada ROS 9
Gambar 3. 1 Diagram Alir 11
Gambar 4. 1 Diagram Analisis Kebutuhan 14
Gambar 4. 2 *Parrot AR.Drone 2.0* 15
Gambar 5. 1 Alur Perancangan Sistem 19
Gambar 5. 2 Skema Komunikasi Sistem 20
Gambar 5. 3 Koneksi Wi-Fi *AR.Drone* 20
Gambar 5. 4 Tampilan Saat Komputer Terhubung Dengan *Quadcopter* 21
Gambar 5. 5 Alur Perhitungan Persamaan Fungsi *Angular* 21
Gambar 5. 6 Diagram Alir Perancangan Pergerakan *Quadcopter* 24
Gambar 5. 7 Gerak *Angular* Maju dengan lintasan Kekanan 27
Gambar 5. 8 Gerak *Angular* Kiri dengan lintasan Kekanan 28

DAFTAR LAMPIRAN

A. Kode Program Gerak *Angular* 49

BAB 1 .PENDAHULUAN

1.1 Latar belakang

Unmanned Aerial Vehicle (UAV) merupakan pesawat tanpa awak yang saat ini banyak menarik perhatian masyarakat hingga menjadi topik untuk melakukan sebuah penelitian dikarenakan memiliki banyak fungsi serta tujuan seperti kebutuhan militer, sipil, pemantauan wilayah bencana, hingga tujuan beresiko lainya. *Quadcopter* merupakan salah satu jenis UAV. *Quadcopter* mempunyai kelebihan pada mobilitas dan fleksibilitas untuk menjelajahi wilayah yang sempit (Setyawan, et al., 2015). Wilayah lintasan yang dilalui *quadcopter* tidak hanya berbentuk lurus namun ada lintasan berbentuk persimpangan. Hal tersebut mengharuskan *quadcopter* untuk beralih arah dalam bergerak. Untuk beralih arah dibutuhkan pengendalian pada pergerakan *quadcopter*.

Beralih arah dalam melakukan pergerakan dapat digunakan untuk menghindari suatu halangan (*obstacle*). Gerakan tersebut mengandalkan nilai ketepatan dan waktu dalam bergerak secara efektif dan fleksibel agar *quadcopter* tidak menabrak benda disekelilingnya. Dharmawan & Rahmawati (2014) pernah melakukan penelitian untuk membangun sistem *quadcopter* yang dapat menghindari suatu halangan. Pada *quadcopter* ditambahkan sensor inframerah yang dihubungkan oleh Arduino Nano untuk mendeteksi jarak antara halangan dengan *quadcopter*. Namun, penelitian tersebut hanya melakukan pergerakan ke arah yang berlawanan pada saat menemui suatu halangan, sedangkan untuk mengikuti lintasan dibutuhkan peralihan arah gerak yaitu berbelok ke kiri atau ke kanan.

Berdasarkan permasalahan dan kekurangan dalam penelitian yang dilakukan oleh Dharmawan & Rahmawati (2014), pada penelitian ini akan membangun sebuah sistem *quadcopter* untuk mengendalikan pergerakan *quadcopter* saat beralih arah. Sistem yang dibangun adalah sistem pemodelan gerak *angular* atau melengkung untuk mengoptimalkan gerak saat beralih arah ke kiri atau ke kanan dilakukan oleh *quadcopter* pada wilayah lintasan yang terdapat persimpangan. Dengan memanfaatkan nilai parameter $angular_z$ dan *linear velocity* yang terdapat pada *quadcopter* maka dibuatlah suatu persamaan fungsi *angular* yang akan diterapkan pada sistem. Dimana dengan menggunakan persamaan fungsi *angular* tersebut *quadcopter* diharapkan dapat bergerak secara *angular* atau beralih arah ke kiri atau ke kanan secara optimal dalam segi waktu maupun ketepatan pergerakan *quadcopter*.

1.2 Rumusan masalah

1. Bagaimana cara membuat sistem navigasi *quadcopter* agar dapat bergerak secara *angular*?
2. Berapa nilai kecepatan yang akurat untuk digunakan dalam melakukan gerak *angular* dengan lintasan ¼ lingkaran pada tiap arah gerakan?
3. Berapa estimasi waktu yang didapat dari pengimplementasian persamaan fungsi *angular* pada sistem?
4. Berapa tingkat akurasi jarak/jari-jari (r) yang ditempuh oleh *quadcopter*?

1.3 Tujuan

1. Mengetahui bagaimana cara membuat sistem navigasi untuk mengatur gerak *angular* dari *quadcopter*.
2. Mengetahui besar nilai kecepatan yang dapat digunakan untuk gerak *angular* dengan lintasan ¼ lingakaran pada tiap arah yang berbeda.
3. Mengtahui estimasi waktu dari setiap pergerakan yang dilakukan dengan menggunakan persamaan *angular*.
4. Mengetahui besar tingkat akurasi jarak(r) yang dapat ditempuh oleh *quadcopter.*

1.4 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Membantu pengguna khususnya pemula agar lebih mudah untuk mengendalikan *quadcopter*.
2. *Quadcopter* dapat melakukan *angular* secara otomatis dengan lintasan ¼ lingkaran dengan memanfaatkan fungsi parameter *angular* pada *quadcopter*.
3. Dapat digunakan sebagai referensi untuk melakukan penelitian terkait pergerakan *quadcopter* secara *non-linear* atau melengkung sehingga dapat menghasilkan penelitian yang baik lagi.

1.5 Batasan masalah

Adapun batasan-masalah pada penelitian ini agar lebih focus pada perumusan masalah dan tidak menyimpang adalah:

1. *Quadcopter* yang dipakai adalah *Parrot AR Drone 2.0.*
2. Menggunakan persamaan fungsi *angular* untuk menentukan gerak *quadcopter.*
3. Penelitian yang dilakukan hanya fokus kepada pergerakan dari *quadcopter.*
4. *Quadcopter* hanya bergerak 2-Dimensi, yaitu pada sumbu x dan sumbu y.
5. *Quadcopter* hanya bergerak dengan lintasan berbentuk ¼ lingkaran.

1.6 Sistematika pembahasan

Sistematika pembahasan bertujuan sebagai penjelasan umum dari bagian bab yang terdapat dalam penelitian ini agar memudahkan pembaca dalam mengikuti alur penelitian. Adapun Sistematika pembahasan yang diterapkan adalah sebagai berikut:

BAB 1 Pendahuluan

Berisi tentang Latar Belakang, Rumusan Masalah, Batasan Masalah, Tujuan penulisan, Manfaat Penulisan, serta Sistematika Penulisan.

BAB 2 *Land*asan Kepustakaan

Membahas mengenai apasaja tinjauan pustaka dari penelitian sebelumnya dan berisi mengenai dasar teori yang menjadi acuan dalam pembuatan sistem ini.

BAB 3 Metodoligi

Menjelaskan tentang langkah-langkah kerja dari penelitian yang akan dilakukan. Dengan metode yang digunakan yaitu metode *waterfall*, yang meliputi studi literature, analisis kebutuhan, perancngan sistem, implemantasi, pengujian beserta analisis, kesimpulan serta saran.

BAB 4 Analisis dan Kebutuhan

Membahas apa saja yang menjadi kebutuhan fungsional dan *non*-fungsional yang diperlukan dalam perancangan sistem.

BAB 5 Perancangan dan Implementasi

Membahas konsep dari sistem yang akan diimplementasikan, mulai dari dasar teori yang telah dipelajari sesuai dengan perancangan sistem

BAB 6 Pengujian dan Analisis

Berisi penerapan atau pengimplementasian dari aplikasi yang sudah dirancang dan dikonsep pada bab perancangan. Tahapan ini biasa disebut pokok atau inti dari penelitian yang diambil, karena hasil akhir yang diperoleh dari suatu penelitian dapat dilihat saat sistem di terapkan atau di implementasikan.

BAB 6 Penutup

Berisikan hasil akhir atau kesimpulan dari hasil penulisan/penelitian dan juga saran dari pengguna atau *user* untuk proses pengembangan lanjutan.

BAB 2 *LAND*ASAN KEPUSTAKAAN

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

2.1 Tinjauan Pustaka

Penelitian pada gerak *quadcopter* untuk mengindari suatu *obstacle* atau halangan diperkuat dengan penelitihan terdahulu yang di lakukan oleh Andika,Yani & Cardewa (2018) yang membahas tentang rancang bangun *quadcopter* dengan menggunakan sensor inframerah untuk mendeteksi suatu halangan. Dimana output dari penelitian tersebut berupa LED *(Light Emitting Diode)* dan suara yang dikeluarkan oleh *buzzer* apabila salah satu sensor mendeteksi halangan. Selain itu, pada penelitian tersebut juga menyatakan apabila terdapat suatu halangan yang terdeteksi oleh sensor maka *quadcopter* akan bergerak kearah yang berlawanan. Data akurasi sensor dalam merespon suatu objek pada penelitian ini memiliki nilai akurasi yang tinggi apabila penelitian di lakukan didalam ruangan namun, data akurasi sensor bernilai rendah apabila penelitian di lakukan diluar ruangan.

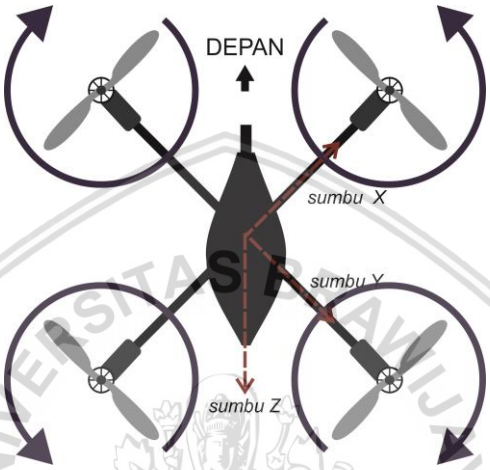
Penelitian lainnya dilakukan oleh Dharmawan & Rahmawati (2014) yaitu membuat sistem penghindar halangan otomatis dan sistem penahan ketinggian gerak *quadcopter* dengan meggunakan *quadcopter* berbasis *Arduino Nano*. Pada penelitian ini juga menggunakan sensor inframerah untuk mendeteksi jarak antara *quadcopter* dengan halangan serta menggunakan sensor ultrasonic untuk mendeteksi ketinggian terbang *quadcopter*. Tidak hanya menggunakan sensor, pada penelitian ini juga menggunakan perhitungan dengan algorima PID untuk mengatur pergerakan motor dari *quadcopter*. Sehingga saat sensor mendeteksi suatu halangan, *quadcopter* akan bergerak kearah yang berlawanan pada sumbu yang sama.

Dari penelitian-penelitian yang telah dilakukan sebelumnya akan menjadi acuan dalam penelitian yang akan dilakuakan. Pada penelitian ini akan dirancang sebuah sistem pemodelan gerak *angular* untuk mencari gerak paling efektif yang dapat dilakukan oleh *quadcopter* menggunakan persamaan fungsi *angular*. Persamaan fungsi *angular* didapatkan dengan memanfaatkan nilai *linear velocity* v_{xy} dan nilai pada parameter $angular_z$ yang terdapat pada *quadcopter*. Adapun tujuan dari penelitian ini yaitu membuat *quadcopter* dapat bergerak secara efektif baik dari segi waktu maupun ketepatan serta *quadcopter* dapat melakukan gerak *angular* (melengkung) sesuai dengan lintasan yang diberikan secara fleksibel.

2.2 Dasar Teori

2.2.1 Quadcopter

Quadcopter adalah robot tanpa awak yang memiliki empat buah motor yang dikonfigurasikan dalam bentuk menyilang. Dimana pada masing masing motornya terpasang sebuah baling-baling. Setiap pasang rotor yang berpasangan berputar pada arah yang sama. Satu pasang berputar searah jarum jam dan satu pasang lainnya berputar pada arah sebaliknya.

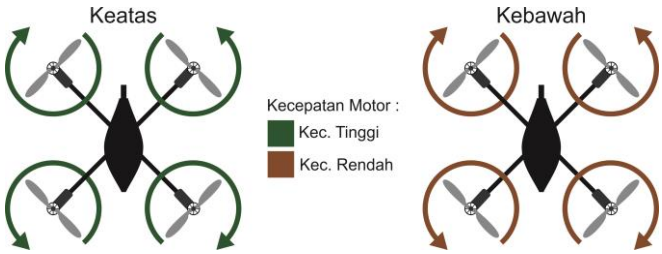


Gambar 2. 1 Pergerakan Motor *Quadcopter*

Dapat dilihat pada Gambar 2.1 bahawa keempat motor pada *quadcopter* dapat memiliki kecepatan yang berbeda-beda, untuk itu dengan mengkombinasikan kecepatan masing masing motor yang ada dapat menghasilkan gaya gerak pada *quadcopter* (Anggarjito, 2013). Beberapa pergerakan yang dapat dilakukan oleh *quadcopter* diantaranya:

a. Throttle

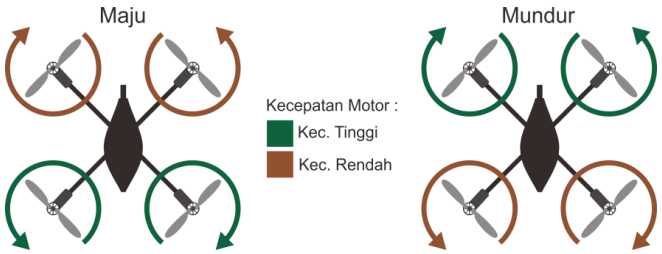
Throttle adalah gerakan kearah atas dan bawah yang bergerak sepanjang sumbu z. Gerakan ini dapat dilakukan apabila keempat motor yang terpasang pada *quadcopter* memiliki besar kecepatan yang sama, seperti pada Gambar 2.2.



