SISTEM PENGENDALIAN KECEPATAN PADA *QUADCOPTER* DENGAN MENGGUNAKAN METODE *LINEAR QUADRATIC REGULATOR* (*LQR*)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Mesra Diana Tamsar
NIM: 145150301111070

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

PENGESAHAN

SISTEM PENGENDALIAN KECEPATAN PADA *QUADCOPTER* DENGAN MENGGUNAKAN METODE *LINEAR QUADRATIC REGULATOR (LQR)*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Mesra Diana Tamsar
NIM: 145150301111070

Skripsi ini telah diuji dan dinyatakan lulus pada
02 Agustus 2018

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Gembong Edhi Setyawan, S.T., M.T
NIK: 201208 761201 1 001

Dosen Pembimbing II

Wijaya Kurniawan, S.T, M.T
NIP: 19820125 201504 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Ir. Agoeng 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 di dalam 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, 08 Juli 2018

Mesra Diana Tamsar

NIM: 145150301111070

KATA PENGANTAR

Assalamualaikum Wr. Wb

Alhamdulillah, puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik serta hidayah-Nya, sehingga penulis dapat menyelesaikan laporan skripsi dengan judul "*Sistem Pengendali Kecepatan Pada Quadcopter Menggunakan Metode Linear Quadratic Regulator (LQR)*" dengan baik.

Dalam penyusunan dan penelitian skripsi ini tidak lepas dari bantuan moral dan materil yang diberikan dari berbagai pihak, maka peneliti mengucapkan banyak terima kasih kepada:

1. Allah SWT atas karunia dan kesehatan yang diberikan selama ini sehingga skripsi ini dapat terselesaikan dengan baik.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Heru Nurwarsito, Ir., M.Kom. selaku Wakil Dekan I Bidang Akademik Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
5. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya Malang.
6. Bapak Gembong Edhi Setyawan, S.T., M.T., selaku dosen pembimbing satu yang telah memberikan ilmu, serta saran dalam membimbing penulis untuk menyelesaikan skripsi dari awal sampai akhir. Terima kasih juga telah memberikan banyak pengalaman dan pelajaran yang sebelumnya belum pernah diterima oleh penulis.
7. Wijaya Kurniawan, S.T, M.T selaku dosen pembimbing dua yang telah memberikan ilmu, saran, penjelasan motivasi, serta membantu dalam penyusunan laporan penulis.
8. Bapak J.Tamsar dan Ibu R.Lingga selaku orang tua penulis, Joy Tamsar, Jhon Tamsar sebagai abang penulis dan Maijon Tamsar sebagai adik penulis, serta seluruh keluarga besar yang selalu memberi dukungan dan do'a agar penulis dapat menyelesaikan skripsi ini dengan lancar.
9. Seluruh civitas akademika Informatika Universitas Brawijaya dan teman-teman Teknik Komputer Angkatan 2014 yang telah banyak memberi bantuan dan dukungan selama peneliti menempuh studi di Teknik Komputer Universitas Brawijaya dan selama penyelesaian skripsi ini.
10. Keluarga permasesa malang yang telah memberikan dukungan, hiburan, dan do'a selama masa pengerjaan skripsi ini.

11. Kak susi, kak lenny, kak eva, kak nidar, kak tiara, kaki ita, kak tari, devi yang telah memberikan dukungan, hiburan, dan do'a selama masa pengerjaan skripsi ini.
12. Ayang Setiyo Putri, Achmad Baecuni, Dimas Angger, Sabita Wildani, Yusril Dewantara, Andyan Bina, Sabitha Wildani, Amroy Casro, Fajar Miftakhul selaku senior *quadcopter research* dan teman seperjuangan *quadcopter* Cindy, Enno, Muliya, Haqqi, Satria serta rekan-rekan lainnya yang turut memberikan motivasi dan hingga pengerjaan skripsi ini selesai.
13. Seluruh pihak yang tidak dapat diucapkan satu persatu, peneliti mengucapkan banyak terima kasih atas segala bentuk dukungan dan doa sehingga laporan skripsi ini dapat terselesaikan.

Penulis menyadari bahwa tulisan ini masih jauh dari kata sempurna dan masih memiliki berbagai macam kekurangan. Oleh karena itu penulis mengharapkan kritik dan saran yang membangun, agar ke depannya penulis dapat menjadi lebih baik lagi. Semoga laporan skripsi ini dapat memberi manfaat bagi perkembangan ilmu pengetahuan di kemudian hari.

Malang, 08 Juli 2018

Penulis

mesratamsar@gmail.com

ABSTRAK

Unmanned Aerial Vehicle (UAV) merupakan pesawat tanpa awak yang banyak mendapat perhatian dari berbagai kalangan pada saat ini, karena dapat menggantikan peran manusia sebagai sistem kendalinya di berbagai bidang. *Quadcopter* memiliki masalah yang sangat kompleks salah satunya adalah untuk mengendalikan kecepatan, pada hal ini *quadcopter* memiliki kecepatan yang tidak sama ketika melakukan pergerakan dengan kecepatan yang ditetapkan oleh pengguna yang dapat menyebabkan terjadinya insiden yang dapat merusak *quadcopter* dan adanya keterlambatan estimasi waktu ditentukan oleh pengguna. Pertama yang dilakukan adalah melakukan perhitungan *Linear Quadratic Regulator* yang akan digunakan untuk menstabilkan kecepatan yang ada pada *quadcopter*. Kemudian *user* akan melakukan *input* jarak untuk ditempuh oleh *quadcopter.* Pengujian dilakukan dengan menguji tingkat kestabilan kecepatan dan tingkat ketepatan jarak yang telah ditentukan. Hasil menunjukan tingkat akurasi ketepatan jarak adalah 100% dan untuk tingkat kestabilan kecepatan juga 100%. Untuk waktu tempuh setiap jarak 1m, 1.5m, 2m, 2.5m, 3m membutuhkan waktu 1.2 detik ,2.43 detik, 2.07 detik, 3.35 detik dan 4.07 detik.

Kata Kunci: *Quadcopter*, *Linear Quadratic Regulator*, Kecepatan, Jarak

ABSTRACT

Unmanned Aerial Vehicle (UAV) is an unmanned aircraft that received much attention from various circles at this time, because it can replace the human role as a control system in various fields. Quadcopter has a very complex problem one of which is to control the speed, in this case the quadcopter has a speed that is not the same when performing the movement with the speed set by the user that can cause an incident that can damage the quadcopter and the existence of the user time estimation is determined by the user. The first thing to do is to calculate the Linear Quadratic Regulator that will be used to stabilize the speed of the quadcopter. Then the user will do input distance to be taken by quadcopter. Testing is done by testing the stability rate of speed and the level of accuracy of a predetermined distance. The result shows the accuracy of the distance accuracy is 100% and for the stability rate the speed is also 100%. For travel time every distance of 1m, 1.5m, 2m, 2.5m, 3m takes 1.2 seconds, 2.43 seconds, 2.07 seconds, 3.35 seconds and 4.07 seconds.

Keywords: Quadcopter, Linear Quadratic Regulator, Speed, Distance

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I PENDAHULUAN

1.1 Latar belakang

Unmanned Aerial Vehicle (UAV) didefinisikan sebagai pesawat tanpa awak. *UAV* banyak mendapat perhatian dari berbagai kalangan pada saat ini, karena dapat menggantikan peran manusia sebagai sistem kendalinya. *UAV* dapat dioperasikan jarak jauh dengan menggunakan sitem *remote control* oleh pilot. Salah satu jenis *UAV* adalah jenis *quadcopter* dengan 4 buah motor yang dipasang simetris pada ujung-ujung kerangka utama. Empat motor pada *quadcopter* memiliki kecepatan yang berbeda-beda (Hamdani *et, al,* 2013).

Quadcopter sering digunakan dalam berbagai bidang, seperti bidang industri, militer, *entertainmen*t dan lain-lain (M, Latif, 2014). Terlepas dari kelebihan *quadcopter*, seperti pesawat terbang lainnya, *quadcopter* memiliki masalah yang sangat kompleks salah satunya adalah untuk mengendalikan kecepatan. Dalam pengendalian kecepatan *quadcopter* ini, terdapat masalah yaitu kecepatan pada saat *quadcopter* melakukan pergerakan tidak sesuai dengan kecepatan yang ditetapkan oleh pengguna. Masalah itu dapat menyebabkan beberapa hal buruk yang tidak diinginkan oleh pengguna. Beberapa dampak dari masalah itu adalah terjadinya insiden yang dapat merusak *quadcopter* dan adanya keterlamabatan estimasi waktu ditentukan oleh pengguna.

Salah satu penelitian yang berhubungan dengan sistem kendali kecepatan adalah penelitian yang dilakukan oleh Ahmadi (2015) dan Parlina (2016), yaitu mengoptimalkan kecepatan pada robot *inverted pendulum* dengan menggunakan metode *Linear Quadratic Regulator (LQR)* dan kontrol kecepatan pada robot *inverted pendulum* menggunakan metode *fuzzy.* Masalah yang ada pada kecepatan robot *inverted pendulum* adalah ketidak stabilan pada kecepatan sudut dan kemiringan yang ada pada robot serta tidak dapat mempertahankan kecepatan gerak robot secara konstan. Penggunaan metode *LQR* untuk mengoptimalkan kecepatan pada robot *inverted pendulum* dapat mengatasi permasalahan yang ada yaitu dapat mengoptimalkan kecepatan pada robot dari ketidak pastian sistem yang ada. Sedangkan penggunaan metode *fuzzy* juga mampu mempertahankan keadaan seimbang pada robot, tetapi respon kecepatan gerak konstan pada robot yang dihasilkan masih mengalami *overshoot*. *Overshoot* merupakan proses robot dalam menyeimbangkan batang *pendulum.*

Penelitian ini akan menggunakan *Parrot Ar Drone 2.0* sebagai media penelitian yang akan di teliti. *Ar Drone* adalah salah satu jenis *quadcopter* yang dapat di kembangkan secara bebas karena *quadcopter* ini sudah dilengkapi dengan beberapa sensor yang akan digunakan dalam penelitian ini yaitu seperti sensor *Inertial Measurement Unit (IMU).* Sensor *IMU* pada *Ar Drone* ini memiliki fungsi untuk menghitung percepatan dan arah pergerakan yang ada pada *quadcopter* berjenis *Ar Drone*.

Sedangkan untuk metode yang akan digunakan untuk melakukan kendali atau kontrol terhadap *quadcopter* tersebut ialah algoritma *LQR*. Metode optimasi

dengan *LQR* adalah dengan menentukan sinyal masukan yang akan memindahkan suatu state sistem linear dari kondisi awal menuju ke suatu kondisi akhir yang akan meminimumkan suatu indeks untuk kerja performa *quadratik* (Sumanti, 2014). Metode ini diharapkan akan menjadi solusi untuk mengoptimalkan kecepatan pada *quadcopter*.

1.2 Rumusan masalah

Berdasarkan permasalahaan yang ada pada uraian diatas, maka rumusan masalah yang akan dibahas adalah sebagai berikut ini:

1. Bagaimana kinerja sistem pengendalian kecepatan yang dibangun menggunakan metode *LQR*?
2. Bagaimana tingkat ketepatan jarak yang ditempuh *quadcopter* dengan kecepatan yang tetap?

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui kinerja sistem kontrol *LQR* pada perubahan kecepatan pada Ar Drone.
2. Mengetahui tingkat ketepatan jarak yang ditempuh dengan kecepatan yang tetap.

1.4 Manfaat

Manfaat dari dilakukanya penelitian ini adalah untuk dapat mempermudah pengguna *quadcopter* dalam mengatasi kecepatan yang tidak stabil pada *quadcopter* untuk mengurangi adanya terjadinya kecelakan atau insiden.

1.5 Batasan masalah

Batasan masalah pada penelitian ini agar lebih fokus pada perumusan masalah dan tidak menyimpang dari rumusan masalah adalah sebagai berikut:

1. *Quadcopter* yang digunakan adalah *Parrot AR Drone 2.0*
2. Sensor yang digunakan adalah sensor *IMU* yang ada pada *Parrot AR Drone 2.0*
3. *Quadcopter* dilakukan pengujian di dalam ruangan
4. Metode yang digunakan adalah algorithma *Linear Quadratic Regulator* (*LQR*)
5. Jarak yang digunakan adalah 1m, 1.5m, 2m,2.5m dan 3 m.

1.6 Sistematika Pembahasan

Sistematika penulisan bertujuan sebagai penjelasan umum dari bagian-bagian bab yang ada pada penelitian ini agar memudahkan pembaca dalam mengikuti alur penelitian. Adapun sistematika penulisan adalah sebagai berikut:

BAB I Pendahuluan

Bab ini berisikan latar belakang masalah, identifikasi masalah, rumusan masalah, batasan masalah, manfaat penelitian, dan sistematika penulisan.

BAB II Landasan Kepustakaan

Bab ini menguraikan sekilas teori ,materi kajian pustaka, penjelasan mengenai pergerkan *quadcopter*, dan dasar metode algorithma *Linear Quadratic Regulator*.

BAB III Metodologi

Bab ini menjelaskan mengenai metode penelitian yang digunakan untuk sistem kontrol kecepatan pada *quadcopter* menggunakan metode *Linear Quadratic Regulator*.

BAB IV Analisis Kebutuhan

Bab ini terdiri dari metode yang digunakan dan perancangan penelitian tetang sistem kontrol optimal pada kecepatan *quadcopter* menggunakan *Linear Quadratic Regulator*.

BAB V Perancangan dan Implementasi

Bab ini berisi implementasi sistem kontrol kecepatan pada *quadcopter* menggunakan metode *Linear Quadratic Regulator.*

BAB VI Pengujian dan Analisa Hasil

Bab ini terdiri dari pembahassan dan pengujian sistem kontrol kecepatan pada *quadcopter* dengan menggunakan metode *Linear Quadratic Regulator.*

BAB VII Penutup

Memuat kesimpulan dari hasil penelitian yang telah dilakukan dan saran terhadap penelitian agar lebih baik kedepanya.

BAB II LANDASAN KEPUSTAKAAN

Untuk memahami sistem secara keseluruhan dengan baik, maka perlu adanya tinjauan pustaka tentang dasar teori maupun perancangan sistem, serta penunjang yang akan dibahas dalam penelitian ini. Adapun teori yang akan dibahas adalah sebagai berikut:

- UAV *Quadcopter*
- Konsep sistem kontrol optimal dengan penggunaan metode *Linear Quadratic Regulator (LQR)*

1.1. Tinjauan Pustaka

Penelitian yang menjadi acuan peneliti dalam melakukan pengujian adalah penelitian yang dilakukan oleh Soraya Parlina (2016) yaitu Kontrol kecepatan pada robot *inverted pendulum* dengan kontroler *fuzzy*. Masalah pada penelitian ini adalah robot yang tidak dapat mempertahankan kecepatan gerak robot secara konstan. Pada penelitian ini dilakukan proses perancangan dan pengujian yang menghasilkan kontroller hasil yang mampu membuat robot bergerak dengan konstanta dan dapat mempertahankan keadaan yang seimbang. Pengujian ini dilakukan secara *real time* dan juga secara simulasi pada matlab. Hasil yang diperoleh dari penelitian ini adalah adanya noise pada sensor gyroscope yang menyebabkan robot mengalami overshoot ketika meyeimbangkan sudut pendulum.

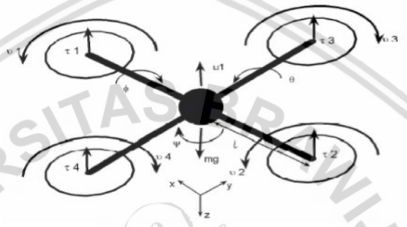
Penelitian yang dilakukan oleh Ahmadi (2015) yaitu penelitian yang melakukan kestabilan robot *Inverted Pendulum* dengan menggunakan metode Linear Quadratic Regulator. Pengujian dilakukan dengan 2 cara yaitu secara real time dan secara simulasi. Pengujian sistem dilakukan secara real time dan dapat melihat ketika sistem mencapai *steady state* dengan waktu yang tidak jauh berbeda pada pengujian secara simulasi. Pada penelitian ini terdapat dua masukan yang ada yaitu kecepatan sudut dan sudut kemiringan. Pada penelitian ini telah dilakukan pengujian sebanyak 7 kali terhadap robot *inverted pendulum* secara *real time* dan juga dilakukan simulasi pada matlab. Hasil yang diperoh adalah metode *LQR* yang digunakan mampu menstabilkan dari gangguan dan ketidak pastian dari sistem.