Gambar 2. 2 Gerakan *Throttle Quadcopter*

b. *Pitch*

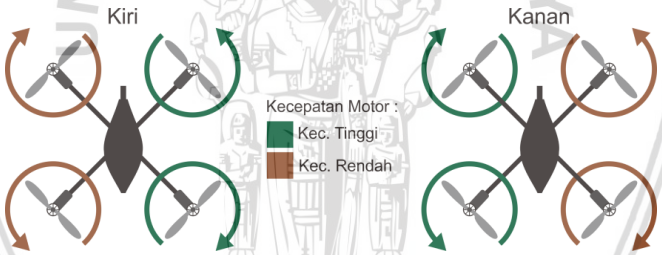
Pitch adalah gerakan kearah depan dan belakang berdasarkan sumbu x. Dapat dilihat pada Gambar 2.3 bahwa, gerakan *pitch* dapat dihasilkan dengan meningkatkan atau menurunkan kecepatan motor bagian depan dan belakang *quadcopter*.



Gambar 2. 3 Gerakan *Pitch Quadcopter*

c. *Roll*

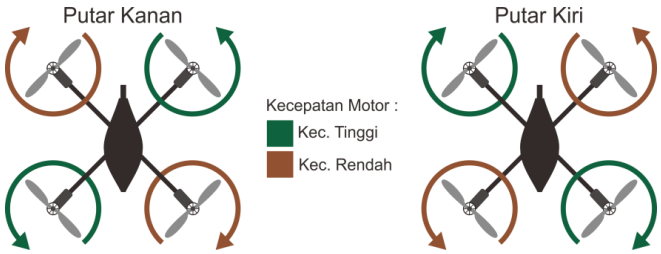
Roll adalah kebalikan dari gerakan *pitch*. Jika gerakan *pitch* adalah gerakan kedepan dan kebelakang berdasarkan sumbu y, maka gerakan roll adalah gerakan kearah kanan dan kiri berdasarkan sumbu y. Gambar 2.4 menunjukkan bahwa, untuk menghasilkan gerakan roll kecepatan motor yang harus diatur ialah motor bagian kanan dan kiri.



Gambar 2. 4 Gerakan Roll *Quadcopter*

d. *Yaw*

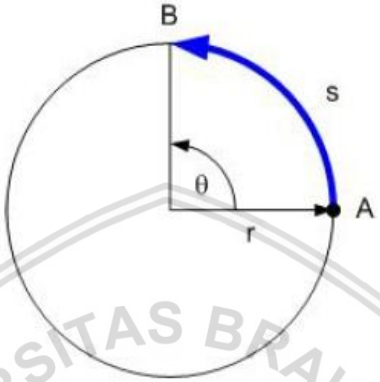
Yaw adalah gerakan memutar kearah kanan dan kiri berdasarkan rotasi pada sumbu z. Gambar 2.5 menunjukkan bahwa gerak *yaw* dapat dihasilkan dengan mengatur kecepatan sepasang motor pada *quadcopter*.



Gambar 2. 5 Gerakan *Yaw Quadcopter*

2.2.2 Gerak *Angular*

Suatu benda dikatakan bergerak secara *angular* atau melingakar apabila benda tersebut dapat bergerak pada suatu lintasan berberntuk lingkaran dengan kecepatan konstan (Hisham.id, 2015). Gerak agular yang berjalan pada dua arah (misalnya pada sumbu x dan sumbu y)merupkan salah atu jenis gerak non-liner 2 dimensi atau biasa disebut dengan gerak lengkung. Gambaran gerak *angular* untuk lebih jelasnya dapat dilihat pada Gambar 2.6.



Gambar 2. 6 Gerak *Angular*

Sumber : (CLC, 2015)

Gambar 2.6 dapat dilihat bahwa sebuah benda dari titik A bergerak menuju titik B dengan lintasan berbentuk lingkaran. Gerak *angular* memiliki besaran berupa posisi sudut θ. Selain itu, gerakan ini juga dapat dilakukan pada qudcopter karena *quadcopter* memiliki suatu parameter $angular_z$ yang dapat membuat *quadcopter* dapat bergerak secara *angular* apabila parameter tersebut diberikan suatu nilai. Adapun untuk nilai parameter $angular_z$ dan nilai parameter lainnya memiliki range nilai -1.0 sampai 1.0 (Monajjemi, 2015) dan persamaan fungsi *angular* yang digunakan pada *quadcopter* adalah sebagai berikut:

$$v_{xy\_n} = \frac{V_{xy\_max}}{V_{xy\_input}} \quad (2.1)$$

$$d_n = d_{min} \times n$$

$$n = \frac{d_n}{d_{min}} \quad (2.2)$$

$$a_{z\_n} = \frac{a_{z\_max}}{n} \quad (2.3)$$

$$i_n = (n \times i) \times v_{xy\_n} \quad (2.4)$$

Dimana:

v_{xy_n} : nilai *linear velocity*

V_{xy_max} : nilai maksimal *linear velocity*

V_{xy_input} : *input* nilai *linear velocity*

n : jumlah langkah perpindahan

d_n : *input* jarak/diameter lingkaran

d_{min} : jarak/diameter minimum lingkaran

a_{z_n} : nilai rotasi gerak pada $angular_Z$

a_{z_max} : nilai maksimal rotasi gerak pada $angular_Z$

i : jumlah iterasi satu lingkaran penuh

i_n : jumlah iterasi ke-n

Untuk membuat pergerakan *Angular* dengan sudut θ 90° atau dengan kata lain bergerak pada lintsan ¼ lingkaran maka nilai dari iterasi dapat dibagi dengan 4.

2.2.3 Sending Command

Setiap robot pesawat tanpa awak dapat melakukan gerak sesuai dengan yang diinginkan apabila mendapat perintah atau intruksi yang diberikan oleh para penggunanya. Pada robot *quadcopter* jenis AR.Drone terdapat fungs sending command untuk mengirim sebuah perintah kepada sebuah package yang terdapat pada *quadcopter* agar dapat melakukan gerak sesuai perintah yang diberikan. Seperti, *Quadcopter* akan melakukan *take-off* dan *landing* apabila pesan ROS *std_msgs / Empty* dipublikasikan pada masing-masing package *ardrone/take-off* dan *ardrone / landing*. Untuk membuat drone terbang atau melakukan *manuver* setelah melakukan *take-off* dapat mempublikasikan pesan bertipe *geometry_msgs :: Twist* pada *cmd_vel*.

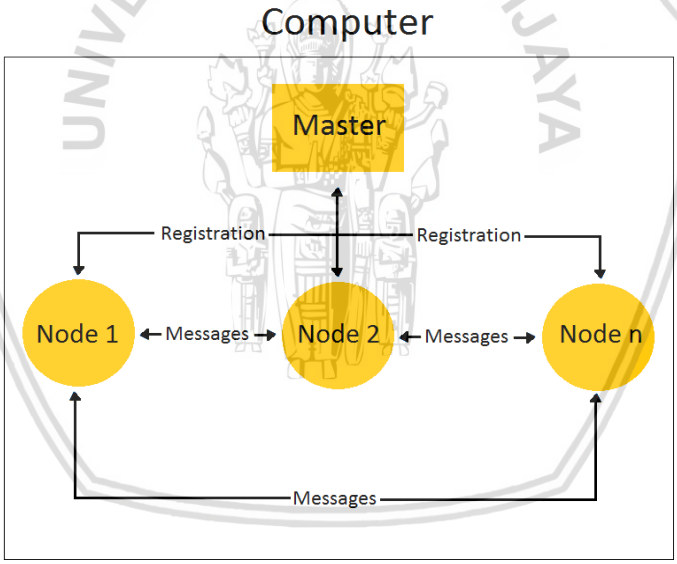
Menurut Monajjmi (2015), *geometry_msgs :: Twist* pada *quadcopter* memiliki enam variabel dengan *range* nilai untuk setiap komponennya atau parameternya harus bernilai antara -1.0 dan 1.0. adapaun enam variabel pada geometry_msgs:: yang dapat dijalan oleh *quadcopter* diantaranya:

1. Linear.x : bernilai positif (+) bergerak maju
   Bernilai negative (-) bergerak mundur
2. Linear.y : bernilai positif (+) bergerak kiri
   Bernilai negative (-) bergerak kanan
3. Linear.z : bernilai positif (+) bergerak keatas
   Bernilai negative (-) bergerak kebawa
4. *Angular*.z : bernilai positif (+) orientasi kiri dalam derajat
   Bernilai negative (-) orientasi kanan dalam derajat

Kedua variable *geometry_msgs :: Twist* yaitu *Angular*.x dan *Angular*.y digunakan untuk mengaktifkan/menonaktifkan mode *auto hover* dengan cara memberikan nilai 0 pada kedua parameter.

2.2.4 *Robot Operating System* (ROS)

*Robot Operating System (*ROS) dikembangkan pada 2007 oleh *Stanford Artificial Intelligence Laboratory* (SAIL) dan Februari 2013, kepengurusan ROS dialihkan kepada *Open Source Robotics Foundation*. ROS dirilis di bawah ketentuan lisensi BSD (Berkeley Software Distribution) dan merupakan perangkat lunak *open source* yang dapat diakses secara gratis untuk keperluan komersil dan penelitian. ROS merupakan sebuah *framework* yang sangat fleksibel. ROS merupakan kumpulan *tools, libraries,* dan konvensi yang memiliki tujuan untuk menyederhanakan task yang kompleks dan menciptakan robot yang handal dalam berbagai *platform* robotik (ER, 2018). ROS menyediakan suatu layanan yang diharapkan seperti abstaksi perangkat keras, kontrol perangkat tingkat rendah, implementasi fungsi yang sering digunakan, *massage-passing* antar proses, manajemen *package.* Sistem pada ROS terdiri dari beberapa *node* independen, dimana pada setiap *node* dapat berkomunikasi dengan node yang lain menggunakan model *publish/subscribe*. Komunikasi antar *node* pada ROS diawali dengan menjalankan ROS MASTER. Apabila Master tidak dijalankan, maka *node* tidak dapat menemukan satu sama lain untuk bertukar pesarn atau meminta suatu layanan. Untuk lebih jelasnya dapat dilihat pada Gambar 2.7.



Gambar 2. 7 Komunikasi Antar Node Pada ROS

Sumber: (ER, 2018)

Node pada ROS ditulis dengan menggunakan bahasa pemrograman c++ dan python. Seperti yang terlihat pada Gambar 2.7 node dapat berkomunikasi satu sama lain dengan cara mengirimkan pesan. Pesan yang dikirimkan memiliki rute skematik yaitu publish/subscribe. Salah satu node mengirim pesan dan dipubish topik yang diberikan. Dimana yang dimaksud dengan topik adalah nama yang digunakan untuk mengidentifikasi konten pesan. Untuk membuat, memodifikasi, atau mengakses package, ROS memberi beberapa tools yang dapat mempermudah untuk mengakses package seperti:

rospack : digunakan untuk mendapatkan informasi atau menemukan package yang ada pada sistem.

roscreate-pkg : digunakan untuk membuat perintah baru

rosmake : Perintah ini digunakan untuk mengkompilasi sebuah package

rosdep : menginstal dependensi sistem package

rxdeps : Perintah ini digunakan jika ingin melihat dependensi package sebagai grafik

Untuk berpindah antara package, folder dan file, ROS memberi sebuah package rosbash, yang menyediakan beberapa perintah seperti:

roscd : Perintah ini membantu mengubah direktori

rosed : Perintah ini digunakan untuk mengedit file

roscp : Perintah ini digunakan untuk menyalin file dari beberapa paket

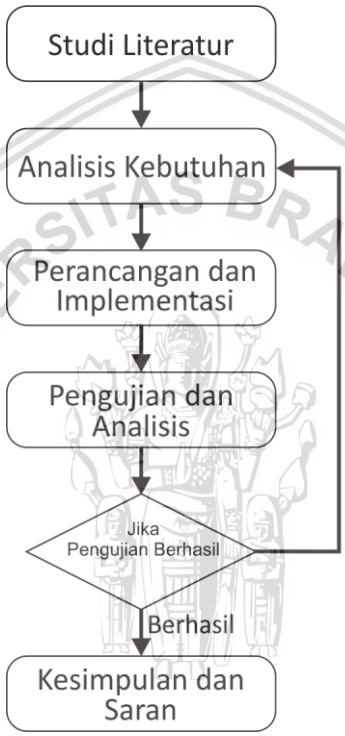
rosd : Perintah ini mencantumkan direktori paket

rosls : Perintah ini berisi daftar file dari sebuah paket

BAB 3 METODOLOGI

3.1 Metode Penelitian

Bab ini menjelaskan mengenai langkah-langkah yang akan dilakukan dalam perancangan sebuah sistem. Pengerjaan pada penelitian ini mengikuti alur dari diagram alir *waterfall* model yang meliputi Analisis Kebutuhan, Perancangan Sistem, Implementasi, Pengujian dan Analisis, serta pengambilan Kesimpulan dan saran. Gambar 3. 1 merupakan diagram alir yang menunjukan tahapan-tahapan pengerjaan penelitian.



Gambar 3. 1 Diagram Alir

Jika dilihat dari Gambar 3. 1 terdapat beberapa alur yang memungkinkan untuk diulang pada tahap sebelumnya. Hal tersebut guna untuk melakukan pengecekan jika terjadi suatu kesalahan dalam pengerjaan penelitian. Seperti pada tahap implementasi dan tahap pengujian, apabila ditemukan suatu kendala seperti hasil output yang dihasilkan tidak sesuai dengan input yang diberikan maka system perlu kembali pada tahap sebelumnya yaitu tahap perancangan.

3.2 Studi Literatur

Bagian ini akan menjelaskan apa saja yang menjadi dasar teori atau literature yang digunakan untuk mendukung penelitian pemodelan gerak *non-linear* dengan menggunakna fungsi gerak *angular* pada *quadcopter*. dasar teori yang digunakan sebagai acuan dalam penelitian ini adalah sebagai berikut.

1. AR Drone 2.0

2. Gerak *angular*
3. *Sending Command* pada Quadcopter
4. *ROS*

3.3 Analisis Kebutuhan

Bagian ini membahas mengenai apa saja yang akan dibutuhkan dalam merancang suatu system sepeti kebutuhan antarmuka pengguna yang menjelaskan mengenai apasaja kebutuhan yang diperlukan agar sistem dapat berinteraksi dengan penggunanya, kebutuhan sistem yang meliputi kebutuhan perangkat keras maupun perangkat lunak, serta kebutuhan fungsional dan non-fungsional

3.4 Perancangan Sistem dan Implementasi

Perancangan sistem merupakan tahapan untuk membangun sistem yang dibutuhkan pada penelitian ini. Perancangan pada sistem ini akan dibagi menjadi perancangan komunikasi antar sistem dan perancangan pergerakan *quadcopter*.

Sedangkn tahap implementasi sistem akan dilakukan berdasarkan perancangan sistem yang telah dibuat sebelumnya. Tahap pertama yang dilakukan adalah menghubungkan komunikasi antara perangkat keras dengan ROS agar kode program dapat dijalankan. Setelah itu tahap pembuatan *source code* untuk mengatur pergerakan *quadcopter* secara otomatis sesuai dengan lintasan yang di tentukan, dan selnjutnya implementasi gerak *quadcopter* untuk mengetahui kinerja dan keefektivan gerakan *quadcopter* sesuai dengan source code yang telah dibuat.

3.5 Pengujian dan Analisis

Pada tahap ini akan dilakukan pengujian serta analisis untuk mengetahui bahwa kinerja dari sebuah sistem yang dibuat sudah sesuai dengan yang diharapkan. Pengujian yang akan dilakukan meliputi ketepatan gerakan, kecepatan dan waktu yang dibutuhkan untuk menempuh suatu lintasan. Apabila tahap pengujian dan analisis telah mendapatkan hasil yang sesuai dengan yang diharapkan maka akan dilanjutkan pada tahap terakhir yaitu penarikan kesimpulan serta saran. Namun, jika hasil yang di peroleh pada pengujian dan analisis belum sesuai maka akan kembali pada tahap perancangan.

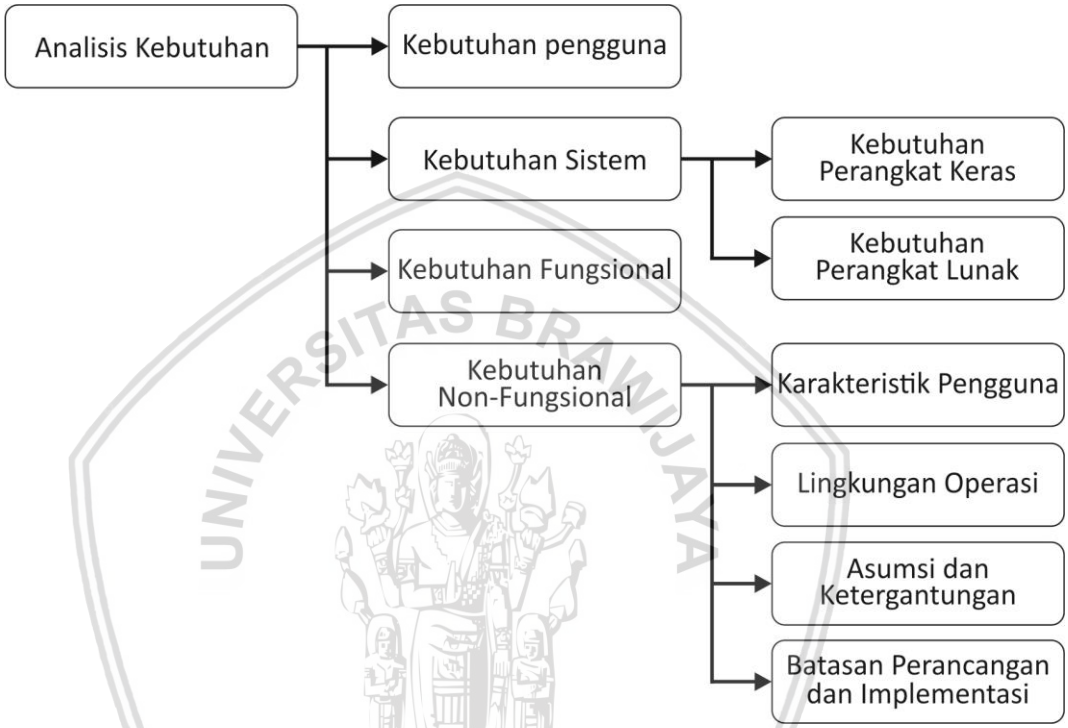
3.6 Kesimpulan dan Saran

Penarikan kesimpulan adalah tahapan terakhir yang dilakukan apabila semua tahapan telah selesai dilakukan. Kesimpulan didapatkan dari pengujian dan analisis yang telah dilakukan sebelumnya. Adanya kesimpulan dari penelitian ini juga dapat dimanfaatkan sebagai acuan dalam penelitian lebih lanjut guna mengembangkan sistem yang telah ada. Selain kesimpulan, juga terdapat saran

pada akhir penulisan yang diharapkan dapat memberikan masukan untuk pengembangan sistem agar penelitian ini dapat menjadi lebih baik lagi.

BAB 4 ANALISIS KEBUTUHAN

Pada bab ini akan dijelaskan mengenai apa saja kebutuhan yang diperlukan oleh sistem. Terdapat empat bagian analisis kebutuhan yang akan dijabarkan pada bab ini yaitu kebutuhan pengguna, kebutuhan sistem, kebutuhan fungsional serta kebutuhan non-fungsional.



Gambar 4. 1 Diagram Analisis Kebutuhan

4.1 Kebutuhan Pengguna

Kebutuhan pengguna adalah kebutuhan yang harus ada agar pengguna dapat memonitoring sistem sehingga sistem dapat berjalan sesuai keinginan. Kebutuhan antarmuka pengguna pada sistem ini menggunakan window pada linux untuk menampilkan informasi yang telah diolah. Informasi yang akan ditampilkan oleh *quadcopter* berupa nilai kecepatan *quadcopter* dan waktu tempuh *quadcopter*.

4.2 Kebutuhan Sistem

Terdapat dua mcam tahapan pada analisis kebutuhan sistem yaitu kebutuhan perangkat keras dan perangkat lunak.

4.2.1 Kebutuhan Perangkat Keras

Adapun kebutuhan perangkat keras yang dibutuhkan dalam pembuatan system ini yaitu,

1. *Parrot AR.Drone 2.0*

   Parrot AR.Drone 2.0 merupakan salah satu jenis pesawat tanpa awak yang memiliki empat buah baling-baling dan dapat dikendalikan oleh perangkat iOS dan Android, komunikasi pada *Parrot AR.Drone 2.0* menggunakan *wireless* (Wi-Fi) tanpa perlu koneksi internet atau *router* (Catanzariti, 2012). Selain itu *Parrot AR.Drone 2.0* juga mendukung kontrol secara otomatis dengan memberi fasilitas kepada *quadcopter* untuk dapat terbang dan mendarat menggunakan sistem yang mudah dioperasikan dengan tampilan yang *user-friendly* (SA, 2016).

(a) **(b)**

Gambar 4. 2 *Parrot AR.Drone 2.0*
Sumber : (SA, 2016)

Pada Gambar 4.2 dapat dilihat bahwa *Parrot AR.Drone 2.0* dapat diterbangkan pada area *indoor* maupun *outdoor*. Perbedaan penggunaan didalam ruangan dengan diluar ruangan terletak pada bentuk *hull* yang digunakan. Gambar 4.2(a) merupakan *Parrot AR.Drone 2.0* yang digunakan untuk diluar ruangan. Sedangaan Gambar 4.2(b) merupakan *Parrot* AR.Drone untuk didalam ruangan, *hull* yang terpasang pada keseluruhan baling baling tersebut berfungsi untuk melindungi kerusakan baling baling akibat benturan ketika terjadi suatu kesalahan saat melakukan penerbangan. Selain itu, *Parrot* AR.Drone dilengkapi dengan kamera beresolusi tinggi 720p 30 fps yang dipasang secara onboard sehingga memungkinkan untuk mengambil gambar atau video dengan jangkauan hingga 50 meter dengan baterai *Lithium-Ion Polymer* sebesar 1000 mAh yang membuat *quadcopter* dapat terbang selama 12 menit, dan bergerak menggunakan sistem operasi *linux* 2.6.32. *Parrot AR.Drone 2.0* telah menggunakan 1GHz 32 bit ARM *Cortex A8 processor* dengan 800MHz *video* DSP TMS320DMC64x. selain sensor *ultrasonic* dan sensor *hall effect* (tekanan), *AR.Drone 2.0* meiliki sensor IMU yang terdiri dari *3-axis gyro sensor, 3-axis accelerometer sensor, 3-axis magnetometer* sensor.

2. Komputer / Laptop

   Komputer merupakan salah satu kebutuhan perangkat keras yang berperan penting dalam pembuatan sistem ini. Untuk itu komputer yang dapat digunakan harus memenuhi spesifikasi minimum agar sistem operasi *Ubuntu* dapat dijalankan. Spesifikasi minimum kompuetr yang direkomendasikan yaitu memiliki prosessor 64 bit, RAM 2GB atau lebih, 4GB *disk space* (untuk installasi operasi sistem *Ubuntu*).

4.2.2 Kebutuhan Perangat Lunak

Sedangkan kebutuhan perangkat lunak yang dibutuhkan yaitu,

1. Sistem Operasi *Linux Ubuntu* versi 14.04

   Ubuntu merupakan salah satu sistem operasi yang bersifat open source dan berbasis Linux Debian. *Ubuntu* memberikan kebebasan kepada pengguna untuk dapat mendapatkan, merubah dan mendistribusikan perangkat lunak sesuai dengan yang dibutuhkan hingga perangkat lunak tersebut dapat bekerja sesuai dengan yang diinginkan. Pada penelitian ini *Ubuntu* yang digunakan adalah *Ubuntu* dengan versi 14.04 dikarenakan versi tersebut dapat terhubung dengan ROS tanpa adanya kendala.
2. ROS

   ROS (*Robot Operating System*) ialah *framework* yang fleksibel dan bersifat *open source*. ROS merupakan kumpulan *tools*, *libraries*, dan konvensi yang memiliki tujuan untuk menyederhanakan suatu *task* yang kompleks dan handal pada berbagai *platform* robot. *Framework* pada ROS dapat ditulis dengan menggunakan bahasa pemrograman *Pyhton*, C++, dan *Lisp* untuk mempermudah pengguna dalam mengoperasikan robot.
3. *Python*

   Python merupakan bahasa pemrograman yang menggunakan mode *interpreted* yang berarti suatu program yang menterjemahkan sebuah intruksi agar dapat di mengerti oleh komputer. Salah satu keuntungan bahasa pemrograman ini adalah dapat melakukan *maintenance* dengan mudah pada kode program yang dibuat. Selain itu bahasa pemrograman python juga dapat digunakan pada semua sistem operasi komputer salah satunya *linux*.
4. *PyCharm*

   PyCharm merupakan salah satu *Interface Development Environntment* (IDE) yang paling banyak digunakan untuk bahasa pemrograman python. Selain mendukung python versi 2.x dan 3.x, *pycharm* juga sangat kompatibel pada windows, macOs dan linux. Pycharm banyak dipilih karena dapat membantu penulisan kode program menggunakan bahasa pemrograman python dengan mudah dan efisien.