Penelitian selanjutnya yang menjadi acuan dalam penelitian ini adalah pengaturan optimal untuk menjaga kestabilan *hover* pada *quadcopter*, yang diteliti oleh kardono (2012). Pada penelitian ini kontrol *LQR* yang dilakukan dalam simulasi dapat mengatasi gangguan yang diberikan pada ketinggian, sudur *roll,* maupun sudut *picth*. Tidak jauh berbeda dengan simulasi, ketika dilakukan implementasi juga metode *LQR* dapat menjaga kestabilan *hover* dengan baik, namum respon yang dihasikan tidak semaksimal ketika dilakukan di simulasi. Hal ini dikarenakan kemampuan mikrokontroler tidak dapat mengolah algorithma aritmatika dengan cepat seperti pengolahan pada matlab.

1.2. Dasar Teori

2.2.1. UAV *Quadcopter*

Quadcopter adalah salah satu jenis pesawat tanpa awak dengan 4 buah motor berfungsi sebagai baling-baling yang digunakan untuk melakukan pergerakan dan dapat bergerak kesegala arah serta dapat melayang di udara (*hovering*). Dengan adanya kemampuan tersebut maka *quadcopter* sering digunakan dalam melakukan *tracking* terhadap objek yang ditangkap oleh kamera yang ada pada *quadcopter*. *Quadcopter* memiliki pengaturan motor dalam melakukan pergerakan yaitu pengaturan motor depan dan motor belakang yang berputar searah dengan jarum jam, dan untuk pengaturan motor kiri dan kanan berputar berlawanan dengan arah jarum jam. Gambar 2.1 menunjukkan pergerakan *quadcopter* secara keseluruhan.



Gambar 2. 1.Pergerakan *Quadcopter*

Pada gambar 2.1. keempat motor diatas bersifat bebas karena kecepatan setiap motor berbeda. Ketika mengubah kecepatan setiap motor maka pergerakan *quadcopter* juga akan berubah. *Quadcopter* memiliki kelebihan yang dapat bergerak ke segala arah dan dapat menjangkau tempat yang tidak dapat di jangkau oleh manusia. Untuk dapat terbang, ada syarat yang harus dipenuhi oleh *quadcopter* yaitu ketika gaya angkat *quadcopter* lebih besar dibandingkan dengan gaya grafitasi. Kondisi *hover* akan tercapai ketika kondisi dari titik berat setiap motor dan kecepatan yang ada pada motor sama. Ada beberapa istilah yang ada pada pergerakan *quadcopter* yaitu, *picth* θ merupakan pergerakan pada rotasi sumbu y (depan dan belakang), roll ϕ merupakan pergerakan pada rotasi sumbu x (kiri dan kanan), yaw ψ merupakan pergerakan pada rotasi sumbu z atau sering disebut rotasi ke kiri atau rotasi ke kanan, Throttle (z) merupakan pergerakan untuk menaikkan dan menurunkan *quadcopter*.

Desain kontrol yang akan dibangun pada *quadcopter* ini adalah kontrol *attitude (roll, pitch, yaw)* dan kontrol ketinggian sebagai *pendaratan.* Sistem kendali pendaratan pada *quadcopter* ini membutuhkan persamaan pergerakan *quadcopter* agar dapat dikendalikan. Persamaan pergerakan model matematika *quadcopter* yang digunakan adalah berdasarkan kardono (2012) dan tomasso (2008), yaitu sebagai berikut :

$$\ddot{\phi} = \dot{\theta}\,\dot{\psi}\frac{I_y - I_z}{I_x} - \dot{\theta}\Omega\frac{J_r}{I_x} + \frac{l}{I_x}U2 \qquad (2.1)$$

$$\ddot{\theta} = \dot{\phi}\,\dot{\psi}\frac{I_z - I_x}{I_y} - \dot{\theta}\Omega\frac{J_r}{I_x} + \frac{l}{I_y}U3$$

$$\ddot{\psi} = \dot{\phi}\,\dot{\theta}\frac{I_x - I_y}{I_z} + \frac{l}{I_z}U4$$

$$\ddot{Z} = -g + (\cos\phi\cos\theta)\frac{1}{m_r}U1$$

Dengan $\ddot{\phi}$ = persamaan pergerakan rotasi pada sumbu x

$\ddot{\theta}$ = persamaan pergerakan rotasi pada sumbu y

$\ddot{\psi}$ = persamaan pergerakan rotasi pada sumbu z

$\ddot{Z}$ = persamaan pergerakan translasi pada ketinggian z

Berdasarkan dari model matematis yang sudah dijabarkan di atas selanjutnya persamaan tersebut diubah menjadi sebuah fungsi yang selanjutnya akan berubah menjadi :

$$\dot{x} = f(X,U) = \begin{pmatrix} \dot{\phi} \\ \dot{\theta}\,\dot{\psi}a_1 - \dot{\theta}a_2\Omega_r + b_1U2 \\ \dot{\theta} \\ \dot{\theta}\,\dot{\psi}a_3 - \dot{\theta}a_4\Omega_r + b_2U3 \\ \dot{\psi} \\ \dot{\theta}\dot{\phi}\,a_5 + b_3U4 \\ \dot{x} \\ -g + (\cos\phi\cos\theta)\frac{1}{m_r}U1 \end{pmatrix} \quad (2.2)$$

Di mana :

$$\begin{Bmatrix} a_1 = \frac{I_y - I_z}{I_x} & a_5 = \frac{I_x - I_y}{I_z} \\ a_2 = -\frac{J_r}{I_x} & b_1 = \frac{l}{I_x} \\ a_3 = \frac{\dot{\psi}(I_z - I_x)}{I_y} & b_2 = \frac{l}{I_y} \\ a_4 = \frac{J_r}{I_x} & b_3 = \frac{l}{I_z} \end{Bmatrix} \quad (2.3)$$

Dan *input* adalah :

$$\begin{Bmatrix} U1 = b\left({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2\right) \\ U2 = b\left(-{\Omega_2}^2 + {\Omega_4}^2\right) \\ U3 = b\left({\Omega_1}^2 - {\Omega_3}^2\right) \\ U4 = d\left(-{\Omega_1}^2 + {\Omega_2}^2 - {\Omega_3}^2 + {\Omega_4}^2\right) \end{Bmatrix} \quad (2.4)$$

Dari persamaan *quadcopter* (2) di atas kemudian dituliskan menjadi *state space quadcopter* di mana $\dot{x} = f(x,u)$ dimana x adalah *state vector* dan u adalah *input vector*.

$$x = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} & x & \dot{x} \end{bmatrix}^T$$
$$u = \begin{bmatrix} U_1 & U_2 & U_3 & U_4 \end{bmatrix}^T \quad (2.5)$$

Persamaan di atas akan dimasukkan pada *state space* dengan :

$$\dot{X} = A_x + B_u$$
$$y = C_x + D_u \quad (2.6)$$

Selanjutnya berdasarkan persamaan (2.1) dan persamaan (2.5) akan menghasilkan matriks sebagai berikut :

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dot{\theta}\,\dot{\psi} a_1 - \dot{\theta} a_2 \Omega_r & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \dot{\psi} a_3 - \dot{\theta} a_4 \Omega_r & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \dot{\phi}\, a_5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.7)$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & b1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & b2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & b3 \\ 0 & 0 & 0 & 0 \\ (\cos\phi \cos\theta)\dfrac{1}{m_r} & 0 & 0 & 0 \end{bmatrix} \quad (2.8)$$

$$C = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix} \text{ dan } D = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \quad (2.9)$$

2.2.2. *Linear Quadratic Regulator*

Linear Quadratic Regulation, disebut *Linear* karena model dan bentuk kontrolernya berupa linear, sedangkan disebut *quadratik* karena *cost function* nya adalah quadratik dan disebut regulator karena referensinya bukanlah berupa fungsi waktu. Sistem kontrol optimal adalah sistem yang mempunyai unjuk kerja yang terbaik (*best performance*) terhadap suatu acuan tertentu.

Pengukuran dilakukan dengan menentukan indeks performansi atau disebut juga *cost function*, yang merupakan suatu fungsi dari nilai untuk membuktikan apakah kinerja sistem sudah sesuai dengan yang di inginkan. Pada sistem ini digunakan teori modern dengan *multi input* dan *multi output* yang linear. Hasil dari teori ini adalah solusi untuk mengoptimalkan antara sistem masukan dan keluaran yang saling berhubungan. Berbeda dengan sistem kontrol yang dirancang dengan teori kontrol konvensional atau yang sering disebut *classic control* yang hanya memiliki satu masukan dan satu keluaran dan metode yang prosedurnya didasarkan pada coba-coba (*trial and error*) yang tidak menghasilkan sistem kontrol yang optimum, berdasarkan hal itu maka pendekatan yang paling sesuai untuk analisis sistem adalah pendekatan ruang keadaan (*state space*).

Untuk menganalisa suatu sistem, terlebih dahulu harus didapatkan persamaan (model matematika) dari sistem yang akan mewakili unjuk kerja dari sistem tersebut. Dari persamaan tersebut kemudain akan direpresentasikan ke dalam persamaan ruang keadaan (*state space*) (Novi, 2005).

Persamaan model matematis dibutuhkan untuk menganalisis sistem pengendali optimal, karena persamaan inilah nantinya yang mewakili unjuk kerja dari sistem. Persamaan ini di representasikan ke dalam persamaan ruang keadaan (*state space*), dan untuk mendapatkan sistem *quadcopter* dalam model matematis sistem ini dinyatakan dalam bentuk persamaan *differensial* orde pertama (Ogata, 1997)

Ruang keadaan(*state space*) dari sistem orde ke-*n* yang dinyatakan oleh persamaan *differensial* linier dengan fungsi penggerak tidak melibatkan bentuk turunan, misalkan suatu persamaan orde pertama. Jika dimisalkan berikut :

$$a_0 y + a_1 y^{(n)} + \cdots + a_{n+1} y^{(n-1)} + a_n y = u \qquad (2.10)$$

Di mana y merupakan keluaran sistem dan u adalah masukan sistem. Untuk membuat model matematis dari persamaan tersebut terlebih dahulu membuat persamaan orde pertama, berikut persamaan model matematisnya :

$$\begin{aligned} x_1 &= y \\ x_2 &= \dot{y}. \\ &. \\ &. \\ x_n &= y^{n-1} \end{aligned} \qquad (2.11)$$

Sehingga persamaan (2.4) dapat ditulis sebagai :

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
&. \\
&. \\
&. \\
\dot{x}_{n-1} &= x_n \\
\dot{x}_n &= -a_n x_1 - \ldots - a_1 x_n + u
\end{aligned} \qquad (2.12)$$

Atau

$$\dot{x} = Ax + Bu \qquad (2.13)$$

Di mana

$$X = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix}; \quad A = \begin{bmatrix} 0 & 1 & 0 & . & . & 0 \\ 0 & 0 & 1 & . & . & 0 \\ . & . & . & & & . \\ . & . & . & & & . \\ 0 & 0 & 0 & . & . & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & . & . & -a_1 \end{bmatrix}; \quad B = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 1 \end{bmatrix} \qquad (2.14)$$

Dan persamaan keluarnya menjadi

$$y = [1 \quad 0 \quad . \quad . \quad . \quad 0] \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} \qquad (2.15)$$

Atau

$$y = Cx \qquad (2.16)$$

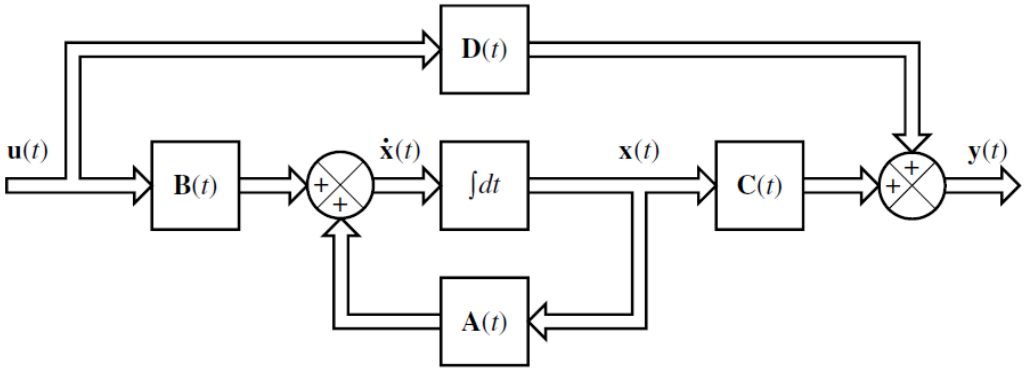
dimana

$$C = [1 \quad 0 \quad . \quad . \quad . \quad 0] \qquad (2.17)$$

Berikut adalah persamaan sistem yang bervariasi terhadap waktu yang dilinearkan terhadap persamaan keadaan (*state space) :*

$$\begin{aligned}
x(t) &= A(t) + B(t)u(t) \\
y(t) &= C(t)x(t) + D(t)u(t)
\end{aligned} \qquad (2.18)$$

Persamaan keadaan di atas dapat digambarkan dalam blok diagram seperti pada Gambar 2.4 :



Gambar 2. 2. Diagram Blok Sistem Kontrol Persamaan Ruang Keadaan

Prinsip dasar metode *Linear Quadratic Regulator* adalah mendapatkan sinyal kendali optimal dari umpan balik keadaan(*state feedback*). Persamaan *differensial* orde pertama persamaan (2.10) adalah persamaan keadaan dan persamaan aljabar, persamaan (2.11) adalah persamaan keluaran.

Pada perancangan sistem pengendalian optimal diperlukan adanya kriteria optimasi yang dapat meminimalkan suatu indeks unjuk kerja dengan cara melakukan pengukuran. Untuk melakukan pengukuran tersebut diperlukan suatu tolak ukur yang disebut *indeks performansi* atau indeks kinerja.

Indeks kinerja adalah suatu nilai yang menunjukkan tingkat kebaikan kinerja sistem, dan suatu sistem kontrol dikatakan optimal jika indeks performansinya minimum, untuk menentukan *indeks performansi* minimum atau maksimum tergantung keadaannya. Dalam hal ini *quadcopter* akan digunakan sebagai objek sistem kontrol, pendekatan sistem kontrol yang digunakan pada *quadcopter* ini adalah sistem kontrol modern, karena pada dasarnya menggunakan beberapa masukan sensor untuk melakukan pengendalian dan pendekatannya adalah *time domain*.

Selah mengetahui konsep dasar teori optimal selanjutnya adalah membahas mengenai metode *Linear Quadratic Regulator,* pada desain *LQR* dengan menggunakan persamaan matematis yang disebut *cost functional* yaitu waktu integral dari bentuk kuadratik pada vektor keadaan x dan vektor masukan u seperti persamaan berikut :

$$J = \int_{\infty}^{0}(x^t Q x + u^t u R)dt \qquad (2.19)$$

di mana *Q* adalah matriks semi definit positif dan R adalah vektor variabel kontrol atau matriks definit positif, berdasarkan persamaan tersebut variasi parameter dari masalah perancangan *LQR* dapat di tentukan. Langkah pertama yang dilakukan adalah memilih matriks bobot nilai *Q* dan *R*. Berikut adalah persamaan-persamaan yang digunakan dalam pengendalian optimal. Dengan sistem :

$$\begin{aligned} x &= Ax + Bu \\ y &= Cx + Du \end{aligned} \qquad (2.20)$$

Prinsip metode *LQR* adalah memperoleh sinyal kendali optimal dari umpan balik keadaan *(state feedback)*, dan hukum pengendalian optimal adalah :

$$u(t) = -Kx(t) \quad (2.21)$$

Umpan balik K adalah konstanta yang dihitung menggunakan persamaan aljabar *Riccati*, dari persamaan *state space* sistem dan indeks performansi didapat nilai K yang optimal untuk indeks performansi sebagai berikut :

$$K = -R^{-1}B^{T}P \quad (2.22)$$

Di mana P adalah unik, solusi semidefinit untuk persamaan *Riccati* harus memenuhi persamaan tereduksi berikut :

$$B^{t}P + PA + Q - PBR^{-1}B^{T}P = 0 \quad (2.23)$$

Persamaan (2.22**)** memberikan matriks K optimal, hukum kontrol optimal kuadratik yang indeks kinerjanya *(index performance)* diberikan oleh persamaan (2.19) adalah linear dan didapatkan dari substitusi persamaan (2.21) dan (2.22):

$$u(t) = -Kx(t) = -R^{-1}B^{T}Px(t) \quad (2.24)$$

Persamaan (2.24**)** dikenal sebagai persamaan *Algebraic Riccati Equation (ARE)*, solusi persamaan tersebut hanya dapat diselesaikan bila pasangan matriks (A,B) adalah *controllable* dan pasangan matriks (A,C) adalah *observable*. Untuk menguji pasangan matriks ini dilakukan pada matlab, yaitu dengan perintah dibawah ini:

$$CO = rank\ (ctbr(A,B)) \quad (2.25)$$

Hasil dari perintah diatas ada seperti gambar 2.3 dibawah ini.

```
>> rank(ctrb(A,B))

ans =

     8

>>
```

Gambar 2. 3 Keterkendalian Matriks

Sesuai dengan tujuan dari sistem kontrol optimal yaitu mendapatkan nilai *feedback* yang optimal yang akan meminimalkan *cost function* J, agar sinyal *feedback* optimal maka dibutuhkan matriks pembobotan Q dan matriks R.

Penentuan dari matriks pembobotan Q dan R pada sistem ini menggunakan Bryson's rule (Navajas & Raad, 2015), yaitu :

$$Qii = \frac{1}{\max acceptable\ value\ of\ {x_i}^2}$$
$$Rii = \frac{1}{\max acceptable\ value\ of\ {U_i}^2} \quad (2.26)$$

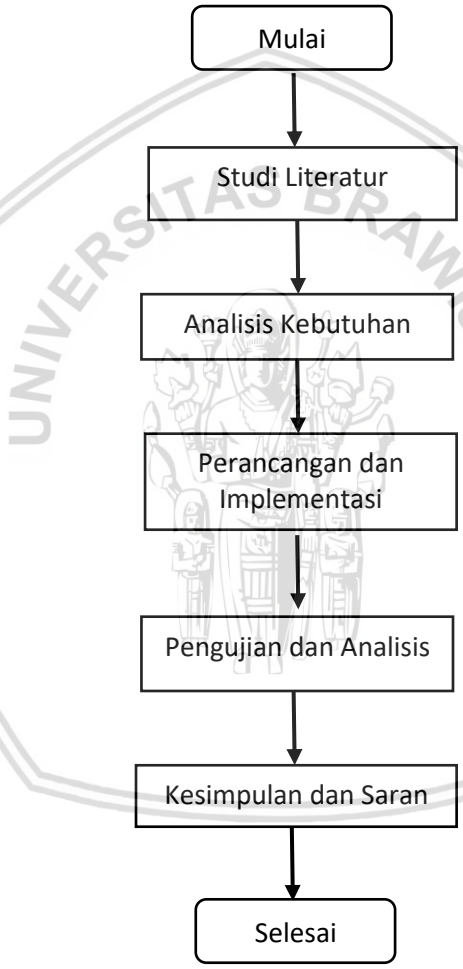
Perhitungan menggunakan metode ini tidak menjamin sistem sesuai dengan yang kita inginkan, tetapi mempermudah untuk memulai proses *trial* dan *error*.

BAB III METODOLOGI PENELITIAN

3.1. Metode Penelitian

Pada penelitian ini, tipe penelitian yang digunakan adalah bersifat implementatif, Karena tidak hanya menjelaskan sistem kecepatan pada *quadcopter*, tetapi juga melakukan perancangan dan menerapkan sistem yang telah dibangun pada *quadcopter*.

Dalam melakukan suatu penelitian perlu adanya suatu metodologi, pada penelitian ini mtodologi yang digunakan adalah seperti pada Gambar 3.1.



Gambar 3. 1.Metodologi pengerjaan penelitian

Pada gambar 3.1 diatas adalah proses yang akan dilakukan pada saat melakukan penelitian, untuk mengurangi kesalahan yang terjadi dalam proses pengerjaan penelitian, ada baiknya dikerjakan dengan langkah demi langkah, hal ini meruakah alasan yang membuat peneliti menggunakan metode ini dalam proses pengerjaan skripsi.

Untuk penjelasan dari masing-masing langkah dalam penelitian ini dijelaskan dalam sub-bab dibawah ini, menurut bab masing-masing.

3.2. Studi Literatur

Untuk melakukan perancangan dan implementasi suatu sistem, maka dalam penelitian ini ada nya suatu studi literature. Literatur merupakan suatu dasar teori, teori penguat serta suatu landasan dasar dalam sebuah penelitian. Teori pendukung dapat diperoleh dari buku, jurnal, dan internet. Adapun berapa literatur yang digunakan dalam penelitian ini adalah:

1. UAV *Quadcopter*
2. Konsep Sistem kontrol pada penggunaan metode linear quadratic regulator (*LQR*).

3.3. Analisis Kebutuhan

Pada sub bab ini akan dijelaskan mengenai gambaran umum sistem serta analisis kebutuhan sistem. Gambaran umum sistem merupakan suata gambaran mengenai cara kerja sistem. Sedangkan analisis kebutuhan dilakukan dengan melakukan identifikasi kebutuhan sistem, kebutuhan fungsional dan kebutuhan non-fungsional. Pada penelitian ini ada beberapa kebutuhan yang digunakan adalah kebutuhan perangkat lunak. Kebutuhan perangkat lunak sangat lah diperlukan baik untuk perancangan, pengujian ataupun implementasi. Perangkat lunak yang dibutuhkan dalam pembuatan sistem ini mempunyai spesifikasi yang dapat melakukan komputasi matematis ataupun simulasi perancangan seperti matlab, Bahasa pemograman *python*, ROS, Gazebo, dan Tum Simulator.

Selanjutanya adalah kebutuhan perangkan keras. Kebutuhan perangkat keras yang dibutuhkan dalam melakukan penelitian ini adalah yang digunakan untuk proses implementasi algorithma yang sudah ditentukan. Adapun perangkat keras yang digunakan adalah seperti: Dekstop PC, dan *quadcopter* parrot ar drone 2.0. Kemudian pada kebutuhan fungsional akan dijelaskan bagaimana sistem akan bekerja untuk mencapai target yang diinginkan, pada penelitian ini kebutuhan fungsional adalah *quadcopter* dapat mencapai jarak yang telah ditentukan dengan kecepatan maksimal dan stabil. Kebutuhan non-fungsional dalam penelitian ini terdapat 2 bagian yaitu untuk keamanan dan dan *performa* dari sistem.

3.4. Perancangan dan Implementasi

Perancangan sistem adalah tahapan untuk membangun sistem yang dibutuhkan dalam melakukan penelitian ini agar kebutuhan fungsional terpenuhi. Perancangan pada sistem ini dibagi menjadi perancangan komunikasi sistem yang meliputi hubungan ROS, Ubuntu, gazebo dan *quadcopter*, dan perhitungan *LQR* pada *quadcopter.*

Kemudian setelah selesai tahap perancangan, akan dilanjutkan dengan implementasi yaitu proses realisasi sitem. Tahap awal dari implementasi adalah komunikasi antara ROS, Ubuntu dan gazebo agar program dapat dijalankan. Tahap

berikutnya adalah implementasi *LQR* pada *quadcopter* dengan menggunakan *source code* yang telah dibuat. Source *code* berfungsi untuk *quadcopter* dapat bergerak dengan kecepatan yang stabil. Program dapat dijalankan pada simulasi untuk mengetahuai sistem dapat bekerja dengan baik sebelum dijalankan secara langsung pada *quadcopter.*

3.5. Pengujian dan Analisis

Quadcopter yang sudah dirancang dengan suatu langkah-langkah dan hasil pengujian ini nantinya akan di analisis sesuai dengan rumusan masalah yang sudah dijabarkan. Pengujian algoritma untuk kecepatan pada *quadcopter* yang sudah diimplementasikan dilakukan dengan berbagai jenis pengujian.

Analisis hasil pengujian dilakukan setelah semua jenis pengujian telah selesai dilakukan. Berdasarkan pada rumusan masalah, kinerja adalah variabel yang akan di analisis dari hasil pengujian ini. Kinerja yang dimaksud pada analisis ini adalah, waktu yang dibutuhkan (setling time) untuk mencapai keadaan yang di inginkan.

3.6. Kesimpulan dan Saran

Penarikan kesimpulan dan saran dilakukan jika semua tahapan analisis kebutuhan, perancangan dan implementasi, pengujian sistem selesai dilakukan berdasarkan rumusan masalah dan tujuan dari penelitian ini. Harapannya kesimpulan yang sudah didapatkan dapat dipergunakan untuk mengatasi masalah kecepatan pada *quadcopter*. Tahapan terakhir dari penulisan adalah saran yang bertujuan untuk menyempurnakan penulisan dan pengembangan sistem lebih lanjut.

BAB IV ANALISIS KEBUTUHAN

4.1 Gambaran Umum Sistem

Sistem ini akan bekerja sesuai dengan diagram blok 4.1. Proses yang terjadi dalam sistem ini adalah program yang telah dibuat dalam komputer dan data akan diteruskan menuju *quadcopter* melalui jaringan *Wi-fi.*

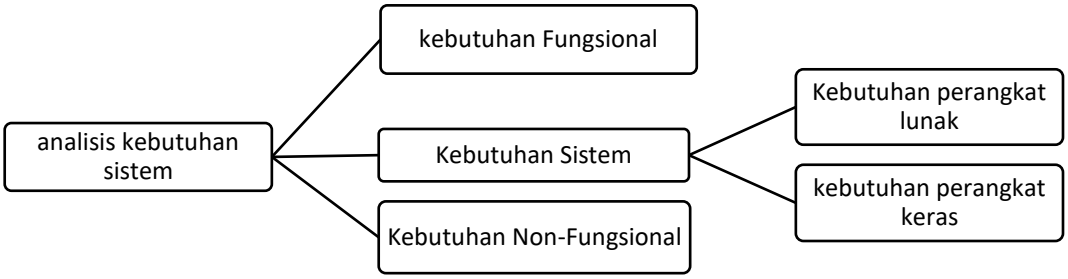
Gambar 4. 1.Diagram blok komunikasi sistem

Pada sistem ini kendali navigasi *quadcopter* ini terdapat pada inputan dari sistem. Dimana input pada sistem ini adalah berupa jaraka yang telah ditentukan oleh pengguna. Output dari sistem ini adalah berupa kecepatan dari *quadcopter*. Kemudian pemograman yang digunakan untuk mengirim data dari computer ke *quadcopter* adalah python. Selain itu *quadcopter* juga dapat menampikan info data navigasi dari *quadcopter*.

Penelitian ini dilakukan untuk mengetahui pengaruh penggunaan metode *Linear Quadratic Regulator (LQR)* terhadap pengendalian kecepatan pada *quadcopter*. Dalam penelitian ini dilakukan pengujian secara realtime dan juga simulasi pada ROS.

4.2 Analisis Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk mengetahui apa saja kebutuhan yang diperlukan sistem pada penelitian ini. Pada pembahasan ini akan dibagi menjadi dua bagian lagi yaitu analisis kebutuhan pengguna dan analisis kebutuhan sistem yang akan dijabarkan lagi berupa perangkat keras dan perangkat lunak sesuai dengan gambar 4.2.



Gambar 4. 2.Analisis Kebutuhan Sistem

4.3 Kebutuhan Fungsional

Kebutuhan yang harus dipenuhi pada penelitian ini adalah kebutuhan fungsional, karena kebutuhan ini adalah bagaimana sistem yang dibangun harus seperti sistem yang diinginkan. Adapun kebutuhan fungsional yang dibutuhkan adalah sebagai berikut:

1. *Quadcopter* dapat bergerak maju dengan menggunakan sistem kontrol *Linear Quadratic Regulator* dengan kecepatan yang stabil, dalam hal ini yang diharapkan oleh user adalah *quadcopter* mampu bergerak dengan kecepatan yang sesuai dengan kecepatan yang ditentukan oleh pengguna tanpa adanya perubahan yang ada.
2. Sistem dapat mencapai jarak yang di *input* oleh pengguna dengan tingkat ketepatan yang sesuai dengan *input,* yaitu untuk jarak yang telah ditetapkan oleh pengguna, *quadcopter* diharapkan mampu mencapai jarak yang ditentukan oleh *user* tanpa adanya jarak yang berbeda dengan yang telah ditetapkan oleh *user.*

4.4. Kebutuhan Sistem

Pada tahap ini, ada dua analisis kebutuhan sistem yaitu, kebutuhan perangkat keras dan kebutuhan perangkat lunak.

4.4.1. Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang dibutuhkan dalam pembuatan sistem ini adalah sebagai berikut:

a. *Parrot AR.Drone 2.0*

Parrot AR.Drone 2.0 adalah sebuah perangkat yang berbentuk *quadcopter* yang dikembangkan oleh sebuah perusahaan di perancis yaitu perusahaan *parrot* yang dapat dikontrol dari jarak yang jauh. *Quadcopter* dirancang untuk dapat dikendalikan oleh operasi *mobile* atau PC/Tablet seperti android dan iOS. P*arrot AR.Drone* diresmikan pertama kali pada tahun 2010 yang dilaksanakan di CES Internasional di Las Vegas. Pada tahun 2012 *Parrot AR.Drone* dipublikasikan dengan versi *2.0* dengan meningkatkan fungsi *quadcopter*.

Pada *Parrot AR.Drone* versi 2.0 memiliki kualitas kamera HD yaitu 720p 30fps dan banyak sensor yang dipasang pada badan *quadcopter* yang menjadikan *quadcopter* lebih sensitive dan bisa dikendalikan seperti *accelorometer, gyroscope*, *magnetometer*, *pressure sensor,* dan *altitude sensor*. *Motherboard* yang telah digunakan pada *Parrot AR.*Drone versi 2.0 ini adalah prosesor 1Ghz 32 bit ARM *Cortex* A8 processor with 800MHz video DSP TMS320DMC64x dan menggunakan system operasi linux dan memiliki RAM sebesar 1GB. *Parrot AR.Drone* versi 2.0 ini juga menggunakan *Wi-Fi* dengan jangkauan maksimal sebesar 10 meter untuk melakukan komunikasi.

Gambar 4. 3. *Parror AR.Drone* versi 2.0

Berdasarkan spesifikasi yang dimiliki oleh *Parrot AR.Drone* dipilih sebagai *quadcopter* dalam penelitian ini. Dengan sistem operasi linux yang tertanam pada *Parrot AR.Drone* membuat *quadcopter* ini mudah di program. Dan dengan spesifikasi RAM 1GB dan prosesor yang dimiliki menjadikan *quadcopter* ini dengan cepat memproses intruksi yang telah diberikan oleh pengguna. Selain itu dari segi keamanan *quadcopter* ini memiliki *hull* atau pelindung di keseluruhan baling-baling seperti pada gambar 4.3, sehingga sangat aman digunakan khususnya untuk penelitian.

Dalam sisi *quadcopter*, ada beberapa perangkat atau modul yang akan digunakan dalam penelitian ini yaitu:

1. Mainboard AR. Drone Parrot 2.0

Main board AR. Drone 2.0 merupakan sebuah perangkat yang berguna dalam melakukan pemrosesan data, untuk menghubungkan user dan *quadcopter*, serta sebagai pengendali semua komponen yang ada pada *quadcopter*, dan menggunakan sistem operasi linux yang dapat diakses dengan telnet menggunakan komputer/ laptop.

2. Gyroscope

Merupakan komponen berupa sensor yang ada pada *quadcopter* yang berfungsi untuk mengukur atau mempertahankan ketetapan momentum sudut, dan sensor ini yang berfungsi untuk menghitung nilai *pitch, roll,* dan *yaw*. Berdasarkan perhitungan ini lah *quadcopter* dapat melakukan pergerakan baik pergerakan maneuver maupun untuk menstabilkan posisi *quadcopter*.

b. Komputer / Laptop

Pada penelitian ini komputer digunakan sebagai perangkat untuk mengirimkan perintah- perintah pada *quadcopter* baik sebagai navigasi mapun untuk menuliskan kode program pada *quadcopter* melalui komunikasi *wi-fi* antar komputer dan *quadcopter*. Untuk sistem operasi yang digunakan dalam penelitian ini adalah Ubuntu 14 dan ROS Indigo sebagai *framework* untuk pengembangannya. Pada penelitian ini digunakan komputer/ laptop dengan sistem operasi Ubuntu karena pada *quadcopter* telah tertanam sistem operasi

berbasis linux atau ubuntu, sehingga mempermudah dalam komunikasi antara *user* dengan *quadcopter*.

Gambar 4. 4. Komputer / Laptop yang digunakan

4.4.2. Kebutuhan Perangkat Lunak

Untuk perangkat lunak yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Sistem Operasi *Linux Ubuntu* versi 14.04

Ubuntu merupakan sistem operasi yang bersifat *open source*, bebas, terbuka sehingga tidak perlu biaya untuk mendapatkan lisensinya. Ubuntu memiliki sistem keamanan yang unggul dibandingkan dengan sistem operasi yang lain, dan untuk penggunaan lebih mudah. Pada penelitian ini menggunakan *Ubuntu* versi 14.04 karena versi ini merupakan versi yang paling stabil dalam pengunaan yang akan dapat terhubung dengan ROS tanpa ada kendala.