4.3 Kebutuhan Fungsional

Kebutuhan fungsional menjelaskan bagaimana sistem dapat menghasilkan keluaran atau output yang sesuai dengan yang diinginkan. Untuk itu kebutuhan yang harus dipenuhi oleh sistem adalah

1. Sistem dapat bergerak secara *angular* sesuai dengan kecepatan v_{xy} dan jari-jari yang di inputkan
2. Sistem dapat menampilkan nilai kecepatan dari data navigasi *quadcopter*
3. Sistem dapat menampilkan estimasi waktu tempuh saat *quadcopter* melakukan gerak *angular*

4.4 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah kebutuhan yang menjelaskan tentang apa saja batasan terhadap ebutuhan sistem. Yang termasuk kebutuhan non-fungsional pada sistem ini adalah sebagai berikut.

4.4.1 Karateristik Pengguna

Karakteristik pengguna ditujukan kepada masyarakat umum terlebih pada pengguna *quadcopter*. Dengan adanya sistem ini diharapkan dapat membantu dan mempermudah dalam mengendalikan *quadcopter*.

4.4.2 Lingkungan Operasi

Persyaratan lingkungan operasi untuk menjalankan sistem ini adalh sebagai berikut.

1. Area yang dibutuhkan agar *quadcopter* dapat bergerak secara optimal adalah pada ruangan yang memiliki luas area 8 x 8 meter.

4.4.3 Asumsi dan Ketergantungan

1. Fungsi parameter $angular_z$ pada *quadcopter* dapat digunakan sebagai pemodelan gerak *quadcopter* secara *angular* atu melengkung sesuai dengan lintasan yang di tentukan untuk mecari gerak yang paling efektif yang dapat dilakukan.
2. Sistem hanya berfungsi apabila komputer telah terhubung dengan Wi-Fi *AR.Drone quadcopter*.
3. Sistem hanya dapat dijalankan pada sistem operasi *Linux Ubuntu* 14.04 yang telah terhubung dengan *Robot Operating System* (ROS).

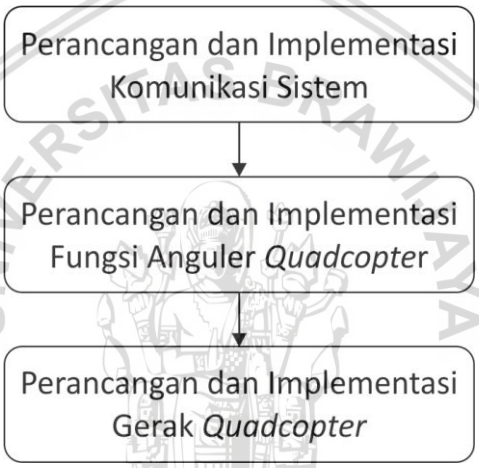
4.4.4 Batasan Perancangan dan Implementasi

1. *Quadcopter* dapat terbang secara *angular* / melingkar pada sudut 90 derajat atau ¼ lingkaran.
2. *Quadcopter* akan melakukan gerak *angular* setelah *quadcopter* berada pada posisi *manuver* dan mendapatkan inputan dari pengguna.

3. *Quadcopter* hanya bergerak pada dua sumbu yaitu pada sumbu x dan y yang berati, *quadcopter* hanya dapat melakukan pergerakn *angular* kearah depan, belakan, kanan dan kiri.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan menjelaskan tahap-tahap dari perancangan dan implementasi sistem. Seperti pada Gambar 5.1 tahapan perancangan dan implementasi sistem ini meliputi perancangan dan implementasi Komunikasi Sistem yang membahas mengenai bagaimana cara menghubungkan *quadcopter* dengan *quadcopter* agar dapat berkomunikasi, Perancangan dan Implementasi Fungsi *Angular* membahas mengenai bagaimna tahapan mengolah suatu data *input* hingga menghasilkan suatu data atau nilai intruksi untuk *quadcopter* melakukan pergerakan nantinya dan yang terakhir, perancangan dan implementasi Gerak *Quadcopter* yang akan membahas mengenai bagaimana cara membuat *quadcopter* dapat bergerak sesuai dengan yang kita inginkan.



Gambar 5. 1 Alur Perancangan Sistem

5.1 Komunikasi Sistem

5.1.1 Perancangan Komunikasi Sistem

Tahap awal yang dilakukan ialah pembuatan kode program dengan menggunakan bahasa pemrograman python dimana pada kode program tersebut terdapat fungsi *ReadData* yang berfungsi sebagai *subscriber* untuk mengambil data yang terdapat pada *driver autonomy_drone* agar dapat diakses oleh pengguna, Fungsi *Angular* untuk mengolah nilai dari parameter *angular* pada *quadcopter* sehingga dapat menghasilkan suatu data perintah atau intruksi yang diproses oleh *setCommand* untuk menentukan gerak dari *quadcopter* itu sendiri. Kode program yang telah dibuat akan diubah dalam bentuk *node* agar dapat berjalan pada ROS yang kemudian akan dikirimkan pada *quadcopter* dengan menggunakan komunikasi Wi-Fi 802.11 b/g/n. Untuk lebih jelasnya dapat dilihat pada Gambar 5.2.

Gambar 5. 2 Skema Komunikasi Sistem

5.1.2 Implementasi Komunikasi Sistem

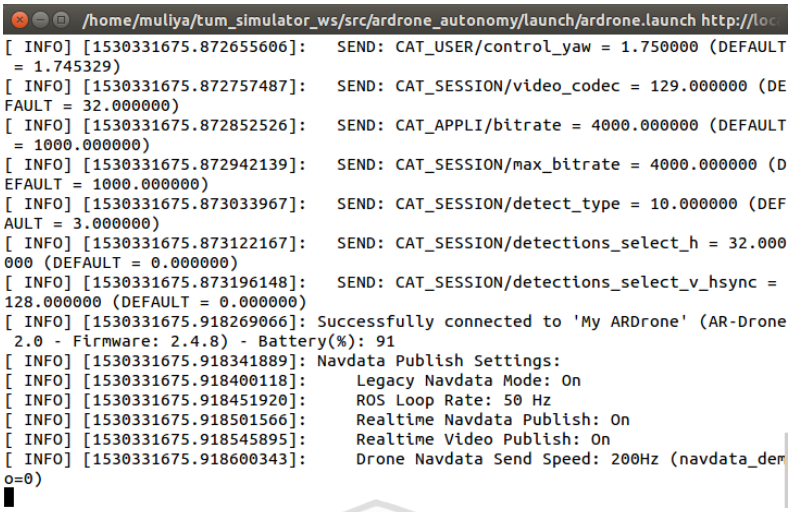
Bagian ini menjelaskan tentag alur implementasi komunikasi sistem pada penelitian ini. Pertama komputer harus terhubung dengan Wi-Fi *quadcopter* terlebih dahulu seperti yang ditunjukan pada Gambar 5.3.

Gambar 5. 3 Koneksi Wi-Fi *AR.Drone*

Setelah terhubung dengan Wi-Fi, selanjutnya adalah menghubungkan *quadcopter* dengan ROS agar pengguna dapat mengkses data navigasi AR.Drone. adapun cara untuk menghubungkannya dengan membuka terminal baru pada *Ubuntu* kemudian masuk pada *packages $ tum_simulator_ws* dan mengetikkan *command* seperti dibawah ini

```
$ roslaunch ardrone_autonomy ardrone.launch
```

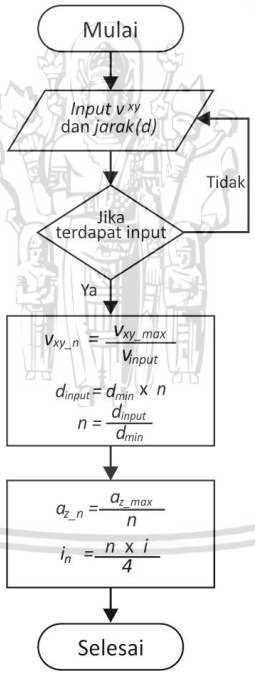
Kemudian jika *quadcopter* telah terhubung dengan ROS maka pada terminal akan muncul informasi mengenai alamat IP AR.Drone yang telah terhubung dan siap digunakan beserta informasi mengenai baterai dan informasi navdata yang memungkinkan untuk dapat di akses seperti pada Gambar 5.4.



Gambar 5. 4 Tampilan Saat Komputer Terhubung Dengan *Quadcopter*

5.2 Fungsi *Angular*

5.2.1 Perancangan Fungsi *Angular*



Gambar 5. 5 Alur Perhitungan Persamaan Fungsi *Angular*

Terihat pada Gambar 5.5 bahwa nilai *input* yang diberikan sangat berpengaruh pada proses yang akan dilakukan berikutnya. Setelah kedua nilai *input* didapatkan selanjutnya menghitung nilai bagi *linear velocity* (v_{xy_n}) untuk menentukan jumlah iterasi yang dilakukan. Yang dilanjutkan dengan menghitung nilai n untuk menentukan jumlah perpindahan jarak yang akan ditempuh. Kemudian nilai n yang dihasilkan akan diolah atau diperhitungkan dalam menentukan nilai dari orientasi gerak pada $angular_z$

(a_z) untuk menentukan besar perputran dari *quadcopter*. Selain itu, nilai n juga mempengaruhi dalam proses perhitungan *iterasi* yang akan dikalikan dengan hasil dari nilai bagi kecepatan v_{xy_n} untuk menentukan berapa banyak *quadcopter* harus berjalan sesuai dengan lintasannya. Sebagai contoh apabila nilai input yang diberikan sebesar 0.8 untuk nilai kecepatan v_{xy_input} dan 2 (dalam satuan meter) nilai untuk jarak (d) dengan nilai orientasi gerak pada $angular_z$ (a_z) sebesar 1, nilai *linear velocity* (v_{xy}) sebesar 1, nilai *iterasi* (i) sebesar 40 dan jarak (d_{min}) 2 meter maka, sesuai dengan persamaan 2.1 dan 2.2 didapatkan hasil nilai bagi kecepatan v_{xy_n} dan nilai n adalah:

$$v_{xy_n} = \frac{1}{0.8} = 1.25$$

$$n = \frac{2}{2} = 1$$

Setelah kedua nilai tersebit di dapatkan, langkah selanjutnya yaitu menghitung nilai a_{z_n} untuk menentukan seberapa besar perputaran sudut yang dilakukan *quadcopter* dan menghitung nilai (i) untuk menentukan jumlah pergerakan yang akan dilakukan oleh *quadcopter* dalam melakukan gerak *angular* dengan lintasan ¼ lingkaran. Dengan mengacu pada persamaan 2.3 dan 2.4 didapatkan nilai a_{z_n} dan i adalah:

$$a_{z_n} = \frac{1}{1} = 1$$

$$i_n = \frac{(1 \times 40) \times 1.25}{4}$$

$$= \frac{50}{4} = 12.5 \text{ dibulatkan menjadi } 13$$

5.2.2 Implementasi Fungsi *Angular*

Perhitungan untuk mencari nilai fungsi *angular* pada penelitian ini menggunakan tiga parameter yaitu parameter $angular_z$ (), $linear_x$ () dan $linear_y$(). Jumlah *iterasi quadcopter* untuk melakukan gerak *angular* dengan lintasan satu lingkaran penuh serta diameter atau jarak minimum yang diperoleh dari gerak *angular* yang dilakukan. Dimana nilai-nilai tersebut akan digunakan sebagai nilai acuan untuk membuat persamaan fungsi *angular*. Nilai 1 pada variable v_xy_max dan a_z_max pada Kode Program 5.1 merupakan nilai maksimal dari range parameter *linear velocity* dan parameter orientasi gerak r_z . Nilai 2 pada variable jarak_min merupakan nilai diameter yang didapatkan dari gerak dengan lintasan satu lingkaran penuh. Dan yang terakhir nilai 22 pada variable i digunakan untuk menentukan jumlah langkah yang harus dilakukan oleh *quadcopter*

Kode Program 5. 1 Inisialisasi Variabel

```
1  if __name__ == '__main__':
2      uav = AutonomousFlight()
3       v_xy_max = 1
4       a_z_max = 1
5       jarak_min = 2
6       i = 22 #1_lingkaran
```

Setelah inisialisasi variabel, langkah selanjutnya yaitu pembuatan kode program untuk memproses nilai *input* dan menghitung nilai inputan menggunakan persamaan fungsi *angular* untuk menghasilkan suatu gerakan yang sesuai. Untuk lebih jelasnya dapat dilihat pada Kode Program 5.2, *syntax* pada baris ke-1 dan baris ke-2 digunakan untuk memanggil nilai kecepatan v_{xy} dan nilai jarak d yang diberikan oleh pengguna. *Syntax* pada baris ke-5 digunakan untuk mencari nilai bagi kecepatan v_{xy}, *syntax* pada baris ke-6 digunakan untuk mencari besar nilai n dan *syntax* pada baris ke-7 digunakan untuk mencari besar nilai orientasi gerak $angular_Z$. Sedangkan pada *syntax* baris ke-8 hingga baris ke-9 digunkan untuk mencari nilai *iterasi* yang harus dilakukan oleh *quadcopter* dalam melakukan gerak dengan lintasan ¼ lingkaran.

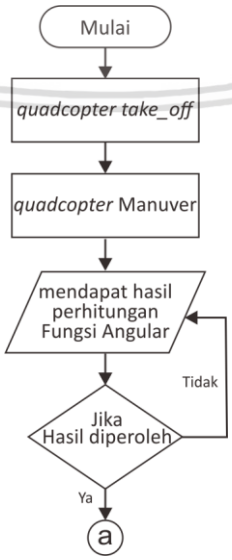
Kode Program 5. 2 Perhitungan Fungsi *Angular*

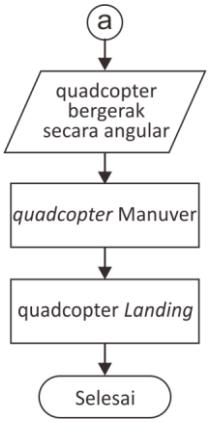
```
1  def hitungdulu(self):
2          uav.inputan1()#nilai linear velocity
3          uav.inputan2()#nilai jarak
4
5          self. v_xy_n = v_xy_max / uav.v_input #input1
6          self.n = self.jarak_n #input2 / jarak_min
7          self.a_z_n = a_z_max / uav.n
8          self.i_n = (i * uav.n)/4
9          self.jalan = uav.i_n * uav. v_xy_n
```

5.3 Gerak *Quadcopter*

5.3.1 Perancangan Gerak *Quadcopter*

Setelah mendapatkan nilai dari hasil perhitungan dengan menggunakan persamaan fungsi *angular* selanjutnya yaitu mengirimkan nilai hasil perhitungan kepada *setCommand* untuk dijadikan sebuah perintah yang mengatur gerak dari *quadcopter*. *Quadcopter* akan bergerak secara *angular* sesuai dengan *input* yang diberikan. Untuk lebih jelasnya dapat dilihat pada diagam alir perancangan gerak *quadcopter* pada Gambar5.3





Gambar 5. 6 Diagram Alir Perancangan Pergerakan *Quadcopter*

Sebelum *quadcopter* terbang dan melakukan berbagai macam gerak *quadcopter* akan melakukan proses lepa *land*as atau *take-off* terlebih dahulu. Proses *take-off* seperti yang telah dijelaskan pada bab-bab sebelumnya akan terjadi apabila pada AR.Drone. Sedangkan untuk melakukan gerak maneuver pada satu koordinat sumbu *quadcopter* pesan yang akan dikirimkan oleh ROS memiliki tipe *geometry_msgs :: Twist* dimana pengguna dapat memberikan nilai sesuai yang diinginkan pada setiap parameter yang dimiliki oleh *quadcopter*. Sama halnya dengan gerak *maneuver,* setelah *quadcopter* mendapatkan nilai hasil dari perhitungan fungsi *angular* maka data tersebut akan dijadikan suatu intruksi atau data perintah *geometry_msgs :: Twist* sebelum dikirimkan pada *package ardrone*. Setelah gerak *angular* selesai dilakukan, *quadcopter* akan melakukan gerak maneuver untuk beberapa saat hingga pengguna dapat memberikan perintah *landing* pada *quadcopter*. berbeda dengan gerak *angular* dan mauver, *quadcopter* dapat melakukan *landing* apabila ROS telah mengirimkan suatu pesan std_msgs / Empty pada package ardrone/*land*.

5.3.2 Implementasi Gerak *Quadcopter*

Setelah prancangan untuk pergerakan *quadcopter* dibuat, langkah selanjutnya adalah pembuatan kode program untuk mengatur gerak dari *qudcopter* baik pada saat *takeoff, maneuver, gerak angular* mapun *landing*. Langkah awal pembuautan kode program untuk mengatur gerak dari *quadcopter* adalah inisialisasi untuk parameter yang akan digunakan dalam melakukan berbagai gerak . Seperti Baris ke-1 pada Kode Program 5. 3 menjelaskan bahwa pada penelitian ini hanya membutuhkan tiga parameter yang mempengaruhi gerak qudcopter yaitu $linear_x$, $linear_y$ *dan* $angular_z$. Snytax pada baris ke-2 hinngga baris ke-4 merupakan inisialisasi koordinat sumbu x, y dan z dengan menggunakan nama varieabel yang sama dengan parameter yang digunakan agar nilai yang diberikkan dapat di akses.

Kode Program 5. 3 Inisialisasi Parameter Gerak *Quadcopter*

```
1  def SetCommand(self, linear_x, linier_y, angular_z):
2          self.command.linear.x = v_x
3          self.command.linear.y = v_y
4          self.command.angular.z = a_z
```

```
5           self.pubCommand.publish(self.command)
```

Setelah itu, kode program untuk melakukan gerak *take-off* dan *landing* dapat diliat pada Kode Program 5.4. Pada Kode Program 5.4 *syntax* pada baris ke-2 dan baris ke-10 berfungsi untuk mempertahankan perulangan pada setiap 10hz dengan memperhitungkan waktu yang digunakan setiap operasinya. Untuk memerintahkan *quadcopter* melakukan gerak *take-off* digunakan *syntax* pada baris ke-1 hingga baris ke-7 dimana perintah tersebut merupakan untuk mengirimkan pesan ardone/*takeoff* pada package ardrone dengan perintah *take-off*. Sama seperti perintah *take-off* untuk melakukan perintah *landing* digunakan *syntax* pada baris ke-9 hingga baris ke-15 yang memiliki perintah untuk mengirimkan pesan ardone/*land* pada package ardrone dengan perintah *landing*. Perintah tersebut digunakan untuk mengirimankan perintah *landing* pada *quadcopter*.

Kode Program 5. 4 Gerak *take-off* dan *landing*

```
1   def SendTakeoff(self):
2       self.rate = rospy.Rate(10)
3       self.pubTakeoff = rospy.Publisher("ardrone/takeoff", Empty,
4       queue_size=10)
5
6       self.pubTakeoff.publish(Empty())
7       self.rate.sleep()
8
9   def SendLand(self):
10      self.rate = rospy.Rate(10)
11      self.pubLand = rospy.Publisher("ardrone/land", Empty,
12      queue_size=10)
13
14      rospy.init_node('land', anonymous=True)
15      self.pubLand.publish(Empty())
16      self.rate.sleep()
```

Pergerakan manuver pada dua sumbu, yaitu sumbu x dan y. dapat dilihat pada Kode Program 5.5 terdapat dua fungsi ‘jalan’ dimana pada setiap fungsi terdapat fungsi *setCommand* dan tiga parameter yang digunakan untuk memerintahkan gerak dari *quadcopter.* fungsi ‘jalan1’ pada baris ke-1 akan dipanggil atau dijalankan setelah *quadcopter* melakukan *take-off* sedangkan fungsi ‘jalan2’ pada baris ke-9 akan dijalankan setelah *quadcopter* melakukan gerak *angular.* Untuk membuat *quadcopter* bermanuver pada sumbu x (bergerak maju atau mundur) maka pada parameter ke-1 diberi nilai 1 dan kedua parameternya bernilai 0, jika *quadcopter* akan melaukan manuver pada sumbu y (bergerak kanan dan kiri) maka pada parameter ke-2 diberi nilai 1 dan yang lain bernilai 0. Arah gerak *quadcopter* ditentukan dari positif(+) negatif(-) nilai dari parameter tersebut. *Syntax* pada baris ke-16 hingga baris ke-18 digunakan untuk menampilkan perintah ‘stop’ yang akan mengakibatkan *quadcopter* melakukan proses *landing* yang mengarah pada baris ke-19 apabila mendapatkan perintah dari pengguna. Baris ke-19 mengakibatkan *quadcopter* melakukan *landing* dikarenakan pemangilan fungsi *landing* yang telah dibuat pada Kodeprogram 5.3 pada baris ke-9.

Kode Program 5. 5 Gerak Manuver

```
1  def jalan1(self):
2        jalan_1 = 0
3        while jalan_1 <= 2:
4              uav.SetCommand(1, 0, 0)
5              jalan_1 += 1
6              if jalan_1 > 2:
7                 uav.hitungdulu()
8
9  def jalan2(self):
10       jalan_2 = 0
11       while jalan_2 <= 5:
12             uav.SetCommand(1, 0, 0)
13             jalan_2 += 1
14             if jalan_2 > 5:
15                uav.SetCommand(0, 0, 0)
16                print "tekan '1' untuk stop"
17                stop = input(': ')
18                if stop == 1:
19                   uav.SendLand()
```

Selanjutnya, pembuatan kode program untuk gerak *angular* itu sendiri. Agar *quadcopter* dapat bergerak secara *angular* maka ketiga nilai parameter harus diberi nilai. Nilai yang diberikan kepada ketiga parmeter tesebut tidak di isikan secara manual, akan tetapi akan secara otomatis terisi dengan nilai yang didapatkan dari perhitungan fungsi *angular* pada bab sebelumnya. Kode Program 5.6 merupakan kode program yang digunakan untuk mengatur gerak dari *quadcopter* itu sendiri. *syntax*.pada baris ke-1, 9 dan 10 berfungsi untu menghitung waktu yang dibutuhkan untuk melakukan gerak melingkar. Sedangkan *syntax* yang lainya berfungsi sebagai pengatur gerak *quadcopter*. *Syntax* pada baris ke-3 merupakan peran utama dari kode program tersebut karena berfungsi untuk menentukan gerak dari *quadcopter*. pada Kode Program 5.6 *syntax* baris ke-3 membuat *quadcopter* bergerak maju dengan lintasan melingkar keaarah kiri. *Quadcopter* dapat melakuakn gerak maju dengan lintasan melingkar kearah kiri karena parameter ke-1 memiliki nilai positif (+) yang berarti gerak maju dan parameter ke-3 bernilai positif (+) yang berarti gerak memutar kekiri. Selain itu, *quadcopter* akan bergerak maju atau mundur apabila parameter ke-2 bernilai 0.

Kode Program 5. 6 Gerak *Angular Quadcopter* Pada Sumbu x

```
1  while gerak <= round(uav.jalan):
2      waktu_awal =  time.clock()
3      uav.SetCommand (uav.kecepatan_n, 0, uav.angular_n)
4      gerak +=1
5
6      if gerak > round(uav.jalan):
7          uav.SetCommand (0,0,0)
8
9  waktu_akhir = time.clock()
10 waktu_total = waktu_akhir - waktu_awal
```

Untuk membuat gerak melingkar pada sumbu x dengan arah yang berbeda-beda pengguana dapat merubah nilai dari setCommand seperti pada

Kodeprogram 5.7. Snytax pada baris ke-1 hingga baris ke-3 berfungsi untuk menentukan arah gerak dari *quadcopter* dalam koordinat usmbu x.

Kode Program 5. 7 Penentuan Arah Gerak *Angular Quadcopter* pada Sumbu x

```
1  uav.SetCommand (uav.kecepatan_n, 0, -uav.angular_n) #Maju - kanan
2  uav.SetCommand (-uav.kecepatan_n, 0, uav.angular_n) #Mundur - kanan
3  uav.SetCommand (-uav.kecepatan_n, 0, -uav.angular_n) # Mundur -kiri
```

Sebagai contoh, *quadcopter* akan melakukan gerak *angular* pada koordinat sumbu x dengan nilai `uav.SetCommand (0.8, 0, -1)`. Yang berarti *quadcopter* akan melakukan gerak *angular* maju dengan arah lintasan kekanan dengan keceptan v_{xy} 0.8 dan besar nilai orientasi pada $angular_z$ adalah 1 akan menghasilkan gerak yang ditunjukkan pada Gambar 5.7.

Gambar 5. 7 Gerak *Angular* Maju dengan lintasan Kekanan

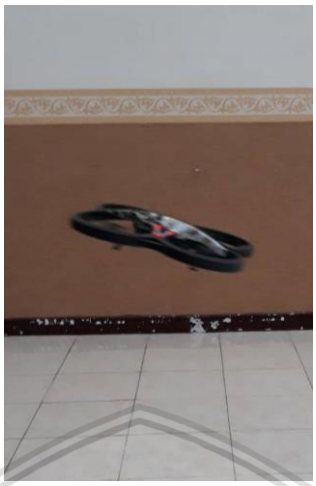
Sama halnya dengan Kode Program 5.7 untuk membuat gerak *angular* ke kanan dan ke kiri dengan arah yang berbeda dapat dilihat pada kode program 5.8. *Syntax* baris ke-1 hingga baris ke-4 berfungsi untuk menentukan apakah *quadcopter* akan bergerak pada ke kanan dan ke kiri dengan berbagai arah lintasan. Dimana arah gerak dari *quadcopter* juga ditentukan okeh positif(+) dan negatif(-) nilai yang diberikan.