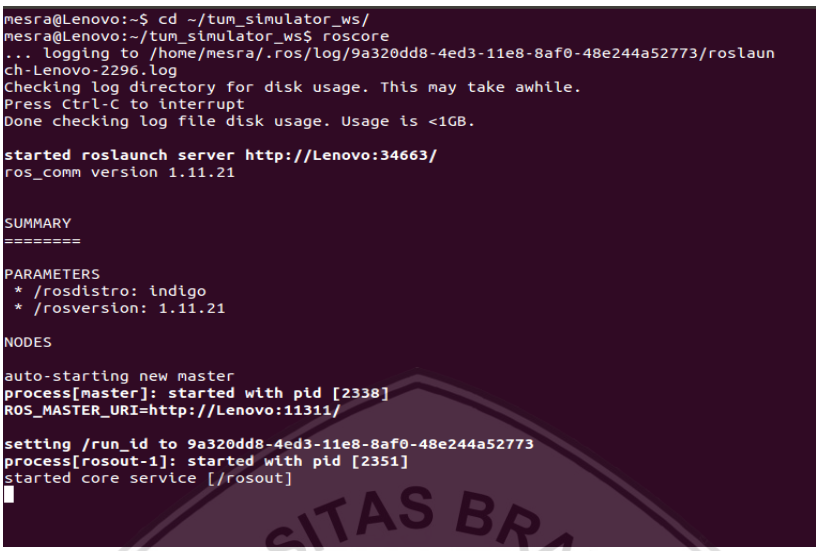
2. Matlab

Matlab adalah singkatan dari *MATrix LABoratory*, merupakan bahasa pemrograman yang dikembangkan oleh The Mathwork Inc. yang hadir dengan fungsi dan karakteristik yang berbeda dengan bahasa pemrograman lain yang sudah ada lebih dahulu seperti *Delphi, Basic* maupun *C++.* Matlab merupakan bahasa pemrograman level tinggi yang dikhususkan untuk kebutuhan komputasi teknis, visualisasi dan pemrograman seperti komputasi matematik, analisis data, pengembangan algoritma, simulasi dan pemodelan dan grafik-grafik perhitungan. Matlab umumnya digunakan untuk mempermudah dalam perhitungan, pemodelan, simulasi dalam pembuatan *prototype*.

3. ROS

Robot Operating System (ROS) yang berfungsi sebagai sistem operasi yang akan menjalankan *quadcopter* dengan cara menghubungkan *ROS* dengan *quadcopter* melalui koneksi *Wi-fi.* Sistem operasi ini merupakan salah satu kebutuhan yang sangat penting pada penelitian ini karena sistem operasi ini

merupakan *framework* yang digunakan untuk menghubungkan antara *user* dan *quadcopter.* Di dalam ROS terdapat *driver, tools, library* yang digunakan untuk mempermudah dalam melakukan pemograman robot yang bersifat kompleks.

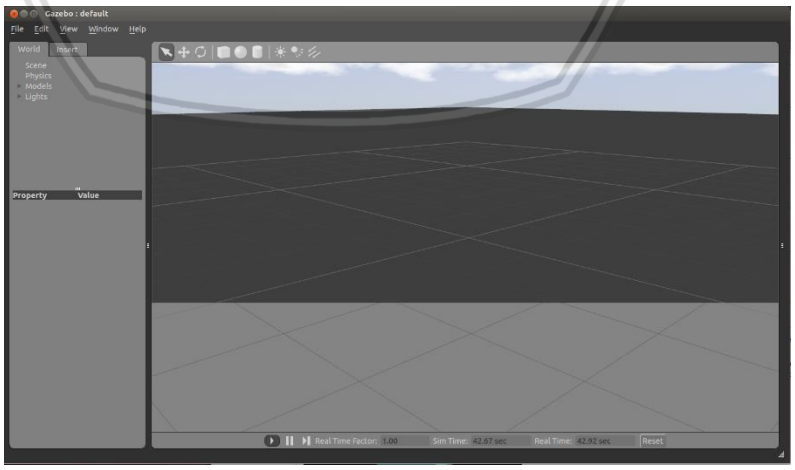


Gambar 4. 5. Tampilan *ROS CORE*

Pada gambar 4.5 merupakan tampilan ketika *ROS CORE* dijalankan, ROS dalam hal ini tidak memiliki *Graphic User Interface (GUI)* seperti sistem operasi lainya. Untuk mengoperasikan sistem operasi ini ROS user hanya menggunakan *terminal console* dengan *syntax*.

4. Gazebo

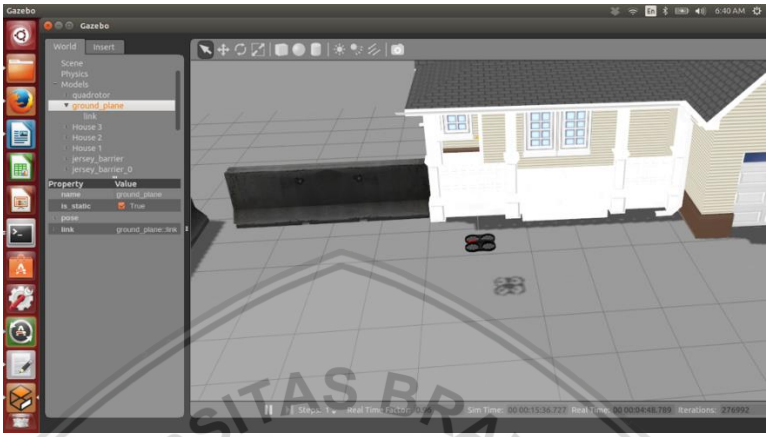
Gazebo merupakan perangkat lunak yang digunakan sebagai *simulator* dari sistem kendali yang dibangun, hal ini dilakukan agar saat implementasi ke perangkat aktual, tidak ada kesalahan.Pada gambar 4.6 merupakan tampilan awal Gazebo yang masih kosong, yang belum menggunakan simulator *Ar Drone*.



Gambar 4. 6. Tampilan Gazebo

5. Tum Simulator

Perangkat lunak yang digunakan sebagai *simulator* dari *quadcopter* dengan jenis *Parrot AR Drone 2.0* yang akan mempermudah dalam penelitian. Pada gambar 4.7 merupakan tampilan tum simulator yang akan digunakan untuk melakukan simulasi *quadcopter* ar drone. Penggunaan simulator ini sama dengan *quadcopter* sesungguhnya, sehingga untuk implementasi pada *quadcopter* sesungguhnya tidak memerlukan program yang baru.



Gambar 4. 7. Tampilan Tum Simulator

4.5. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional pada pengembangan sistem ini adalah sebagai berikut :

1. *Performance*

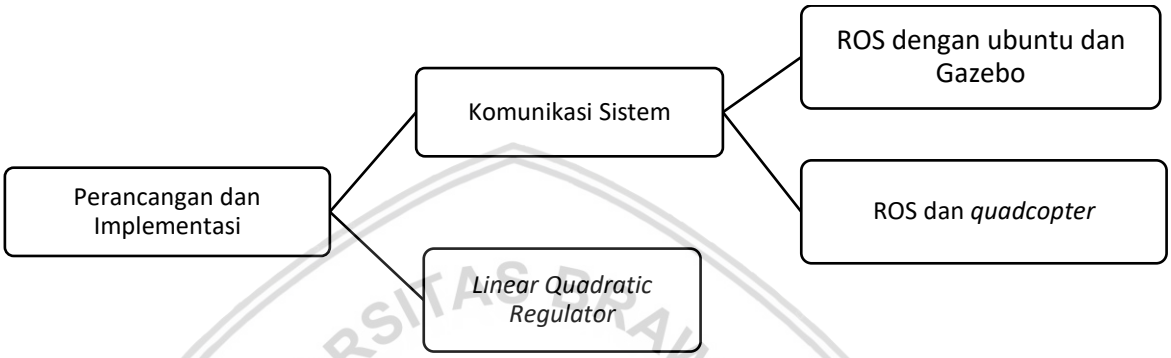
   Kebutuhan performa adalah kebutuhan yang berhubungan dengan kinerja dari sistem kendali yang diimplementasikan pada kecepatan *quadcopter Parrot Ar Drone 2.0,* kinerja dari sistem akan terukur dengan parameter *Setling Time,* yaitu waktu yang dibutuhkan oleh sistem untuk dapat mencapai nilai yang di inginkan.

2. *Safety*

   Kebutuhan keselamatan pada sistem ini adalah mencakup keselamatan dari *quadcopter* ketika *quadcopter* bergerak maju dengan kecepatan yang stabil untuk mencapai jarak tertentu, dan ketika sudah mencapai jarak yang diinginkan *quadcopter* akan secara otomatis *landing* untuk mencegah kecelakan yang ada jika ada halangan pada jarak yang melebihi jarak yang telah di *input* oleh user.

BAB V PERANCANGAN DAN IMPLEMENTASI

Pada bab ini adalah menjelaskan mengenai tahapan perancangan yang dilakukan sebelum program di implementasikan pada *quadcopter.* Tahapan perancangan sistem secara keseluruhan pada penelitian ini dapat dilihat pada gamabr 5.1. Pertama dilakukan perancangan komunikasi pada sistem. Komunikasi pada sistem terdiri dari empat buah komponen yaitu, ROS, Ubuntu, Gazebo dan komputer. Dan dilakukan perancangan *Linear Quadratic Regulator (LQR)* yaitu melakukan penghitungan *LQR* untuk dilakukan implementasi pada *quadcopter.*



Gambar 5. 1.Tahapan Perancangan dan Implementasi

Setelah dilakukan perancangan terhadap kecepatan *quadcopter* dengan tujuan untuk menstabilkan kecepatan pada *quadcopter* dan untuk mendapatkan tingkat ketepatan jarak yang sesuai dengan keinginan pengguna dan akan *landing* secara otomatis ketika sudah mencapai jarak yang digunakan. Setelah tahapan perancangan dan implementasi dilakukan akan menghasilkan hubungan yang akan membangun proses dari sistem secara keseluruhan. Hubungan tersebut sesuai dengan gambar 5.1.

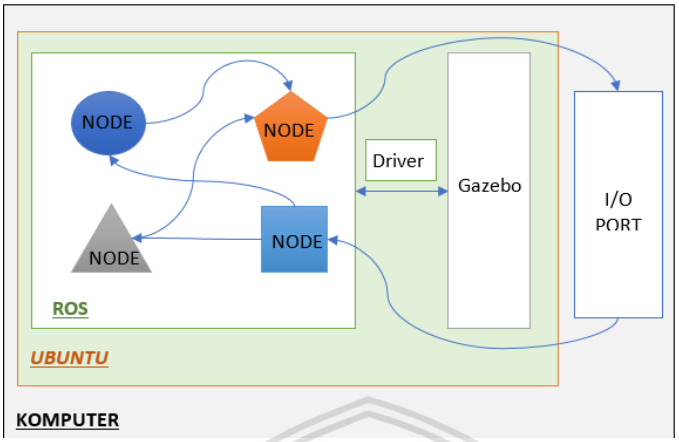
5.1. Komunikasi Sistem

Pada perancangan sistem ini memudahkan pemahaman terhadapam perancangan sistem secara keseluruhan. Pada tahap ini terdapat dua jenis perancangan sistem yang digunakan dalam penelitian ini yaitu perancangan dan implementasi ROS, Ubuntu dan gazebo, serta perancangan dan implementasi ROS dan quadcopter.

5.1.1. Perancangan Komunikasi ROS dengan Ubuntu dan Gazebo

Pada perancangan sistem ini, untuk mempermudah pemahaman terhadap perancangan sistem antara ROS, Ubuntu dan gazebo. ROS adalah sistem operasi yang digunakan untuk membuat program robot yang fleksibel dan mudah digunakan. ROS dapat digunakan untuk semua jenis robot yang ada. ROS dilengkapi dengan fasilitas simulator untuk meminimlakan resiko yang terjadi ketika menjalankan suatu algorithma pada sebuah robot. Ubuntu merupakan suatu sistem *platform* yang digunakan agar ROS dapat dijalankan atau digunakan. Node yang ada didalam ROS merupakan suatu perintah untuk menjadi *subscriber*

ataupun *publisher*. Untuk perancangan sistem antara ROS, Ubuntu dan gazebo seperti pada gambar 5.2.



Gambar 5. 2. Hubungan antara ROS, Ubuntu dan gazebo

Pada gambar 5.2 merupakan hubungan antara ROS, Ubuntu dan gazebo menggambarkan hubungan antara Ros dan Ubuntu yang dihubungkan dengan menggunakan *driver*, kemudian untuk keluaran dan masukan akan ditampilkan pada komputer, yang dimana ROS akan memberikan data yang telah diolah oleh sistem, dan untuk *inputan* yang ada pada komputer akan dikirim kembali melalui node yang ada pada ROS.

5.1.2. Implementasi Komunikasi ROS dengan Ubuntu dan Gazebo

Bagian ini akan membahas bagaimana alur dari bentuk implementasi komunikasi sistem pada penelitian yang dilakukan. Untuk langkah awal yang dilakukan adalah menjalankan ROS diatas Ubuntu dengan menggunakan terminal dengan perintah:

```
$  Roscore
```

Hasil tampilan dari perintah diatas adalah seperti gambar 5.3 berikut:

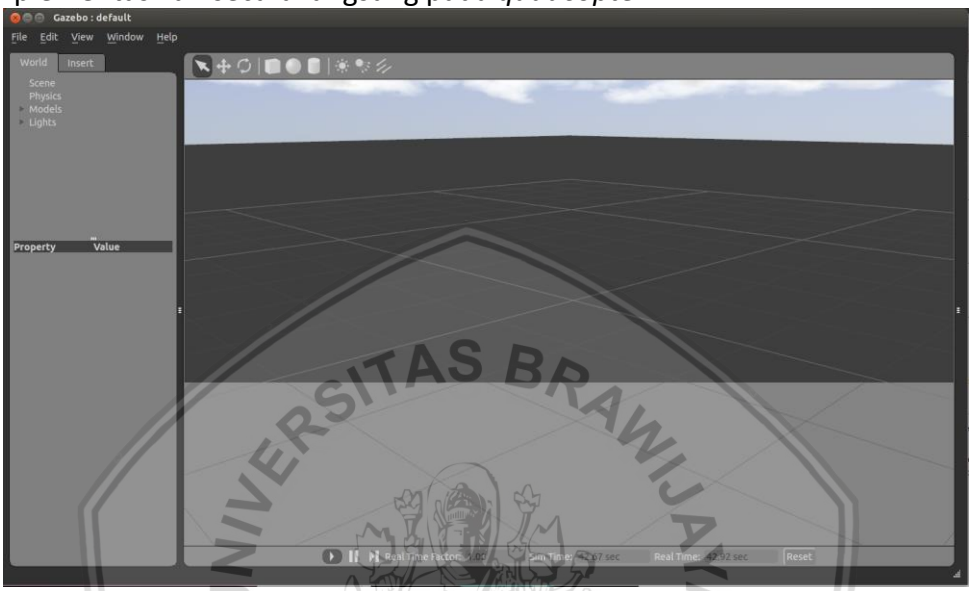
```
mesra@Lenovo:~$ cd ~/tum_simulator_ws/
mesra@Lenovo:~/tum_simulator_ws$ roscore
... logging to /home/mesra/.ros/log/9a320dd8-4ed3-11e8-8af0-48e244a52773/roslaun
ch-Lenovo-2296.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://Lenovo:34663/
ros_comm version 1.11.21


SUMMARY
========

PARAMETERS
 * /rosdistro: indigo
 * /rosversion: 1.11.21

NODES

auto-starting new master
process[master]: started with pid [2338]
ROS_MASTER_URI=http://Lenovo:11311/

setting /run_id to 9a320dd8-4ed3-11e8-8af0-48e244a52773
process[rosout-1]: started with pid [2351]
started core service [/rosout]
```

Gambar 5. 3. ROS saat dijalankan

Dan untuk menjalankan Gazebo di atas *ubuntu* menggunakan perintah:

```
$  Gazebo
```

Pada gambar 5.4 merupakan tampilan awal untuk memulai menggunakan gazebo. Gazebo ini digunakan untuk melakukan simulasi program yang telah dibuat untuk mengurangi resiko terjadinya hal-hal yang tidak diinginkan, jika ingin diimplementasikan secara langsung pada *quadcopter.*