Kode Program 5. 8 Penentuan Arah Gerak *Angular Qudcopter* Pada Sumbu y

```
1  uav.SetCommand (0, uav.kecepatan_n, uav.angular_n) #Kiri – kiri
2  uav.SetCommand (0, uav.kecepatan_n, -uav.angular_n) #Kiri –kanan
3  uav.SetCommand (0, -uav.kecepatan_n, uav.angular_n) #Kanan – kanan
4  uav.SetCommand (0, -uav.kecepatan_n, -uav.angular_n) #Kanan –kiri
```

Sebagai contoh, *quadcopter* akan melakukan gerak *angular* pada koordinat sumbu y dengan nilai `uav.SetCommand (0, 1, -0.5)`. Yang berarti *quadcopter* akan melakukan gerak *angular* kiri dengan arah lintasan kekanan dengan

keceptan v_{xy} 1 dan besar nilai orientasi pada $angular_z$ adalah 0.5 akan menghasilkan gerak yang ditunjukkan pada Gambar 5.8



Gambar 5. 8 Gerak *Angular* Kiri dengan lintasan Kekanan

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini akan membahas mengenai hasil pengujian sistem yang telah dilakukan serta analisisnya. Ada beberapa pengujian yang dilakukan dalam penelitian ini yaitu, pengujian untuk menentukan gerak *angular* pada *quadcopter* dengan lintasan satu lingkaran penuh, pengujian ketepatan *quadcopter* dalam melakukan gerak *angular* dengan kecepatan yang berbeda - beda dan pengujian untuk menentukan estimasi waktu yang dibutuhkan untuk melakukan pergerakan *angular*.

6.1 Pengujian Gerak *Angular*

6.1.1 Tujuan Pengujian

Pengujian ini bertujuan untuk menentukan berapa saja nilai parameter yang dibutuhkan agar *quadcopter* dapat melakukan gerak *angular*.

6.1.2 Pelaksanaan Pengujian

Pengujian ini dilakuakan dengan memberi nilai pada pada parameter $angular_z$ dan salah satu parameter $linear_x$ atau $linear_y$ untuk menentukan gerak dari *quadcopter*. Pengujian dilakukan dengan memberikan nilai maksimum pada salah satu parameter $linear_x$ atau $linear_y$ (v_{xy}) dan parameter $angular_z$ (a_z). Selain itu, juga memberikan nilai iterasi (i) untuk mengatur berapa banyak gerak yang dilakukan oleh *quadcopter* untuk mencapai satu lingkaran. Pengujian ini dilakukan hingga menemukan nilai i yang tepat untuk *quadcopter* melakukan gerak *angular* dengan lintasan satu lingkaran penuh.

6.1.3 Prosedur Pengujian

Pengujian ini dilakukan sesuai dengan prosedur pengujian berikut:

1. Pasang baterai *quadcopter* yang sudah terisi penuh. *quadcopter* dapat digunakan setelah *quadcopter* terkalibrasi atau drone mengeluarkan bunyi beep sebanyak 4 kali.
2. Menghubungkan komputer dengan Wi-Fi yang terdapat pada *quadcopter*
3. Memberikan nilai iterasi i untuk menentukan berapa kali *quadcopter* akan berjalan.
4. Buka terminal pada *Ubuntu* ketikkan *command "$ cd tum_simulator_ws"* kemudian ketikkan *"$ source devel/setup.bash"*, dan yang terakhir ketikkan *command "$ roslaunch ardrone_autonomy ardrone.launch*" agar *quadcopter* terhubung dengan ROS.
5. Buka terminal baru *command "$ cd tum_simulator_ws"* kemudian ketikkan *"$ source devel/setup.bash"*, dan untuk menjalankan kode program ketikan *"$ rosrun sekrip baru.py".*

6. Amati pergerakan yang dilakukan oleh *quadcopter.* Apakah *quadcopter* dapat menghasilkan gerak *aangular* dengan lintasan satu lingkaran penuh atau tidak.
7. Ulang langkah ke-5 dengan memberikan nilai i yang berbeda - beda apabila gerak dengan lintasan satu lingkaran penuh belum terpenuhi.

6.1.4 Hasil Pengujian

Setelah pengujian dilakukan, didapatkan nilai iterasi (i) untuk gerak *quadcopter* menempuh satu lingkaran penuh ditampilkan pada Tabel 6.1.

Tabel 6. 1 Hasil Pengujian Gerak *Angular* Satu Lingkaran Penuh

v_{xy}	a_z	i	Hasil Gerak	Status
1	1	10	½ lingkaran	Gagal
1	1	15	¾ lingkaran	Gagal
1	1	20	1 ¾ lingkaran	Gagal
1	1	22	Satu lingkaran penuh	Berhasil

6.1.5 Analisis Pengujian

Dari pengujian yang telah dilakukan keberhasilan gerak *quadcopter* ditentukan dari hasil pergerakan yang dilakukan. Status pergerakan *quadcopter* dapat dikatakan berhasil apabila gerak yang dihasilkan memiliki lintasan satu lingkaran penuh, jika gerak *quadcopter* tidak sesuai maka status dari gerak *quadcopter* dikatakan gagal. Didapatkan hasil dari pengujian bahwa dengan nilai v_{xy} dan nilai a_z bernilai maksimal (1) didapatkan hasil bahwa gerak *angular* pada *quadcopter* dengan lintasan satu lingkaran penuh dapat dilakukan apabila iterasi (i) bernilai 22. Selain itu, dari pergerakan tersebut *quadcopter* juga menghasilkan nilai diameter minimum sebesar 2 meter. Hasil dari data pengujian pada Tabel 6.1 akan digunakan sebagai acuan dalam menghitung persamaan fungsi *angular* 2.1 hingga 2.4.

6.2 Pengujian Ketepatan

6.2.1 Tujuan Pengujian

Pengujian ini dilakukan untuk mengetahui apakah dengan menggunaan persamaan fungsi *angular quadcopter* dapat bergerak dengan lintasan ¼ lingkaran seperti yang diinginkan dengan kecepatan yang berbeda-beda.

6.2.2 Pelaksanaan Pengujian

Pengujian gerak *angular* dilakukan berdasarkan koordinat sumbu x dan y. Pada koordinat sumbu x (*pitch*) akan dilakukan gerak ke depan dan belakang dengan arah kekanan dan kekiri. Sedangkan pada kordinat sumbu y (*roll*) dilakukan gerak kanan dan kiri dengan arah kekanan dan kekiri. Untuk setiap

pengujian gerakan *quadcopter* akan dilakukan dengan berbagai nilai kecepatan. Dimana untuk mengatur nilai kecepatan dari laju *quadcopter* dengan cara memberikan nilai pada parameter *linear velocity* (v_{xy}) dengan range nilai 0 hingga 1.

6.2.3 Prosedur Pengujian

Pengujian ini dilakukan sesuai dengan prosedur pengujian berikut:

1. Pasang baterai *quadcopter* yang sudah terisi penuh. *quadcopter* dapat digunakan setelah *quadcopter* terkalibrasi atau drone mengeluarkan bunyi beep sebanyak 4 kali.
2. Menghubungkan komputer dengan Wi-Fi yang terdapat pada *quadcopter*
3. Buka terminal pada *Ubuntu* ketikkan *command "$ cd tum_simulator_ws"* kemudian ketikkan *"$ source devel/setup.bash"*, dan yang terakhir ketikkan *command "$ roslaunch ardrone_autonomy ardrone.launch*" agar *quadcopter* terhubung dengan ROS.
4. Buka terminal baru *command "$ cd tum_simulator_ws"* kemudian ketikkan *"$ source devel/setup.bash"*, dan untuk menjalankan kode program ketikan *"$ rosrun sekrip demo.py".*
5. Pengguna akan menerima perintah untuk memasukkan nilai kecepatan laju *quadcopter*. Sebagai contoh, pengguna memberikan masukan nilai dari yang terendah hingga tertinggi (dengan range nilai 0 hingga 1).
6. Amati pergerakan yang dilakukan oleh *quadcopter* apakah *quadcopter* dapat bergerak dengan lintasan ¼ lingkaran.
7. Data *navigasi* yang ditampilkan pada terminal akan disimpan di *libre office* dalam bentuk xls.
8. Ulang langkah ke-5 sebnyak lima kali dan catat apakah jarak (r) yang dihasilkan oleh *quadcopter* dengan kecepatan yang berbeda - beda dapat bergerak sesuai dengan yang diberikan.

6.2.4 Hasil Pengujian

6.2.4.1 Gerakan Pada Sumbu x

1. Gerak Maju – Kiri

   Pada gerak ini *quadcopter* bergerak maju secara *angular* dengan arah lintasan kekiri. Dari pengujian gerak tersebut didapatkan hasil seperti pada Tabel 6.2. Dimana data yang ditampilkan pada Tabel merupakan nilai rata-rata jarak (r) yang dapat ditempuh oleh *quadcopter* dari lima kali percobaan yang telah dilakukan dengan kecepatan yang berbeda

Tabel 6. 2 Jarak Gerak *Angular* Maju - Kiri

Nilai v_{xy}	Jari – jari lintasan dalam centi meter (cm)					Rata-rata jari-jari lintasan (cm)	Bentuk lintasan
	Pengujian ke-						
	1	2	3	4	5		
1	104	100	94	98	99	99	¼ Lingkaran
0,9	84	94	112	93	104	97,4	¼ Lingkaran
0,8	102	94	88	92	96	94,4	¼ Lingkaran
0,7	92	89	84	82	97	88,8	¼ Lingkaran
0,6	92	115	88	114	78	97,4	½ Lingkaran
0,5	112	93	112	73	82	94,4	¾ Lingkaran
0,4	96	92	96	112	86	96,4	¾ Lingkaran
0,3	106	102	88	82	83	92,2	1¾ Lingkaran
0,2	105	105	83	65	83	88,2	Lingkaran
0,1	65	78	82	88	82	79	Lingkaran

2. Gerak Maju – Kanan

Demikian juga pada gerak ini, *quadcopter* bergerak maju secara *angular* dengan arah lintasan kekanan dengan kecepatan v_{xy} yang berbeda beda. Dari pengujian gerak tersebut didapatkan nilai rata-rata jarak(r) seperti pada Tabel 6.3 pada saat melakukan gerak *angular*.

Tabel 6. 3 Jarak Gerak *Angular* Maju - Kanan

Nilai v_{xy}	Jari – jari lintasan dalam centi meter (cm)					Rata-rata jari-jari lintasan (cm)	Bentuk lintasan
	Pengujian ke-						
	1	2	3	4	5		
1	93	120	94	91	98	99,2	¼ Lingkaran
0,9	102	84	104	77	93	92	¼ Lingkaran
0,8	94	104	95	93	96	96,4	¼ Lingkaran
0,7	82	89	79	105	92	89,4	¼ Lingkaran
0,6	92	88	115	88	98	96,2	½ Lingkaran
0,5	86	82	116	108	73	93	¾ Lingkaran
0,4	78	82	80	96	92	85,6	¾ Lingkaran

0,3	88	82	93	106	86	91	1¾ Lingkaran
0,2	83	88	86	102	83	88,4	Lingkaran
0,1	88	82	82	68	86	81,2	Lingkaran

3. Gerak Mundur – Kanan

Pada gerak ini *quadcopter* bergerak mundur secara *angular* dengan arah lintasan kekanan. Sama denga pengujian yang dilakukan sebelumnya. Pengujian ini juga menmpilkan nilai jarak (r) *quadcopter*, dimana pada kelima hasil percobaan tersebut akan di cari nilai rata-rata jarak(r) yang paling tepat dengan kecepatan v_{xy} yang optimal. Adapun hasil data pengujian pada sistem ini dituliskan pada Tabel 6.4.

Tabel 6. 4 Jarak Gerak *Angular* Mundur - Kanan

Nilai v_{xy}	Jari – jari lintasan dalam centi meter (cm)					Rata-rata jari-jari lintasan (cm)	Bentuk lintasan
	Pengujian ke-						
	1	2	3	4	5		
1	120	88	98	99	98	100,6	¼ Lingkaran
0,9	100	135	84	82	94	99	¼ Lingkaran
0,8	120	97	105	86	83	98,2	¼ Lingkaran
0,7	105	89	92	79	82	89,4	¼ Lingkaran
0,6	92	98	88	113	89	96	½ Lingkaran
0,5	108	112	93	86	96	103,8	¾ Lingkaran
0,4	105	115	96	78	102	99,2	¾ Lingkaran
0,3	105	83	86	102	96	94,4	1¾ Lingkaran
0,2	83	65	83	92	79	80,4	Lingkaran
0,1	40	88	78	83	84	74.6	Lingkaran

4. Gerak Mundur – Kiri

Pada gerak ini *quadcopter* bergerak mundur secara *angular* dengan arah lintasan kekanan. Dengan nilai kecepatan v_{xy} yang berbeda-beda di dapatkan rata-rata nilai jarak(r) seperti pada Tabel 6.5.

Tabel 6. 5 Jarak Gerak *Angular* Mundur - Kiri

Nilai v_{xy}	Jari – jari lintasan dalam centi meter (cm)					Rata-rata jari-jari lintasan (cm)	Bentuk lintasan
	Pengujian ke-						
	1	2	3	4	5		
1	102	115	98	95	99	101.8	¼ Lingkaran
0,9	120	104	84	107	97	102,4	¼ Lingkaran
0,8	104	97	99	90	94	96,8	¼ Lingkaran
0,7	124	92	127	120	105	113,6	¼ Lingkaran
0,6	115	113	98	92	88	101,2	½ Lingkaran
0,5	116	112	120	106	100	110,8	¾ Lingkaran
0,4	112	96	120	96	115	107.8	¾ Lingkaran
0,3	86	88	102	83	105	92.8	1¾ Lingkaran
0,2	83	65	93	88	86	83	Lingkaran
0,1	82	67	84	78	85	79,2	Lingkaran

6.2.4.2 Gerakan Pada Sumbu y

1. Gerak Kanan – Kanan

   Pada gerak ini *quadcopter* bergerak kesamping kanan secara *angular* dengan arah lintasan kekanan. Sama halnya dengan gerak pada koordinat sumbu x, pengujian gerak pada koordinat sumbu y juga dilakukan untuk mencaritahu berapa nilai kecepatan yang optimal untuk gerak ini dan apakah nilai jarak (r) yang ditempuh sesuai dengan yang di berikan. Dari pengujian gerak tersebut didapatkan hasil seperti pada Tabel 6.6.

Tabel 6. 6 Jarak Gerak *Angular* Kanan - Kanan

Nilai v_{xy}	Jari – jari lintasan dalam centi meter (cm)					Rata-rata jari-jari lintasan (cm)	Bentuk lintasan
	Pengujian ke-						
	1	2	3	4	5		
1	115	105	98	95	94	101,4	¼ Lingkaran
0,9	98	115	105	97	112	105,4	¼ Lingkaran
0,8	94	100	92	97	106	97,8	¼ Lingkaran
0,7	108	110	97	89	88	98,4	¼ Lingkaran
0,6	98	92	115	109	94	101,6	½ Lingkaran

0,5	110	98	88	82	92	94	¾ Lingkaran
0,4	125	115	107	94	83	104,8	¾ Lingkaran
0,3	120	105	88	102	93	101,6	1¾ Lingkaran
0,2	115	108	93	98	101	103	Lingkaran
0,1	60	67	85	82	88	76.4	Lingkaran

2. Gerak Kanan – Kiri

Pada gerak ini *quadcopter* bergerak kesamping kanan secara *angular* dengan arah lintasan kekanan. Hasil dari gerak yang dilakukan ditunjukkan oleh Tabel 6.7 meunjukkan rata-rata jarak(r) yang berhasil di tempuh oleh *quadcopter* dengan kecepatan yang berbeda.

Tabel 6. 7 Jarak Gerak *Angular* Kanan - Kiri

Nilai v_{xy}	Jari – jari lintasan dalam centi meter (cm)					Rata-rata jari-jari lintasan (cm)	Bentuk lintasan
	Pengujian ke-						
	1	2	3	4	5		
1	120	88	95	98	104	101	¼ Lingkaran
0,9	98	105	115	94	84	99,2	¼ Lingkaran
0,8	86	97	88	104	99	94,8	¼ Lingkaran
0,7	105	92	110	82	84	94,6	¼ Lingkaran
0,6	115	98	88	92	88	96,2	½ Lingkaran
0,5	108	112	93	86	105	108	¾ Lingkaran
0,4	92	112	96	105	107	102,4	¾ Lingkaran
0,3	88	106	101	97	93	97	1¾ Lingkaran
0,2	82	93	108	113	86	96,4	Lingkaran
0,1	85	78	83	69	87	80,4	Lingkaran

3. Gerak Kiri – Kanan

Tabel 6.8 adalah hasil dari pengujian gerak *quadcopter* kesamping kanan secara *angular* dengan arah lintasan kekanan. Dari gerak tersebut didapatkan rata-rata jarak(r) yang dapat ditempuh oleh *quadcopter* dengan kecepatan yang berbeda beda.

Tabel 6. 8 Jarak Gerak *Angular* Kiri - Kanan

Nilai v_{xy}	Jari – jari lintasan dalam centi meter (cm)					Rata-rata jari-jari lintasan (cm)	Bentuk lintasan
	Pengujian ke-						
	1	2	3	4	5		
1	88	98	95	120	105	101,2	¼ Lingkaran
0,9	102	107	94	84	104	98,2	¼ Lingkaran
0,8	90	99	120	97	88	98,8	¼ Lingkaran
0,7	105	115	97	92	89	99,6	¼ Lingkaran
0,6	88	120	113	97	92	102	½ Lingkaran
0,5	100	93	73	86	114	93,2	¾ Lingkaran
0,4	92	80	84	98	103	91,4	¾ Lingkaran
0,3	88	106	93	102	95	96,8	1¾ Lingkaran
0,2	65	83	86	97	93	85,8	Lingkaran
0,1	85	77	88	82	79	82,2	Lingkaran

4. Gerak Kiri – Kiri

Yang terakhir adalah gerak *quadcopter* kesamping kiri secara *angular* dengan arah lintasan kekiri. Hasil dari gerak tersebut ditunjukkan pada Tabel 6.9 dimana pada Tabel terdapat nialai dari jarak(r) yang ditempuh oleh *quadcopter* dengan kecepatan yang berbeda-beda pada saat melakukan gerak *angular*.

Tabel 6. 9 Jarak Gerak *Angular* Kiri - Kiri

Nilai v_{xy}	Jari – jari lintasan dalam centi meter (cm)					Rata-rata jari-jari lintasan (cm)	Bentuk lintasan
	Pengujian ke-						
	1	2	3	4	5		
1	115	93	98	94	101	100,2	¼ Lingkaran
0,9	105	104	97	96	89	98,2	¼ Lingkaran
0,8	97	104	94	117	94	101,2	¼ Lingkaran
0,7	127	120	92	108	110	111,4	¼ Lingkaran
0,6	115	98	88	112	108	104,2	½ Lingkaran
0,5	108	102	99	94	82	97	¾ Lingkaran
0,4	96	96	82	93	112	95,8	¾ Lingkaran

0,3	102	86	83	88	93	90,4	1¾ Lingkaran
0,2	65	93	83	65	86	78,4	Lingkaran
0,1	85	65	82	82	88	80,4	Lingkaran

6.2.5 Analisis Pengujian

Setelah melakukan pengujian ketepatan gerak sistem berdasarkan nilai kecepatan dan jari-jari yang diperoleh, didapatkan hasil bahwa sistem dapat melakukan pergerakan *angular* dengan lintasan ¼ lingkaran dan dengan jarak 100 cm adalah pada kecepatan v_{xy} 0,7 hingga 1. Dengan menggunakan rumus persentase eror pada 6.1 didapatkan hasil dari besar nilai eror sistem saat melakukan gerak *angular* pada berbagai arah dengan kecepatan v_{xy} yang berbeda dan untuk menghitung tingkat akurasi sistem dapat menggunakan rumus pada 6.2.

$$presentase\ eror = \frac{r\ acuan - r\ lintasan}{r\ acuan} \times 100\% \quad (6.1)$$

$$tingkat\ akurasi = \frac{\sum nilai\ persentase\ eror\ terendah - 100\%}{total\ data} \quad (6.2)$$

6.2.5.1 Analisis Gerakan Pada Sumbu x

1. Gerak Maju – Kiri

   Dari pengujian yang telah dilakukan terhadap gerak maju secara *angular* dengan arah lintasan ke kiri didapatkan hasil bahwa peresentase eror ketepatan gerakan *quadcopter* terendah adalah 0,01% dengan kecepatan v_{xy} sebesar 1.

Tabel 6. 10 Besar Eror Gerak Maju - Kiri

Nilai v_{xy}	r acuan (cm)	rata-rata r lintasan (cm)	Persentase Eror
1	100	99	$\frac{100-99}{100} \times 100$ = 0,01%
0,9	100	97,4	$\frac{100-97,4}{100} \times 100\%$ = 0,026%
0,8	100	94,4	$\frac{100-94,4}{100} \times 100\%$ = 0,056%
0,7	100	88,8	$\frac{100-88,8}{100} \times 100\%$ = 0,112%

2. Gerak Maju – Kanan

Sedangkan pada pengujian gerak maju secara *angular* dengan arah lintasan ke kanan didapatkan hasil peresentase eror ketepatan gerakan *quadcopter* terendah adalah 0,008% dengan kecepatan v_{xy} sebesar 1.

Tabel 6. 11 Besar Eror Gerak Maju - Kanan

Nilai v_{xy}	r acuan (cm)	rata - rata r lintasan (cm)	Persentase Eror
1	100	99,2	$\frac{100-99,2}{100} \times 100 = 0,008\%$
0,9	100	92	$\frac{100-92}{100} \times 100\% = 0,08\%$
0,8	100	96,4	$\frac{100-96,4}{100} \times 100\% = 0,036\%$
0,7	100	89,4	$\frac{100-89,4}{100} \times 100\% = 0,106\%$

3. Gerak Mundur – Kanan

Berbeda dengan pengujian gerak mundur secara *angular* dengan arah lintasan ke kanan. Pada pengujian ini didapatkan hasil peresentase kegagalan ketepatan gerakan *quadcopter* terendah adalah 0,01% dengan kecepatan v_{xy} sebesar 0,9.

Tabel 6. 12 Besar Eror Gerak Mundur - Kanan

Nilai v_{xy}	r acuan (cm)	rata - rata r lintasan (cm)	Persentase Eror
1	100	100,6	$\frac{100-100,6}{100} \times 100 = 0,06\%$
0,9	100	99	$\frac{100-99}{100} \times 100\% = 0,01\%$
0,8	100	98,2	$\frac{100-98,2}{100} \times 100\% = 0,018\%$

0,7	100	89,4	$\frac{100-89,4}{100} \times 100\% = 0,106\%$

4. Gerak Mundur – Kiri

Pada pengujian gerak mundur secara *angular* dengan arah lintasan ke kanan didapatkan hasil peresentase eror ketepatan gerakan *quadcopter* terendah adalah 0,018% dengan kecepatan v_{xy} sebesar 1.

Tabel 6. 13 Besar Eror Gerak Mundur - Kiri

Nilai v_{xy}	r acuan (cm)	rata - rata r lintasan (cm)	Persentase Eror
1	100	101.8	$\frac{100-101,8}{100} \times 100 = 0,018\%$
0,9	100	102,4	$\frac{100-102,4}{100} \times 100\% = 0,024\%$
0,8	100	96,8	$\frac{100-96,8}{100} \times 100\% = 0,032\%$
0,7	100	113,6	$\frac{100-113,6}{100} \times 100\% = 0,136\%$

6.2.5.2 Analisis Gerakan Pada Sumbu y

1. Gerak Kanan – Kanan

Tidak jauh berbeda dengan pengujian pada sumbu x. Pengujian gerak kesamping kanan secara *angular* dengan arah lintasan ke kanan pada pengujian ini didapatkan hasil persentase eror ketepatan gerakan *quadcopter* terendah adalah 0,014% dengan kecepatan v_{xy} sebesar 1.

Tabel 6. 14 Besar Eror Gerak Kanan - Kanan

Nilai v_{xy}	r acuan (cm)	rata - rata r lintasan (cm)	Persentase Eror
1	100	101,4	$\frac{100-101,4}{100} \times 100 = 0,014\%$
0,9	100	105,4	$\frac{100-105,4}{100} \times 100\% = 0,054\%$

0,8	100	97,8	$\frac{100-97,8}{100} \times 100\% = 0,022\%$
0,7	100	98,4	$\frac{100-98,4}{100} \times 100\% = 0,016\%$

2. Gerak Kanan – Kiri

Sedangkan pada pengujian gerak kesamping kanan secara *angular* dengan arah lintasan ke kanan didapatkan hasil persentase eror ketepatan gerakan *quadcopter* terendah adalah 0,008% dengan kecepatan v_{xy} sebesar 0,9.

Tabel 6. 15 Besar Eror Gerak Kanan - Kiri

Nilai v_{xy}	r acuan (cm)	rata - rata r lintasan (cm)	Persentase Eror
1	100	101	$\frac{100-101}{100} \times 100 = 0,01\%$
0,9	100	99,2	$\frac{100-99,2}{100} \times 100\% = 0,008\%$
0,8	100	94,8	$\frac{100-94,8}{100} \times 100\% = 0,052\%$
0,7	100	94,6	$\frac{100-94,6}{100} \times 100\% = 0,054\%$

3. Gerak Kiri – Kanan

Pada pengujian gerak kesamping kiri secara *angular* dengan arah lintasan ke kanan didapatkan hasil persentase eror ketepatan gerakan *quadcopter* terendah adalah 0,004% dengan kecepatan v_{xy} sebesar 0,7.

Tabel 6. 16 Besar Eror Gerak Kiri - Kanan

Nilai v_{xy}	r acuan (cm)	rata - rata r lintasan (cm)	Persentase Eror
1	100	101,2	$\frac{100-101,2}{100} \times 100 = 0,012\%$

0,9	100	98,2	$\frac{100-98,2}{100} \times 100\% = 0,018\%$
0,8	100	98,8	$\frac{100-98,8}{100} \times 100\% = 0,012\%$
0,7	100	99,6	$\frac{100-99,6}{100} \times 100\% = 0,004\%$

4. Gerak Kiri – Kiri

Dan yang terakhir, pengujian gerak kesamping kiri secara *angular* dengan arah lintasan kekiri didapatkan hasil persentase eror ketepatan gerakan *quadcopter* terendah adalah 0,002% dengan kecepatan v_{xy} sebesar 1.