Gambar 5. 4. Gazebo Simulator

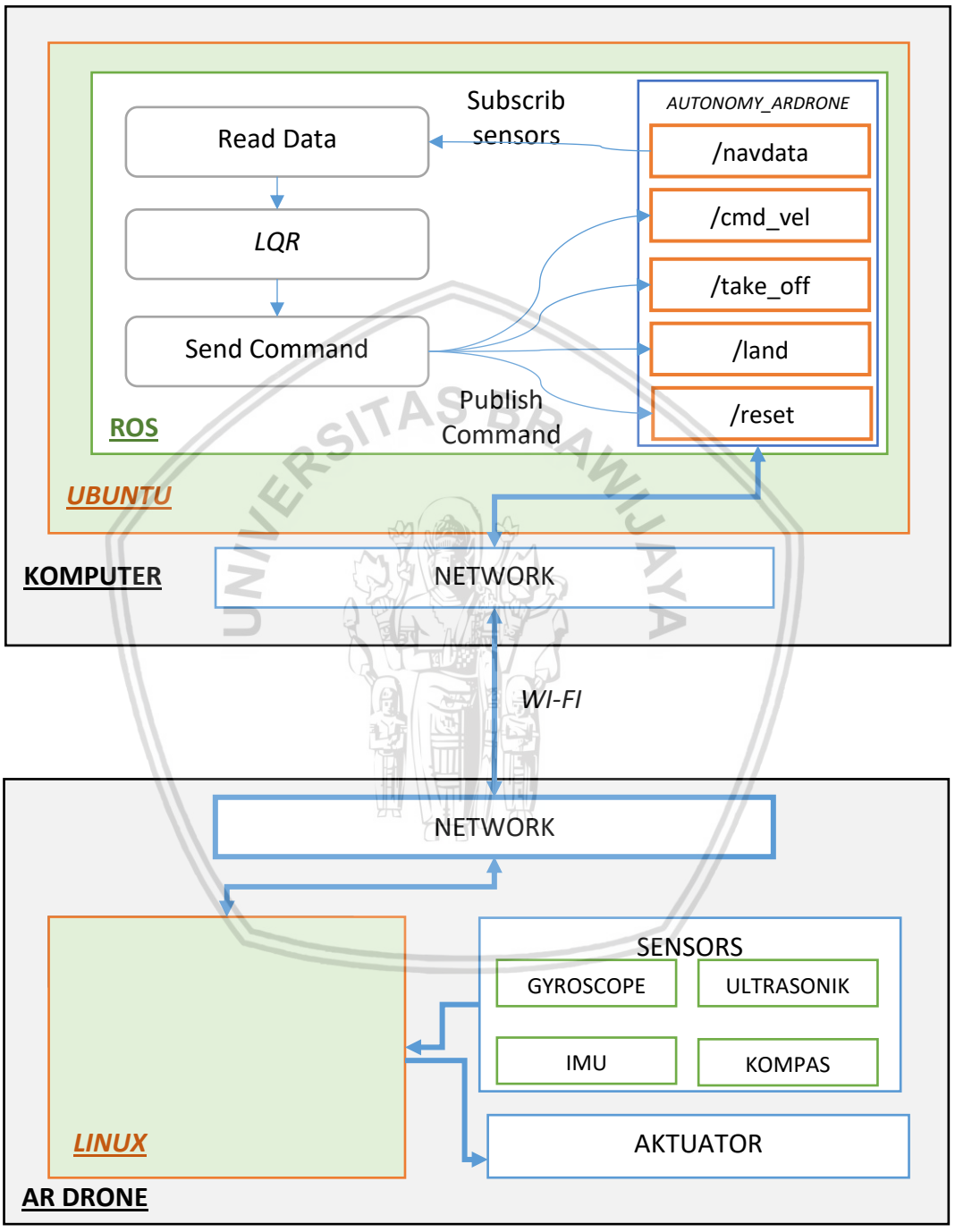
5.1.3. Perancangan Komunikasi ROS dengan *Quadcopter Ar Drone*

Parrot- Ar Drone merupakan jenis *quadcopter* yang memiliki sistem operasi *linux* yang dapat dikembangkan pada *ROS. ROS* dapat mengenali sistem dari *Ar Drone*, sehingga diperlukan sebuah *driver package* yang dinamakan *autonomy Ar Drone* agar sistem dapat dikendalikan, *driver* ini memungkinkan kita dapat mengakses data navigasi dari *quadcopter* dan mengirim *common* pada sistem. Pada *package autonomy ad drone* ada terdapat beberapa *library* atau berupa *command* yaitu, ada *navdata, cmd_vel, take off, landing dan reset. Navdata* berfungsi untuk mengirim data yang ada pada *quadcopter* yang akan di tampilkan pada PC, kemudian data tersebut akan diolah dalam perhitungan *LQR*. Hasil dari perhitungan tersebut akan dikirim kembali dalam bentuk pergerakan pada *quadcopter* sesuai dengan perintah yang ada pada program. Komunikasi antara PC dengan *quadcopter* untuk melakukan pengiriman data adalah menggunakan komunikasi *Wi-Fi,* sedangkan untuk menghubungkan ROS dengan sistem *quadcopter* menggunakan *driver autonomy Ar Drone. Qudacopter* memiliki beberapa sensor yang digunakan untuk memperoleh data yang diinginkan yaitu seperti *IMU, Gyroscope, ultrasonic* dan kompas.

Perbedaan komunikasi antara *tum simulator* dan *Ar Drone* aktual adalah, *tum simulator* dijalankan di atas *gazebo* dan berkomunikasi dengan *ROS* menggunakan

port, sedangkan untuk menghubungkan *Ar Drone* aktual dengan *ROS* menggunakan koneksi *wi-fi,* lalu menjalankan driver.

Untuk perancangan sistem ROS dengan *quadcopter Ar Drone* dapat dilihat seperti gambar 5.5 berikut:



Gambar 5. 5. Skema komunikasi Ar Drone dengan ROS

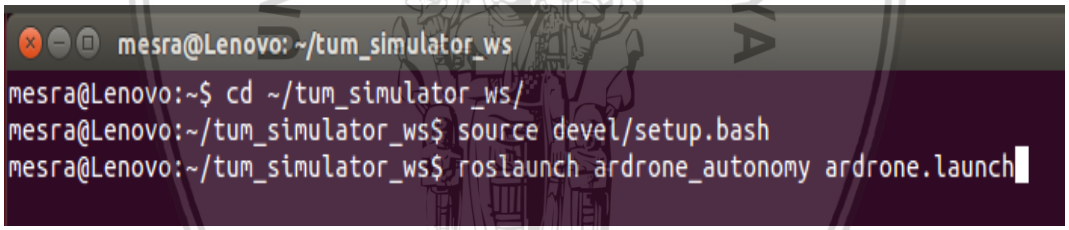
5.1.4. Implementasi Komunikasi ROS dengan *quadcopter AR Drone*

Bagian ini membahas bagaimana alur dari bentuk implementasi ROS dan *quadcopter AR Drone*. Untuk Implementasi komunikasi sistem ini ditunjukkan

pada gambar 5.6 sebagai langkah awal untuk menghubungkan komputer dengan *Wi-fi quadcopter* yang bernama “ArDrone_2”.

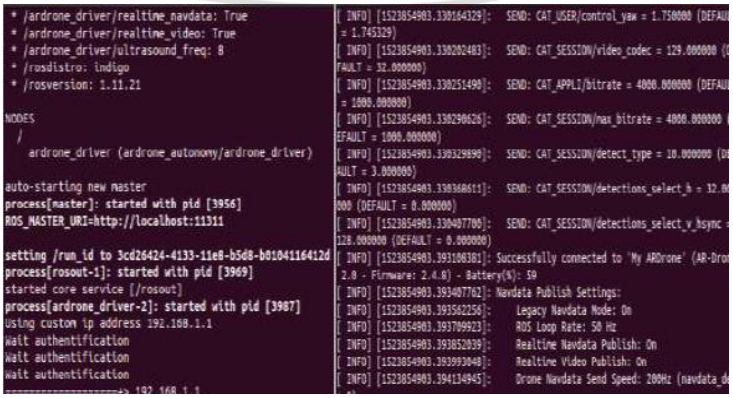
Gambar 5. 6. Menghubungkan komputer dengan *Wi-fi quadcopter*

Kemudian buka terminal dan ketikan *syntax* seperti pada Gambar 5.5 yang berfungsi untuk menghubungkan IP *quadcopter* dan juga pengiriman *navdata* dari *quadcopter* ke komputer agar dapat diakses oleh *user*.



Gambar 5. 7. Menghubungkan komputer dengan IP *quadcopter*

Tunggu beberapa saat hingga muncul pesan “connected” pada terminal sebagai tanda bahwa komputer telah terhubung dengan IP yang dimiliki oleh *quadcopter* sesuai pada Gambar 5.8.



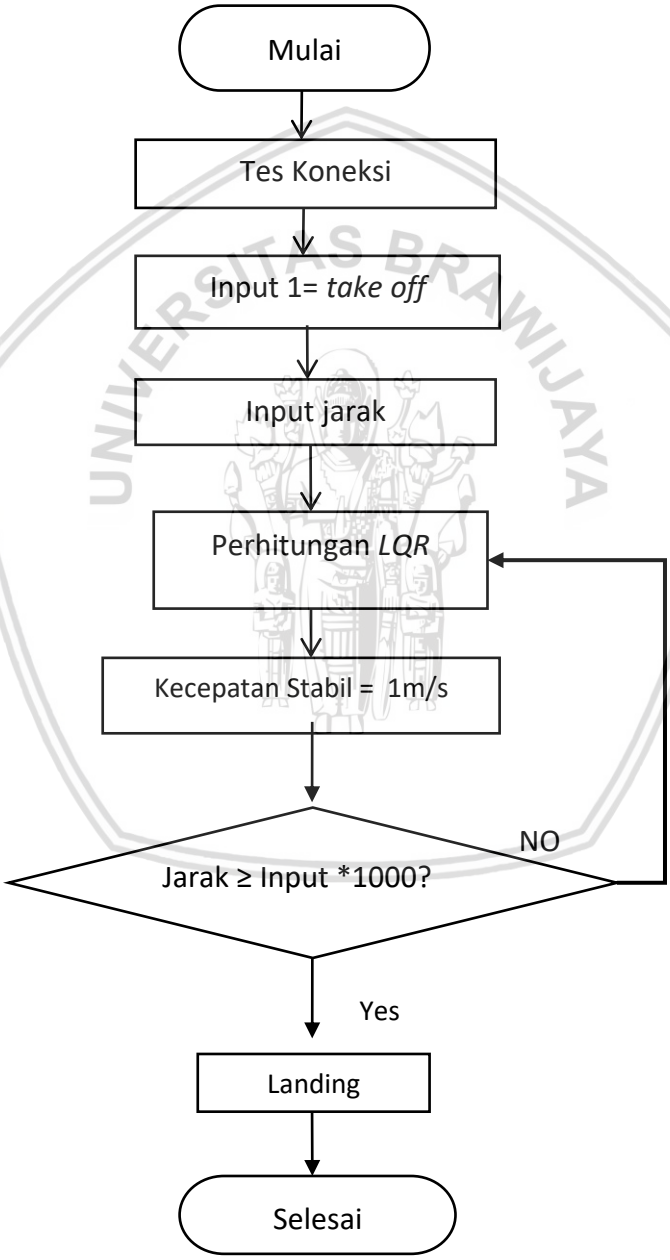
Gambar 5. 8.Komputer telah berhasil terhubung dengan *quadcopter*

Jika komputer telah terkoneksi dengan IP *quadcopter,* maka program akan dijalankan dan *quadcopter* akan bergerak hingga jarak yang telah ditentukan oleh pengguna.

5.2. Linear Quadratic Regulator (LQR)

5.2.1. Perancangan *Linear Quadratic Regulator (LQR)*

Dalam perancangan *LQR*, yang akan diperlukan menjadi *input* dalam sistem ini adalah jarak yang ditentukan oleh user. Jarak tersebut akan diolah pada sistem pengendalian *LQR.* Untuk proses berikutnya seperti gambar 5.9.



Gambar 5. 9. Flowchart Implementasi *LQR* Pada *Quadcopter*

Pada gambar 5.9 diatas merupakan langkah-langkah yang digunakan untuk perhitungan *LQR.* Untuk langkah awal dalam perancangan ini adalah dengan melakukan tes koneksi pada *quadcopter,* yang akan menghubungkan antara PC dan *Ar Drone* adalah *wi-fi* seperti yang ada pada sub bab (5.1.4). Selanjutnya mengirim *command takeoff* yaitu dengan menginputkan angka 1 dan akan meminta pengguna untuk melakukan *input* jarak yang diinginkan, pada pengujian yang akan dilakukan adalah dengan jarak 1m, 1.5m, 2m, 2.5m, dan 3m. Program *LQR* dijalankan jika posisi dari *quadcopter* sudah bergerak pada jarak yang sudah di tentukan sesuai dengan pengujian. Sistem pengendali akan melakukan perhitungan sesuai dengan algoritma. Ketika *quadcopter* sama dengan jarak yang telah diinputkan oleh *user* maka secara otomatis *quadcopter* akan landing

5.2.2.1.Melakukan *Take off*

Quadcopter akan terlebih dahulu melakukan *take off* untuk dapat melakukan semua program yang akan digunakan oleh *user*. Pada hal ini, jika ingin melakukan *take off* terhdap *quadcopter user* harus melakukan *input* menggunakn angka 1 agar dapat melakukan *take off.*Untuk *quadcopter* yang sudah melakukan *take off* dapat dilihat seperti gambar 5.10 berikut.

Gambar 5. 10. *Quadcopter* saat *take off*

5.2.2.2.*Input* Jarak

Dalam perancangan sistem menggunakan *LQR* perlu dilakukan *input* jarak yang diinginkan oleh pengguna. Dalam hal ini ada beberapa jarak yang digunakan oleh pengguna dalam sistem ini, yaitu 1m, 1.5m, 2m, 2.5m, dan 3m. Sistem ini akan mencapai jarak yang telah ditentukan dengan tingkat ketepatan yang sesuai dengan *input* yang telah dimasukkan oleh user setelah *quadcopter* telah melakukan *take off.* Untuk *quadcopter* yang telah mencapai jarak seperti pada gambar 5.11 berikut:

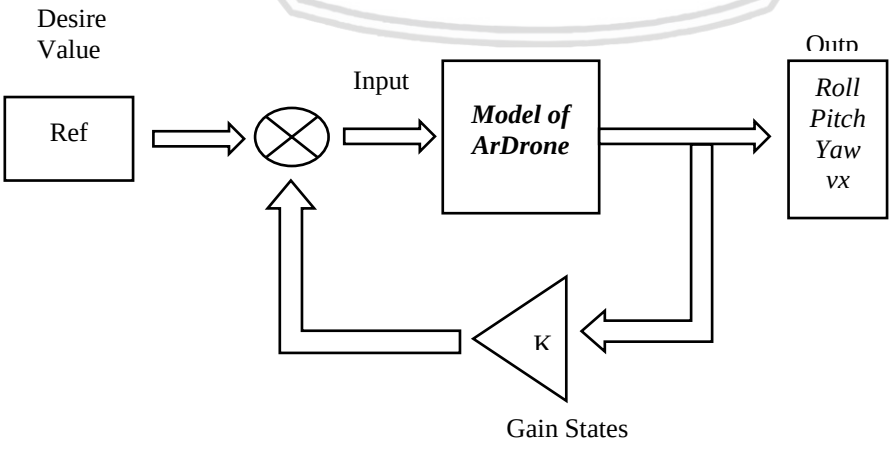
Gambar 5. 11. *Quadcopter* akan *landing* saat mencapai jarak yang ditentukan

5.2.2.3.Perhitungan *Linear Quadratic Regulator (LQR)*

Dalam perhitungan *lqr* ini ada beberapa tahap lagi yang akan dilakukan yaitu, inisialisasi matriks, perhitungan hitungan matriks, *A, B, C, D, Q, R* yang sesuai dengan penjelasan yang ada pada sub bab (2.2.2.). Berdasarkan algortima *Linear Quadratic Regulator*, maka tahapan proses sistem kendali adalah sebagai berikut:

1. Membaca *desire value* sinyal(referensi).
2. Membaca nilai *output* berupa nilai sensor yang disebut *state vector.*
3. Menghitung *gain K* dengan menggunakan metode *(Linear Quadratic Regulator)*
4. Membandingkan *gain feedback* dengan sinyal referensi.
5. Mengirim sinyal kontrol pada *plant.*
6. Melakukan peruangan dari tahap 1.