Tabel 6. 17 Besar Eror Gerak Kiri - Kiri

Nilai v_{xy}	r acuan (cm)	rata - rata r lintasan (cm)	Persentase Eror
1	100	100,2	$\frac{100-100,2}{100} \times 100 = 0,002\%$
0,9	100	98,2	$\frac{100-98,2}{100} \times 100\% = 0,018\%$
0,8	100	101,2	$\frac{100-101,2}{100} \times 100\% = 0,012\%$
0,7	100	111,4	$\frac{100-111,4}{100} \times 100\% = 0,114\%$

Dari hasil perhitungan persentase eror yang telah dilakukan, dan dengan menggunakan rumus 6.2 didapatkan tingkat akurasi sistem berdasarkan jarak yang di tempuh sebesar 99,99%

6.3 Pengujian Estimasi Waktu Gerak sistem

6.3.1 Tujuan Pengujian

Pengujian ini bertujuan untuk mengetahui seberapa tingkat estimasi waktu yang ditempuh *quadcopter*.

6.3.2 Pelaksanaan Pengujian

Pengujian ini dilakuakan dengan memberi nilai pada pada parameter $angular_z$ dan salah satu parameter $linear_x$ atau $linear_y$ untuk menentukan gerak dari *quadcopter*. Pengujian ini dilakukan untuk mencari berapa lama waktu yang dibutuhkan oleh *quadcopter* untuk melakukan gerak *angular*. Untuk mengakses waktu pada python dapat digunakan fungsi import time. Untuk menghitung waktu proses dapat dilakuan dengan cara mengurangi waktu saat program telah di eksekusi dengan waktu awal saat program berjalan.

6.3.3 Prosedur Pengujian

Pengujian ini dilakukan sesuai dengan prosedur pengujian berikut:

1. Pasang baterai *quadcopter* yang sudah terisi penuh. *quadcopter* dapat digunakan setelah *quadcopter* terkalibrasi atau drone mengeluarkan bunyi beep sebanyak 4 kali.
2. Menghubungkan komputer dengan Wi-Fi yang terdapat pada *quadcopter*
3. Buka terminal pada *Ubuntu* ketikkan *command "$ cd tum_simulator_ws"* kemudian ketikkan *"$ source devel/setup.bash"*, dan yang terakhir ketikkan *command "$ roslaunch ardrone_autonomy ardrone.launch*" agar *quadcopter* terhubung dengan ROS.
4. Buka terminal baru *command "$ cd tum_simulator_ws"* kemudian ketikkan *"$ source devel/setup.bash"*, dan untuk menjalankan kode program ketikan *"$ rosrun sekrip baru.py"*.
5. Amati pergerakan yang dilakukan dari *quadcopter* apakah sesuai dengan yang diinginkan atau tidak.
6. Data *navigasi* yang ditampilkan pada terminal akan disimpan di *libre office* dalam bentuk xls.
7. Ulang langkah ke-5 sebanyak lima kali dengan nilai v_{xy} yang digunakan sebagai pengujian adalah nilai yang memiliki persentase eror terendah.

6.3.4 Hasil Pengujian

6.3.4.1 Gerakan Pada Sumbu x

1. Gerak Maju - Kiri

   Tabel 6.18 adalah estimasi waktu yang ditempuh oleh *quadcopter* dalam lima kali percobaan saat melakukan gerak maju secara *angular* dengan arah lintasan kekiri dengan nilai kecepatan v_{xy} optimal yang didapatkan pada pengujian sebelumnya yaitu 1.

Tabel 6. 18 Estimasi Waktu Gerak Maju - Kiri

Nilai v_{xy}	Pengujian ke-	Waktu yang didapatkan (s)
1	1	0.018526

	2	0.018282
	3	0.017373
	4	0.013669
	5	0.019195
rata-rata waktu tempuh (s)		0.017409

2. Gerak Maju – Kanan

Estimasi waktu yang ditempuh oleh *quadcopter* saat melakukan lima kali percobaan pada gerak maju secara *angular* dengan arah lintasan kekanan ditunjukkan pada Tabel 6.19. Dengan kecepatan v_{xy} optimal yang didapatkan dari pengujian sebelumnya yaitu 1.

Tabel 6. 19 Estimasi Waktu Gerak Maju - Kanan

Nilai v_{xy}	Pengujian ke-	Waktu yang didapatkan (s)
1	1	0.017325
	2	0.019792
	3	0.018371
	4	0.018793
	5	0.02337
rata-rata waktu tempuh (s)		0.01953

3. Gerak Mundur – Kanan

Estimasi waktu yang ditempuh oleh *quadcopter* saat melakukan gerak mundur secara *angular* dengan arah lintasan kekanan dengan lima kali percobaan ditunjukkan pada Tabel 6.20. Dengan kecepatan v_{xy} yang optimal dalam melakukan gerak ini adalah 0.9.

Tabel 6. 20 Estimasi Waktu Gerak Mundur - Kanan

Nilai v_{xy}	Pengujian ke-	Waktu yang didapatkan (s)
0,9	1	0.019418
	2	0.022043
	3	0.019933
	4	0.01788
	5	0.020311
rata-rata waktu tempuh (s)		0.019917

4. Gerak Mundur – Kiri

Tabel 6.21 menunjukkan estimasi waktu yang ditempuh oleh *quadcopter* saat melakukan gerak mundur secara *angular* dengan arah lintasan kekiri. Dengan kecepatan v_{xy} yang didapatkan dari pengujian sebelumnya untuk gerak mundur-kiri adalah 1.

Tabel 6. 21 Estimasi Waktu Gerak Mundur - Kiri

Nilai v_{xy}	Pengujian ke-	Waktu yang didapatkan (s)
1	1	0.016957
	2	0.017842
	3	0.016014
	4	0.0146
	5	0.01762
rata-rata waktu tempuh (s)		0.016607

6.3.4.2 Gerakan Pada Sumbu y

1. Gerak Kanan – Kanan

Estimasi waktu dalam lima kali percobaan yang ditempuh *quadcopter* dalam melakukan gerak kesamping kanan secara *angular* dengan arah lintasan kekanan dengan kecepatan v_{xy} optimal adalah 1 ditunjukkan oleh Tabel 6.22 .

Tabel 6. 22 Estimasi Waktu Gerak Kanan - Kanan

Nilai v_{xy}	Pengujian ke-	Waktu yang didapatkan (s)
1	1	0.018526
	2	0.011091
	3	0.014852
	4	0.010509
	5	0.013795
rata-rata waktu tempuh (s)		0.013755

2. Gerak Kanan – Kiri

Tabel 6.23 menunjukkan estimasi waktu yang ditempuh dalam lima kali percobaan melakukan gerak kesamping kanan secara *angular* dengan arah lintasan kekiri dengan besar nilai kecepatan v_{xy} optimal untuk pergerakan ini adalah 0.9.

Tabel 6. 23 Estimasi Waktu Gerak Kanan - Kiri

Nilai v_{xy}	Pengujian ke-	Waktu yang didapatkan (s)
0,9	1	0.016997
	2	0.020657
	3	0.017871
	4	0.018343
	5	0.020293
rata-rata waktu tempuh (s)		0.018832

3. Gerak Kiri– Kanan

Estimasi waktu yang ditempuh *quadcopter* dalam melakukan gerak kesamping kiri secara *angular* dengan arah lintasan kekanan ditunjukkan oleh Tabel 6.24 dedngan kecepatan v_{xy} optimal untuk gerak ini sebesar 0.7.

Tabel 6. 24 Estimasi Waktu Gerak Kiri - Kanan

Nilai v_{xy}	Pengujian ke-	Waktu yang didapatkan (s)
0,7	1	0.02073
	2	0.022043
	3	0.02018
	4	0.01788
	5	0.020311
rata-rata waktu tempuh (s)		0.020229

4. Gerak Kiri – Kiri

Dan yang terakhir adalah estimasi waktu yang ditempuh *quadcopter* dalam melakukan gerak kesamping kanan secara *angular* dengan arah lintasan kekanan mengunakan kecepatan v_{xy} optimalnya yang didapatkan dari pengujian sebelumnya yaitu 1 ditunjukkan oleh Tabel 6.25.

Tabel 6. 25 Estimasi Waktu Gerak Kiri - Kiri

Nilai v_{xy}	Pengujian ke-	Waktu yang didapatkan (s)
1	1	0.021348
	2	0.022179
	3	0.018635
	4	0.019529
	5	0.016942

rata-rata waktu tempuh (s)	0.019727

6.3.5 Analisis Pengujian

Dari pengujian yang telah dilakukan, dengan menggunakan perhitungan

$$rata - rata = \frac{\sum rata{-}rata\ waktu}{total\ data} \quad (6.3)$$

menunjukkan bahwa *quadcopter* dapat melakukan gerak *angular* ke berbagai arah dengan rata-rata waktu yang ditempuh sebesar 0,018225 detik dengan kecepatan laju dari *quadcopter* v_{xy} bernilai 0,7 hingga 1. Dapat dilihat pada hasil pengujian Tabel 6.18 hingga Tabel 6.25 bahwa semakin kecil nilai v_{xy} yang diberikan maka estimasi waktu yang dibutuhan untuk melakukan gerak *angular* akan lebih banyak.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan analisis hasil dari pengujian yang telah dilakukan, maka dapat ditarik beberapa kesimpulan sesuai dengan rumusan masalah yang telah ditentukan sebelumnya yaitu.

1. Sistem navigasi untuk gerak *angular* pada *quadcopter* dapat dibuat dengan memanfaatkan parameter $angular_z$ pada *quadcopter*. Dimana untuk menghasilkan gerak *angular* dengan lintasan satu lingkaran penuh nilai pada parameter $angular_z$ dan v_{xy} bernilai maksimum yaitu 1 dengan nilai iterasi (i) yaitu 22.
2. Dalam pengujian ketepatan gerak *angular* dengan lintasan ¼ lingkaran pada berbagai arah dengan kecepatan yang berbeda - beda didapatkan nilai kecepatan v_{xy} yaitu 1 untuk gerak agular maju – kiri, maju – kanan, mundur – kiri, kanan – kanan dan kiri – kiri. Kecepatan v_{xy} sebesar 0,9 untuk gerak *angular* mundur – kanan dan kanan - kiri. Dan kecepatan v_{xy} sebesar 0,7 untuk gerak *angular* kiri – kanan.
3. Estimasi waktu pada saat *quadcopter* melakukan gerak *angular* dengan berbagai arah pada lintasan ¼ lingkaran memiliki rata – rata 0,018225 detik.
4. Tingkat akurasi jarak (r) yang ditempuh oleh *quadcopter* sebesar 99,99%.

7.2 Saran

Terdapat beberapa saran untuk pengembangan sistem agar sistem mendapatkan hasil yang lebih baik lag diantaranya adalah.

1. Sistem dapat bergerak secara *angular* tidak hanya pada koordinat sumbu x dan y, melainkan pada sumbu z.
2. Sistem dapat menambahkan sebuah sensor atau memanfaatkan kamera yang terpasang pada *quadcopter* agar dapat bergerak secara *angular* saat mendeteksi adanya suatu objek yang berada di depannya.

DAFTAR PUSTAKA

Andika, I. G., Yanti, C. P. & Cardewa, M., 2018. *QUADCOPTER* OBSTACLE AVOIDANCE DENGAN SENSOR INFRAMERAH UNTUK PEMANTAUAN BENCANA ALAM MELALUI UDARA. *Jurnal Pendidikan Teknologi dan Kejuruan,* Volume Vol. 15, No. 1, p. Hal :71.

Anggarjito, M. J., 2013. Perancangan dan Implementasi Kontroler PID dengan Nonlinear Decoupling pada Sistem Kendali Way-to-Way Point UAV *Quadcopter*. *JURNAL TEKNIK POMITS,* Volume Vol. 2, No. 2,, pp. 2337-3539.

Catanzariti, R., 2012. *PC World From IDG.* [Online] Available at: www.pcworld.idg.com.au [Diakses 27 April 2018].

CLC, C. L. C., 2015. *Gerak Melingkar Suatu benda.* [Online] Available at: http://fisikazone.com/ [Diakses 13 June 2018].

Dharmawan, A. & Rahmawati, N., 2014. Sistem Penghindar Halangan Otomatis dan Penahan Ketinggian Penerbangan pada *Quadcopter*. *IJETS,* Volume Vol.4, No.1, p. pp. 1~12.

Hisham.id, 2015. *Pengertian dan Rumus Kecepatan Linier dan Anguler.* [Online] Available at: https://hisham.id/2015/12/pengertian-dan-rumus-kecepatan-linier-dan-anguler.html [Diakses 4 June 2018].

Monajjemi, M., 2015. *ardrone_autonomy Documentation Release indigo-devel.* s.l.:s.n.

SA, P. D., 2016. *PARROT.* [Online] Available at: https://www.*Parrot*.com/us/drones/*Parrot*-ardrone-20-elite-edition#technicals [Diakses 27 April 2018].

Setyawan, G. E., Setiawan, E. & Kurniawan, W., 2015. SISTEM KENDALI KETINGGIAN *QUADCOPTER* MENGGUNAKAN PID. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* Volume Vol. 2, No. 2,, pp. 125-131.

Sphinx, 2014. *ardrone_autonomy.* [Online] Available at: http://ardrone-autonomy.readthedocs.io [Diakses 28 June 2018].