Tahapan kendali di atas dapat digambarkan pada diagram blok .



Gambar 5. 12. Diagram Blok Sistem Kontrol *LQR*

Untuk mendapatkan output yang di inginkan, sistem akan melakukan perulangan secara *realtime* dengan menghitung sinyal *feedback* sistem dengan metode *LQR*, *state vector* dapat ditentukan dari pembacaan sensor yang tertanam pada sistem *Ar Drone,* kemudian hasil pembacaan sensor tersebut dapat di akses dengan *ROS* melalui 2 *node* yaitu *node /Ar Drone/navdata, dan node /Ar Drone/imu*, *node* ini berfungsi untuk menerima data sensor berupa data *float* yang sudah di konversi oleh sistem *ROS*.

Hasil dari sinyal *feedback* selanjutnya akan dibandingkan dengan *desire value* kemudian sinyal erornya akan diteruskan pada sistem input, di mana sistem input pada *ROS* menggunakan *node /Ar Drone/cmd_vel*, *node* ini bekerja dengan mengirimkan perintah berbentuk *cuman* yang terdiris dari 6 atribut yaitu *Linear X, Linear Y, Linear Z, Angular X, Angular Y, Angular Z*, nilai maksimum yang di perbolehkan oleh *driver Ar Drone* adalah nilai antara -1 sampai 1.Sistem ini akan melakukan perulangan secara *realtime* dengan kecepatan 10Hz, sampai nilai yang diinginkan sesuai dengan nilai aktualnya.

Untuk Penentuan matriks Q dan R biasanya dilakukan dengan menggunakan metode *genetik algorithm*, tetapi pada penelitian ini tidak menggunakan metode tersebut, melainkan menggunakan *trial and error* dengan syarat matriks Q adalah matriks simetris semi definit positif. Sedangkan matriks R adalah matriks simetris definit positif. Penentuan matriks pembobotan Q dan R dilakukan dengan menentukan nilai maksimum yang diperbolehkan pada setiap *state* Berdasarkan perhitungan menggunakan persamaan (2.26).

Menentukan nilai yang diperbolehkan pada *state* $\begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} \end{bmatrix}^T$ merupakan hal yang penting sebagai acuan untuk menentukan pembobotan Q dan R.Untuk melakukan pengujian penentuan matriks Q dan R berdasarkan *trial and error. Trial and error* pembobotan matriks Q dan matriks R dilakukan dengan menguji 5 nilai yang berbeda.

5.2.2. Implementasi Sistem Kontrol *LQR* pada *AR Drone*

5.2.2.1. Implementasi *Take off*

Implementasi *take off* diawalai dengan melakukan *input* 1 yang dilakukan oleh *user* pada PC agar *quadcopter* dapat menjalankan sistem yang telah dibuat. Seperti *source code* yang ada pada kode program 5.1 merupakan implementasi *take off* yang akan diimplementasi pada *quadcopter.*

Kode Program 5.1. Melakukan *Take off*

Untuk melakukan *take off*

```
1  pilih=input("enter 1:")
2  if pilih == 1:
3       lqr.SendTakeOff()
```

Pada kode program 5.1 diatas merupakan kode program untuk melakukan *take off.* Baris 1 merupakan *command* untuk *inisialisai* pilihan "enter 1" agar pengguna mengikuti perintah. Baris ke 2 merupakan kondisi yang harus

dipilih oleh pengguna dengan menginput 1, maka *quadcopter* akan melakukan *take off.*

5.2.2.2. Implementasi *Input* Jarak

Input yang digunakan dalam sistem ini adalah jarak yaitu 1m, 1.5m, 2m, 2.5m, dan 3m seperti yang terlihat pada kode program 5.2 berikut ini. Sistem akan meminta user melakukan masukan jarak untuk menjalankan program.

Kode Program 5.2 Implementasi *Input* jarak

Untuk melakukan *input* jarak dan perhitungan jarak

```
1  s_input=input("enter distance:")
2          start_time = time.clock()
3          while not s_input is None :
4                  if int(posisi) <= s_input*1000 :
5                          self.SetCommand(1,0,0,0,0,0)
6                          end_time = time.clock()
7                          timer = end_time - start_time
8                          """rata = total/input2"""
9                          posisi2 =
10 uav.X*time.clock()*time.clock()
11                         posisi=posisi+abs(posisi2)
12                         print("kecepatan x = " +
13 str(uav.X_dot))
14                         print("time = " +
15 str(time.clock()))
16                         print("posisi = " + str(posisi))
17
18                 elif int(posisi) >= s_input*1000:
19                         self.SendLand()
20
```

Pada table kode program 5.2 ini adalah kode program yang digunakan untuk *input* jarak yang digunakan dalam sistem. Pada baris 1 merupakan inisialisai untuk *inputan* jarak yaitu dengan kalimat "enter distance ". Baris ke 2 merupakan inisialisai waktu untuk mengetahui waktu yang digunakan untuk mencapai jarak. Baris ke 4 sampai dengan baris ke 16 merupakan suatu kondisi yang digunakan untuk sistem. Kondisi yang pertama adalah ketika *user* melakukan *input* jarak, maka jarak akan dikalikan dengan 1000, karena pada *quadcopter* satuan yang digunakan adalah dalam mm, sehingga untuk mengubah jarak ke dalam satuan meter jarak akan dikalikan dengan 1000.

Pada baris 6 ditentukan tingkat kecepatan pada *quadcopter* adalah kecepatan maksimal yaitu 1m/s. untuk mencari jarak tempuh *quadcopter* dilakukan dengan perhitungan kecepatan pada *quadcopter* dikalikan dengan waku. Selanjutnya kondisi kedua adalah ketika *quadcopter* telah mencapai jarak yang telah di *input* oleh user, secara otomatis *quadcopter* akan *landing.*

5.2.2.3. Implementasi Perhitungan *LQR.*

Implementasi pehitungan *LQR* yang ada pada perancangan adalah inisialisai matriks yang ada dan adanya penentuan matrikas dan juga pehitungan

lqr. Untuk implementasi inisialisai matrik seperti pada kode program 5.3 berikut ini.

Kode Program 5.3 Implementasi Inisialisai Matriks

Untuk inisialisai matriks

```
        class Plant():
    def __init__(self):

        self.l = 0.1785
#mjarak antara rotor dan pusat massa
        self.Jr = 2.20321*10**-5            #kgm^2
        self.b = 1.27*10**-7
#g/rpm^2 thrust coefficient
        self.d = 3.19*10**-11
#g/rpm^2 drag  coefficient
        self.g = -9.80665
#m/s**2 gravity
        self.m = 0.38                       # or
0.428  kg

        #variabel momen inersia dari quadrotor
        self.Ixx = 2.2383*10**-3            #kgm^2
        self.Iyy = 2.9858*10**-3            #kgm^2
        self.Izz = 4.8334*10**-3            #kgm^2

        #deklarasi nilai konstanta a1..a5 dan
b1..b3
        self.a1 = 0.8254478845552427
        #(self.Iyy - self.Izz) / self.Ixx
        self.a2 = -0.00984322923647411
        #(-self.Jr / self.Ixx)
        self.a3 = 0.15465303926842378
        #(self.Izz - self.Ixx) / self.Iyy
        self.a4 = 0.004558302644101462
        #(self.Jr/self.Y_inertia)
        self.a5 = -0.8691472972067786
        #(self.Ixx -self. Iyy) / self.Izz
        self.b1 = 79.74802305321
        #1 /self.Ixx
        self.b2 = 36.93052509620557
        #1 /self.Iyy
        self.b3 = 59.78297273762476
        #1 /self.Izz

        #variabel untuk matriks A
        self.a24 = 0
        self.a42 = 0
        self.a64 = 0
        self.b1 = 0
        self.b2 = 0
        self.b3 = 0
        self.A = np.matrix([])
        self.B = np.matrix([])
        self.C = np.matrix([])
        self.D = np.matrix([])
```

```
Untuk inisialisai matriks
60          self.Q = np.matrix([])
61          self.R = np.matrix([])
```

Pada kode program 5.3 diatas merupakan program untuk inisialisai matriks. Pada baris 1 sampai baris 2 merupakan insialisai kelas matriks. Baris ke 3 sampai dengan baris ke 12 merupakan inisialisai variabel pada *quadcopter.*variabel *l* adalah panjang lengan pada *quadcopter* dengan satuan adalah meter. *Jr* adalah momen inesrsia pada motor. Variablel *b* adalah *thrust factor* yang merupakan gaya angkat pada *quadcopter,* dan untuk variabel *d* merupakan *drag coefficient* yaitu gaya hambay yang dihasilkan oleh baling-baling terhadap udara pada *quadcopter.* Nilai dari setiap variable diperoleh dari nilai yang telah ditetapkan pada setiap *quadcopter AR Drone Parrot 2.0*. Baris 12 sampai baris 22 merupakan deklarasi nilai *momen inersia* yang merupakan ukuran untuk kelembaman terhadap perubahan posisi atau perubahan gerak. Nilai yang ada pada *momen inersia* merupakan nilai yang telah ditetapkan sesuai dengan ketentuan *quadcopter Ar Drone.* Untuk baris 27 sampai dengan baris 44 merupakan kode yang digunakan untuk memeperoleh nilai yang ada pada variable matriks *A,* dan *B.* nilai yang tersebut dihasilkan dari penggunaan persamaan (2.3) . Baris ke 47 sampai dengan baris ke 61 merupakan deklarasi variable yang digunakan untuk matriks *A, B, C, D.*

Dan untuk implementasi penentuan matriks dan perhitungan *lqr* seperti pada kode program 5.4 berikut ini.

Kode program 5.4 Implementasi Perhitungan Matriks

```
Untuk perhitungan matriks
1       def VarMatrix(self):
2           self.a24 = uav.yaw*self.a1 + self.a2*uav.rotorR
3           self.a42 = uav.yaw*self.a3 + self.a4*uav.rotorR
4           self.a64 = uav.roll_dot*self.a5
5           self.b1 =
6   self.g+(np.cos(uav.roll)*np.cos(uav.pitch))/self.m
7
8       def SetMatrix(self):
9           self.A = np.matrix([[0, 1, 0, 0, 0, 0, 0, 0],
10                              [0, 0, 0, self.a24, 0, 0, 0, 0],
11                              [0, 0, 0, 1, 0, 0, 0, 0],
12                              [0, self.a42, 0, 0, 0, 0, 0, 0],
13                              [0, 0, 0, 0, 0, 1, 0, 0],
14                              [0, 0, 0, self.a64, 0, 0, 0, 0],
15                              [0, 0, 0, 0, 0, 0, 0, 1],
16                              [0, 0, 0, 0, 0, 0, 0, 0]])
17
18
19          self.B = np.matrix([[0, 0, 0, 0],
20                                      [0, self.b1,0, 0],
21                                      [0, 0, 0, 0],
22                                      [0, 0,self.b2, 0],
23                                      [0, 0, 0, 0],
24                                      [0, 0, 0,self.b3],
25                                      [0, 0, 0, 0],
26                                      [0, 0, 0, 1]])
27
```

Untuk perhitungan matriks

```
32
33
34
35
36          self.C = np.matrix([[1, 0, 0, 0, 0, 0, 0, 0],
37                              0, 1, 0, 0, 0, 0, 0, 0],
38                              [0, 0, 1, 0, 0, 0, 0, 0],
39                              [0, 0, 0, 1, 0, 0, 0, 0],
40                              [0, 0, 0, 0, 1, 0, 0, 0],
41                              [0, 0, 0, 0, 0, 1, 0, 0],
42                              [0, 0, 0, 0, 0, 0, 1, 0],
43                              [0, 0, 0, 0, 0, 0, 0, 1]])
44
45
46          self.D = np.matrix([[0,0,0,0],
47                                [0,0,0,0],
48                                [0,0,0,0],
49                                [0,0,0,0],
50                                [0,0,0,0],
51                                [0,0,0,0],
52                                [0,0,0,0],
53                                [0,0,0,0]])
54
55
56          self.Q = np.matrix([[0.0005, 0, 0, 0, 0, 0, 0, 0],
57                              [0, 0.05, 0, 0, 0, 0, 0, 0],
58                              [0, 0, 0.05, 0, 0, 0, 0, 0],
59                              [0, 0, 0, 0.05, 0, 0, 0, 0],
60                              [0, 0, 0, 0, 0.05, 0, 0, 0],
61                              [0, 0, 0, 0, 0, 0.05, 0, 0],
62                              [0, 0, 0, 0, 0, 0, 1, 0],
63                              [0, 0, 0, 0, 0, 0, 0, 1]])
64
65
66          self.R = np.matrix([[0.1, 0, 0, 0],
67                                [0, 0.1, 0, 0],
68                                [0, 0, 0.1, 0],
69                                [0, 0, 0, 0.1]])
70              #menghitung gain K, persamaan riccati X, dan
71  eigen value matriks
72          self.K, X, closedLoopEigVals =
73  controlpy.synthesis.controller_lqr(matrix.A, matrix.B,
74  matrix.Q, matrix.R)
75      def StateFeedBack(self):
76          self.feedBack = self.x-(matrix.A-
77  matrix.B*self.K)*self.x
78          print("----------------------")
79          print(self.feedBack)
80          print("----------------------")
81
82
83
```

Pada kode program 5.4 merupakan perhitungan matriks yang telah di deklarasi seperti pada kode program 5.3. Pada baris 1 merupakan inisialisai *class* matriks. Baris 2 sampai baris ke 5 merupakan rumus yang digunakan untuk memperoleh nilai a*24, a42, a64,* dan *b1*, nilai untuk variable ini adalah nilai dari perhitungan yang ada pada table kode program (5.3). Baris 7 sampai baris ke 70 merupakan matriks yang digunakan yaitu matriks *A, B, C, D, Q, R.*

Matriks *A, B* merupakan *input* matriks yang digunakan untuk sistem. Nilai dari setiap matriks merupakan nilai dari hasil dari perhitungan persamaan yang ada pada baris 2 sampai dengan baris 5. *Output* matriks yang ada pada program ini adalah matriks *C, D, Q, R.* Baris 71 sampai baris 76 merupakan perhitungan *gain K* dengan menggunakan persamaan *riccati X* dan nilai dari matriks. Baris 77 sampai dengan baris 83 merupakan pehitungan untuk *feedback* pada sistem dan untuk menampilkan hasil *feedback* pada perhitungan *LQR.*

Dan untuk perhitungan matriks yang ada pada Matlab seperti pada kode program 5.5. Kode program ini merupakan kode program yang digunakan untuk menghasilah matriks *Q* dan matriks *R* yang ada pada *LQR*. Untuk matriks *Q* dan *R* di hasilkan dengan *trial and error* seperti pada persamaan (2.26).

Kode program 5.5 Implementasi Perhitungan Matriks

Perhitungan matriks pada matlab

```
1  l =  0.1785;
2  Jr = 2.20321*10^-3;
3  b = 1.27*10^-7;
4  d = 3.19*10^-11;
5  g = 9.80665;
6  m = 0.38;
7  a1 = 0.8254478845552427;
8  a2 = -0.00984322923647411;
9  a3 = 0.15465303926842378;
10 a4 = 0.004558302644101462;
11 a5 = -0.8691472972067786;
12 b1 = 79.74802305321;
13 b2 = 36.93052509620557;
14 b3 = 59.78297273762476;
15 motorR = 2000;
16 pitch = 0.5;
17 pitch_dot = 0;
18 roll = 2.3;
19 roll_dot = 0;
20 yaw = 3.3;
21 yaw_dot = 0;
22 z = 0;
23 z_dot = 0;
24 x = 0;
25 x_dot = 0;
26 y = 0;
27 y_dot = 0;
28 ux = cos(roll)*sin(pitch)*cos(yaw) + sin(roll)*sin(yaw);
29 uy = cos(roll)*sin(pitch)*cos(yaw) - sin(roll)*sin(yaw);
30 a24 = roll*a1 + a2*motorR;
31 a42 = roll*a3 + a4*motorR;
32 a64 = roll_dot*a5;
33 b81 = cos(roll)*cos(pitch)/m;
34 b1 = ux/m;
35 b2 = uy/m;
36 A = [0 1 0 0 0 0 0 0 ;
37      0 0 0 a24 0 0 0 0 ;
38      0 0 0 1 0 0 0 0 ;
39      0 a42 0 0 0 0 0 0 ;
40      0 0 0 0 0 1 0 0 ;
41      0 0 0 a64 0 0 0 0 ;
42
43
44
45
```

Perhitungan matriks pada matlab

```
46        0 0 0 0 0 0 0 1];
47
48
49    B = [0 0 0 0;
50         0 b1 0 0;
51         0 0 0 0;
52         0 0 b2 0;
53         0 0 0 0;
54         0 0 0 b3;
55         0 0 0 0;
56         0 0 0 1];
57
58
59    C = [1 0 0 0 0 0 0 0 ;
60         0 1 0 0 0 0 0 0 ;
61         0 0 1 0 0 0 0 0 ;
62         0 0 0 1 0 0 0 0 ;
63         0 0 0 0 1 0 0 0 ;
64         0 0 0 0 0 1 0 0 ;
65         0 0 0 0 0 0 1 0 ;
66         0 0 0 0 0 0 0 1];
67
68
69      D = [0 0 0 0;
70           0 0 0 0;
71           0 0 0 0;
72           0 0 0 0;
73           0 0 0 0;
74           0 0 0 0;
75           0 0 0 0;
76           0 0 0 0];
77    a = 45 ;
78    b = 1 ;
79    c = 3;
80
81
82       Q = [1/a^2 0 0 0 0 0 0 0;
83            0 1/a^2 0 0 0 0 0 0;
84            0 0 1/a^2 0  0 0 0 0;
85            0 0 0 1/a^2 0 0 0 0;
86            0 0 0 0 1/a^2 0 0 0;
87            0 0 0 0 0 1/a^2 0 0;
88            0 0 0 0 0 0 1/b^2 0;
89            0 0 0 0 0 0 0 1/b^2];
90
91
92       R = [1/c^2 0 0 0;
93            0 1/c^2 0 0;
94            0 0 1/c^2 0;
95            0 0 0 1/c^2];
96
97
```

Kode program 5.5 diatas adalah kode program yang digunakan untuk perhitungan matriks yang ada pada matlab. Pada baris 1 sampai baris 35 merupakan deklarasi variabel yang akan digunakan untuk nilai pada setiap matriks, sama hal nya dengan kode program 5.3 yang melakukan deklarasi pada program yang ada pada *python,* serta adanya deklarasi untuk setiap motor pada *quadcopter, roll, pitch, yaw, y.*Nilai untuk setiap variable sudah ditentukan dari persamaan yang ada. Pada baris 1

sampai baris 7 untuk nilai setiap variable adalah ketetapan dari *quadcopter* yaitu untuk nilai massa, jarak antara rotor pada *quadcopter, grafitasi,* konstanta *drag motor*.

Baris 8 sampai dengan baris 16 adalah nilai untuk variable matriks yang digunakan sebagai *inputan.* Nilai tersebut dihasilkan melalui rumus yang ada pada persamaan (2.3). Baris ke 22 sampai dengan baris 35 merupakan nilai untuk setiap *roll, roll_dot, pitch, picth_dot, yaw, yaw_dot, x, x_dot, y, y_dot, z, z_dot. Range* baik untuk nilai *yaw* yang baik untuk pergerakan adalah antara 0.7 sampai dengan 6.1 rad/s. Pada program ini digunakan nilai 3.3 karena nilai ini nilai tengah yang ada. Ketika digunakan nilai maksimal maka akan terjadi perpindahan *quadcopter* yang lebih *aggressive* dan tidak stabil. *Range* nilai untuk *pitch* adalah 0.1 – 0.5 rad/s, pada penelitian ini digunakan 0.5 digunakan karena nilai tersebut memiliki respon pada sistem lebih cepat dan stabil. Pada baris 17 merupakan kecepatan motor yang digunakan untuk sistem, digunakan kecepatan 2000rpm karena kecepatan ini dapat menghasilakan gaya dorong untuk *quadcopter* dengan *massa* sebesar 0.38*kg* dengan kecepatan yang stabil untuk *quadcopter.*

Baris 36 sampai dengan baris 43 merupakan rumus persamaan yang digunaakan untuk memeperoleh nilai setiap variable yang akan menjadi nilai untuk matriks *A, B, C, D*. Baris 44 sampai baris 81 merupakan matriks *A, B, C, D.* Baris 82 dan baris 83 merupakan nilai sudut maksimal yaitu 45 dan nilai kecepatan 1m/s yang akan digunakan untuk menghitung nilai untuk matriks *Q.* Nilai sudut maksimal digunakan 45 karena nilai memperoleh hasil yang lebih maksimal dengan respon yang baik dibandingkan dengan sudut yang lain. *Range* yang ada pada kecepatan *quadcopter* adalah -1 sampai dengan 1 dan di program ini kecepatan yang digunakan adalah kecepatan maksimal yaitu 1m/s dengan tujuan untuk memeperoleh respon kecepatan yang lebih stabil. Baris 84 merupakan jumlah *n* yang digunakan untuk menghitung nilai matriks *R.* baris 87 sampai baris 97 merupakan matriks *Q* dan matriks *R*, yang nilai setiap matriks diperoleh dari perhitungan pada persamaan (2.26) dengan nilai tiap variable adalah pada baris 81.

VI. PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk menguji sistem yang sudah dibangun, apakah sistem yang sudah di implementasikan sesuai dengan tujuan dari penelitian ini, pengujian ini dilakukan untuk menjawab rumusan masalah yang telah dijabarkan sebelumnya. Selain menjawab rumusan masalah, pengujian ini juga akan menjawab kebutuhan non-fungsional yang telah dijelaskan pada gamabr 4.2 yang telah dijabarkan pada bab Analisis kebutuhan. Adapun hal yang terkait pada pengujian yang dilakukan adalah sebagai berikut.

6.1. Pengujian Matriks Q dan R pada Matlab

6.1.1. Tujuan Pengujian

Tujuan dari pengujian ini adalah untuk mendapatkan matriks *Q* dan matriks *R*. matriks ini akan digunakan untuk melakukan implementasi *Linear Quadratic Regulator (LQR)* pada *quadcopter.*

6.1.2. Pelaksanaan Pengujian

Pelaksanaan pengujian ini dilakukan pada perangkat lunak yang digunakan yaitu matlab. Pengujian ini dilakukan dengan melakukan *trial and error* pada matriks *Q* dan matriks *R.*

6.1.3. Prosedur Pengujian

Pengujian ini dilakukan seperti langkah-langkah berikut ini:

1. Pastikan bahwa aplikasi matlab telah dibuka pada computer
2. Jalankan kode program yang ada pada kode program 5.5.
3. Lihat hasil grafik pada matlab

6.1.4. Hasil Pengujian

Matriks Q dan matriks R ditentukan berdasarkan pengujian dari kelima nilai *trial and error* di atas, dan percobaan yang digunakan yaitu pada percobaan pertama:

$$Q = \begin{bmatrix} 0.0005 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0005 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0005 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0005 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0005 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0005 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Dan

$$R = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$$

Berdasarkan pengujian yang dilakukan menggunakan matriks pembobotan Q dan R di atas menghasilkan grafik kecepatan, dan grafik kecepatan pada Nilai tertinggi pada *state* $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^T$ yang digunakan pada penelitian ini adalah sebesar 45 derajat, dan nilai tertinggi pada *state* $[\ x \ \ \dot{x}\ ]^T$ adalah sebesar 1m/s.

6.1.5. Analisis Hasil Pengujian

1. Percobaan pertama dengan nilai matriks Q dan matriks R dengan acuan nilai maksimum yang diperbolehkan pada *state* $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^T$ adalah 45 derajat, sedangkan pada $[\ x \ \dot{x}\ ]^T$ adalah 1000 mm/s dihitung dengan menggunakan persamaan **Error! Reference source not found.** yang menghasilkan:

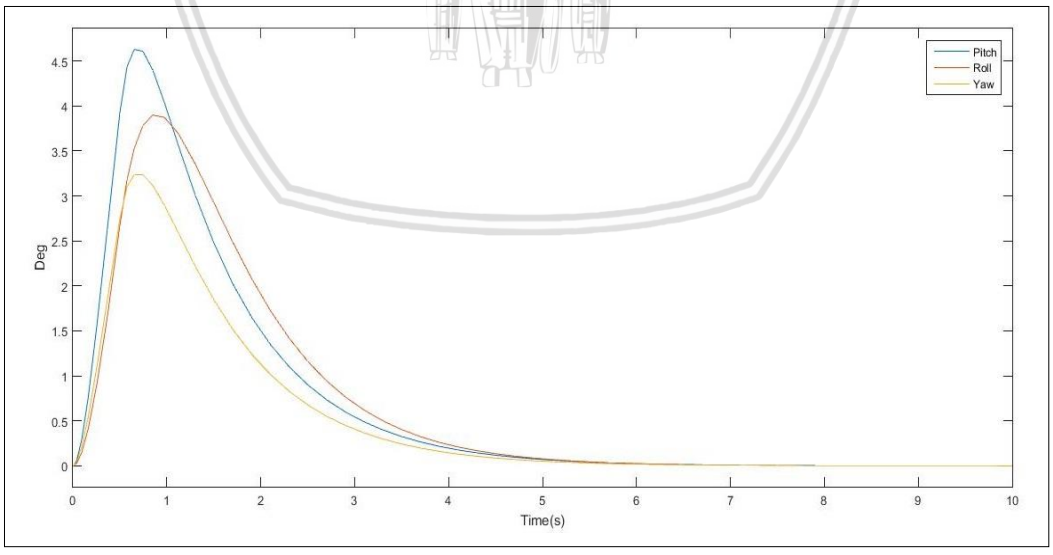
Matriks Q :

Q_{11}	Q_{22}	Q_{33}	Q_{44}	Q_{55}	Q_{66}	Q_{77}	Q_{88}
0.0005	0.0005	0.0005	0.0005	0.0005	0.0005	1	1

Matriks R dengan percobaan :

R_{11}	R_{22}	R_{33}	R_{44}
0.1	0.1	0.1	0.1

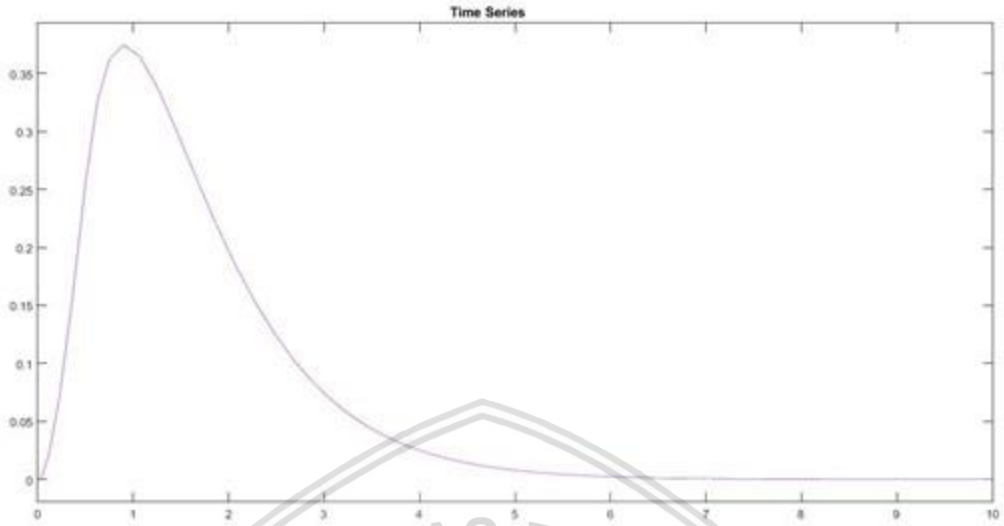
Akan menghasilkan respon sinyal seperti gambar 6.1



Gambar 6. 1. Grafik Percobaan Pertama

Pada Gambar 6.1 merupakan grafik *pitch, roll, yaw*, dimana pada grafik diatas menunjukkan bahwa nilai tertinggi pada sumbu y adalah *state* maksimal yang

dapat di capai oleh sudut *pitch, roll, yaw* sebesar 45 derajat, sedangkan respon kecepatan seperti Gambar 6.2.



Gambar 6. 2. Grafik Percobaan kecepatan

2. Percobaan kedua dilakukan dengan menaikkan nilai matriks Q pada *state* $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^T$ dari nilai sebelumnya pada percobaan pertama, sedangkan nilai *state* $[\ x \ \ \dot{x}\ ]^T$ tidak ada perubahan.

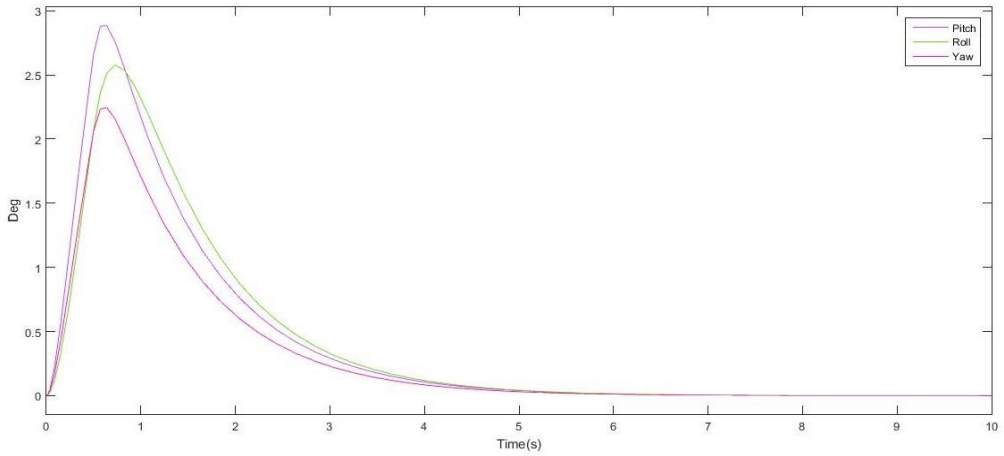
Matriks Q :

Q_{11}	Q_{22}	Q_{33}	Q_{44}	Q_{55}	Q_{66}	Q_{77}	Q_{88}
0.005	0.005	0.005	0.005	0.005	0.005	1	1

Matriks R :

R_{11}	R_{22}	R_{33}	R_{44}
0.1	0.1	0.1	0.1

Pada Matriks Q terdapat perubahan *state* $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^T$ sedangkan nilai matriks R tidak ada perubahan, dan menghasilkan :



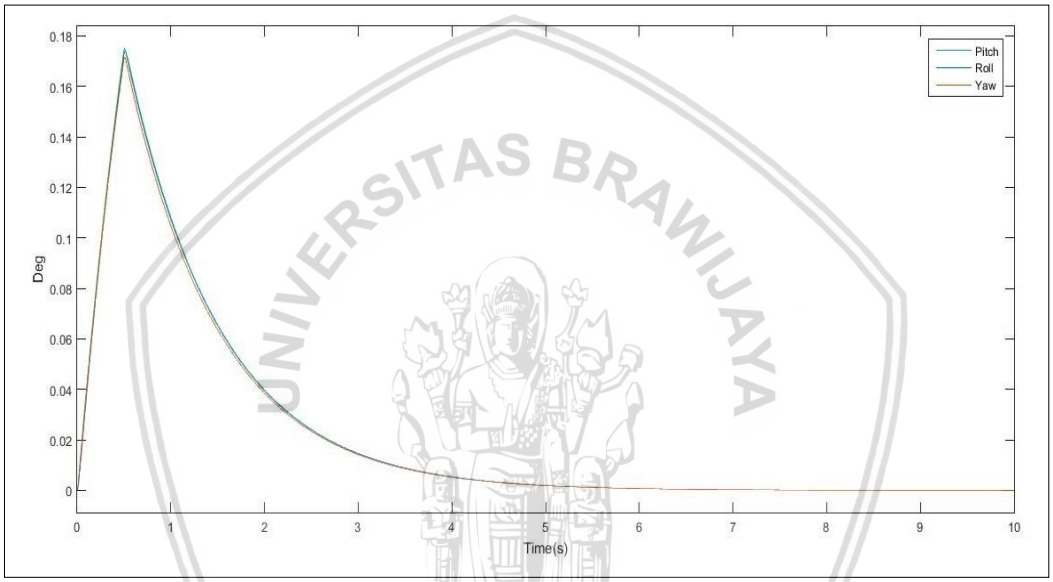
Gambar 6. 3. Grafik Percobaan Kedua

Pada gambar 6.3 merupakan grafik *pitch, roll, yaw*, di mana pada grafik di atas menunjukkan bahwa nilai tertinggi pada grafik sumbu y adalah *state* maksimal yang dapat di capai oleh, sudut *pitch, roll yaw* sebesar 30 derajat.

3. Percobaan ketiga dilakukan dengan menaikkan nilai matriks Q pada state $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^{\mathrm{T}}$ dari nilai sebelumnya pada percobaan kedua, sedangkan nilai state $[\ \mathrm{x} \ \ \dot{\mathrm{x}}\ ]^{\mathrm{T}}$ tidak ada perubahan.

Q_{11}	Q_{22}	Q_{33}	Q_{44}	Q_{55}	Q_{66}	Q_{77}	Q_{88}	R_{11}	R_{22}	R_{33}	R_{44}
0.05	0.05	0.05	0.05	0.05	0.05	1	1	0.1	0.1	0.1	0.1

Pada Matriks Q terdapat perubahan *state* $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^{T}$ sedangkan nilai matriks R tidak ada perubahan, dan menghasilkan :



Gambar 6. 4. Grafik percobaan ketiga

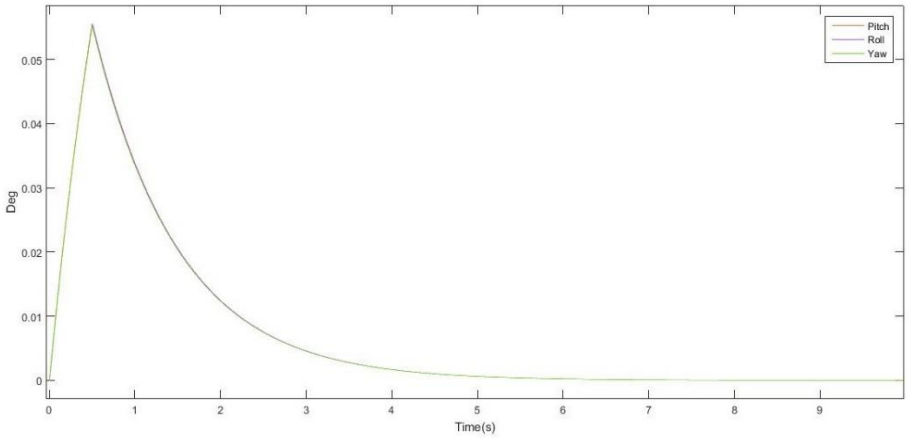
Pada gambar 6.4 merupakan grafik *pitch, roll, yaw*, dimana pada grafik diatas menunjukkan bahwa nilai tertinggi sumbu y yang adalah *state* maksimal yang dapat di capai oleh sudut *pitch, roll, yaw* sebesar 18 derajat.

4. Percobaan keempat dilakukan dengan menaikkan nilai matriks Q pada *state* $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^{T}$ dari nilai sebelumnya pada percobaan kedua, sedangkan nilai *state* $[\ x \ \ \dot{x}\ ]^{T}$ tidak ada perubahan.

Q_{11}	Q_{22}	Q_{33}	Q_{44}	Q_{55}	Q_{66}	Q_{77}	Q_{88}	R_{11}	R_{22}	R_{33}	R_{44}
5	5	5	5	5	5	1	1	0.1	0.1	0.1	0.1

Pada Matriks Q terdapat perubahan *state* $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^{T}$ sedangkan nilai matriks R tidak ada perubahan, dan menghasilkan :

Seperti yang terlihat pada gambar 6.5 berikut ini:



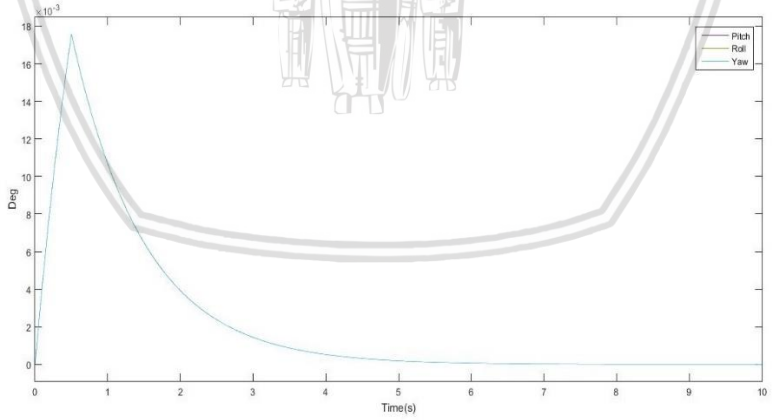
Gambar 6. 5. Grafik percobaan keempat

Pada Gambar 6.5 merupakan grafik *pitch, roll, yaw*. Pada grafik diatas menunjukkan bahwa nilai tertinggi sumbu y yang adalah *state* maksimal yang dapat di capai oleh sudut *pitch, roll, yaw* sebesar 6 derajat.

5. Percobaan kelima dilakukan dengan menaikkan nilai matriks Q pada *state* $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^T$ dari nilai sebelumnya pada percobaan kedua, sedangkan nilai *state* $[\ x \ \ \dot{x}\ ]^T$ tidak ada perubahan.

Q_{11}	Q_{22}	Q_{33}	Q_{44}	Q_{55}	Q_{66}	Q_{77}	Q_{88}	R_{11}	R_{22}	R_{33}	R_{44}
50	50	50	50	50	50	1	1	0.1	0.1	0.1	0.1

Pada Matriks Q terdapat perubahan *state* $[\phi \ \dot{\phi} \ \theta \ \dot{\theta} \ \psi \ \dot{\psi} \ ]^T$ sedangkan nilai matriks R tidak ada perubahan, dan menghasilkan :



Gambar 6. 6. Grafik Percobaan kelima

Pada gambar 6.6 merupakan grafik *pitch, roll, yaw*. Pada grafik diatas menunjukkan bahwa nilai sumbu y yang tertinggi adalah *state* maksimal yang dapat di capai oleh sudut *pitch, roll, yaw* sebesar 0.018 derajat.

6.2. Pengujian Kecepatan Tanpa Algorithama *LQR*

6.2.1. Tujuan Pengujian

Pada pengujian ini bertujuan untuk mengetahui kecepatan *quadcopter* tanpa menggunakan algorithma yang mempengaruhi waktu tempuh pada *quadcopter*, dengan parameter yang digunakan adalah jarak tempuh *quadcopter*.

6.2.2. Pelaksanaan Pengujian

Pelaksanaan pengujian dilakukan dengan menerbangkan *quadcopter* dengan kecepatan maksimal untuk mencapai jarak yang ditentukan oleh user. Adapaun jarak yang digunakan dalam pengujian ini adalah 1m, 1.5m, 2m, 2.5m, dan 3m.

6.2.3. Prosedur Pengujian

Pengujian ini dilakukan dengan prosedur sebagai berikut:

1. Siapkan *quadcopter*, pasang baterai yang sudah terisi, dan tunggu hingga *quadcopter* terhubung. *Quadcopter* dapat digunakan ketika terdengar suara beep sebanyak empat kali.
2. Menghubungkan *PC* dengan *hotspot Wi-Fi* yang ada pada *quadcopter.*
3. Membuka *terminal* pada *ubuntu* dan masuk ke direktori dengan menggetik "cd tum_simulator_ws", kemudian mengetik "source devel/setup.bash". setelah itu mengetik "roslaunch ardrone_autonomy ardrone.launch" untuk menghubungkan ROS dengan *quadcopter*.
4. Membuka terminal baru lagi pada Ubuntu dan masuk ke direktori dengan mengetikan "cd tum_simulator_ws", kemudian mengetik "source devel/setup.bash". Setelah itu mengetikan rqt plot untuk menampilkan grafik kecepatan pada *quadcopter*.
5. Membuka *terminal* lagi pada *ubuntu* dan masuk ke direktori dengan menggetik "cd tum_simulator_ws", kemudian mengetik "source devel/setup.bash". setelah itu mengetik "rosrun *lqr* velocitytanpa*lqr*.py" untuk menjalankan program.
6. Mengamati *output* pada *terminal* untuk mengetahui waktu tempuh *quadcopter* untuk mencapai jarak yang telah ditentukan berdasarkan kecepatan yang digunakan dan grafik pada rqt untuk mengetahui kestabilan *quadcopter*.
7. Untuk nilai jarak dapat dilihat dari pengukuran yang telah dilakukan yaitu dengan menggunakan meteran sesuai dengan nilai jarak yang digunakan.
8. Ulangi pengujian sebanyak 5 kali untuk setiap jarak yang telah ditentukan user.

6.2.4. Hasil Pengujian

Setelah dilakukan pengujian maka, dihasilkan output seperti pada tabel 6.1:

Tabel 6.1. Pengujian Kecepatan Tanpa Menggunakan *LQR*

NO	Jarak (Meter)	Kecepatan (mm/s)	Waktu Tempuh (s)	Tingkat Ketepatan Jarak	Tingkat Akurasi (%)

1	1	670	1.932	1867.2	0
2	1.5	726.6	1.864	2296.4	0
3	2	800	2.12	3053.4	0
4	2.5	856.4	2.56	3041.2	0
5	3	988.4	3.118	4171.8	0

Pada tabel 6.1 merupakan rata-rata dari hasil dari pengujian setiap jarak dengan kecepatan yang maksimum pada *quadcopter* yang dilakukan sebanyak 5 kali pengujian untuk setiap jarak. Untuk tingkat akurasi yang ada setelah dilakukan pengujian dihasilkan tingkat akurasi sebesar 0%.

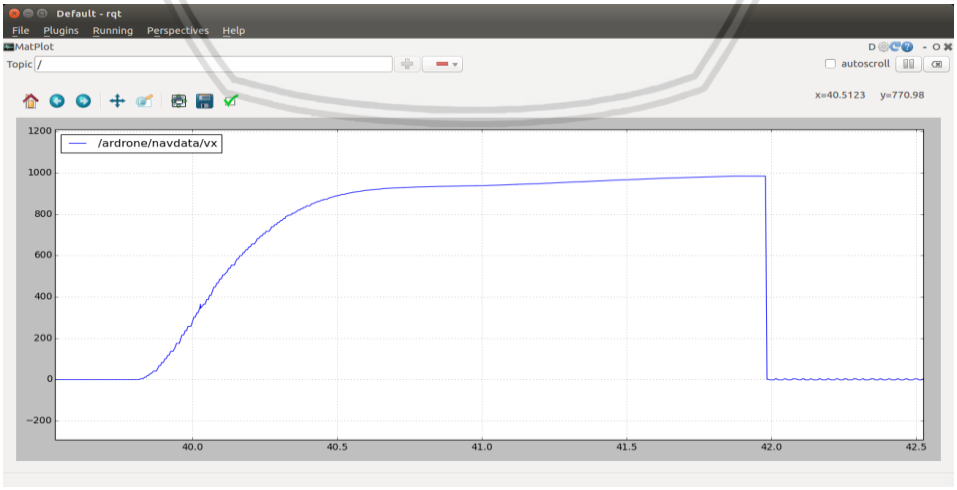
6.2.5. Analisis Hasil Pengujian

Setelah dilakukan pengujian sebanyak 5 kali pada jarak yang berbeda-beda didapatkan hasil tidak sesuai dengan yang diharapkan user. Untuk hasil persentase dari ketepatan kecepatan dan jarak yang dilakukan pada sistem *quadcopter* dapat dihitung dengan rumus pada persamaan 6.1

$$Persentase\ ketepatan = \frac{jumlah\ gerakan\ benar}{jumlah\ pengujian}\ x\ 100\% \quad (6.1)$$

a. Jarak 1 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 1 meter, dalam pengujian ini *quadcopter* mencapai jarak 1 meter dengan waktu yang dibutuhkan adalan 1.932 detik. Pengujian tanapa metode *LQR*, kecepatan yang pada pengujian ini terdapat ketidak stabilan. Dan ketika *quadcopter* telah mencapai jarak yang ditentukan, secara otomatis akan landing dan untuk tingkat ketepatan jarak yang ditempuh oleh *quadcopter* tidak sesuai dengan keinginan user yaitu selalu melebihi jarak yang diinputkan oleh user. Untuk grafik kecepatan tanpa *lqr* dengan jarak 1 m dapat di lihat pada gambar 6.7 berikut:



Gambar 6. 7. Grafik kecepatan dengan jarak 1m

Untuk hasil pengujian yang telah dilakukan sebanyak 5kali pada jarak 1m seperti pada tabel 6.1 berikut ini:

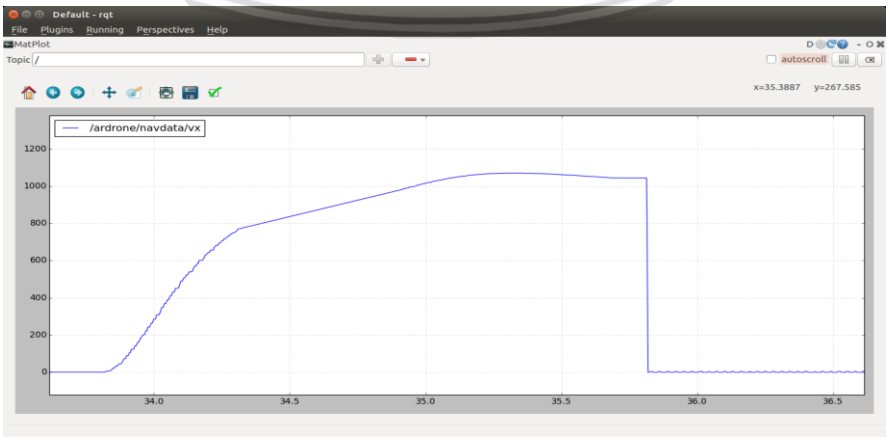
Tabel 6. 1. Pengujian kecepatan dengan jarak 1m tanpa *LQR*

Jarak (m)	Percobaan	Kecepatan (mm/s)	Waktu (s)	Tingkat Ketepatan jarak (mm)	Tingkat akurasi (%)
1	1	934	1.63	2013	0
	2	486	2.28	2858	
	3	660	1.80	2165	
	4	485	2.23	2703	
	5	785	1.72	2170	
Rata-Rata		670	1.932	1867.2	

Pada tabel 6.1 merupakan pengujian kecepatan dengan jarak 1m tanpa menggunakan *Linear Quadratic Regulator* yang dilakukan pengujian sebanyak lima kali. Pengujian yagn dilakukan memperoleh hasil yang tidak sesuai dengan keinginan user, yaitu adanya perbedaan kecepatan setiap pengujian yang dilakukan dan untuk tingkat ketepatan jarak yang tempuh oleh *quadcopter* melebihi batas yang telah digunakan yaitu 1m. Untuk rata-rata jarak yang dicapai oleh *quadcopter* pada pengujian ini adalah 1867 milimeter atau 1,867 meter dan kecepatan rata-rata untuk kecepatan adalah 670 mm/s.

b. Jarak 1.5 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 1.5 meter, dalam pengujian ini *quadcopter* mencapai jarak 1.5 meter dengan waktu rata-rata yang dibutuhkan adalah 1.864 detik. Pengujian tanapa metode *LQR*, kecepatan yang pada pengujian ini terdapat ketidak stabilan, dan untuk tingkat ketepatan jarak yang ditempuh tidak sesuai dengan yang diingikan oleh user, selalu melebihi jarak yang di inginkan oleh pengguna. Seperti yang terlihat pada Gambar 6.8 berikut ini:



Gambar 6. 8. Grafik kecepatan dengan jarak 1.5m

Untuk hasil pengujian yang telah dilakukan sebanyak 5 kali pada jarak yang digunakan adalah 1.5m dapat dilihat pada tabel 6.2 berikut ini:

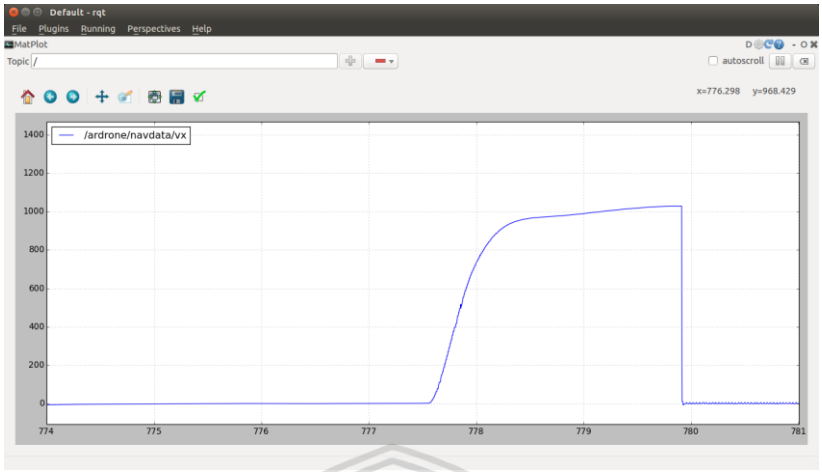
Tabel 6. 2. Pengujian Kecepatan dengan jarak 1.5 m tanpa *LQR*

Jarak (m)	Percobaan	Kecepatan (mm/s)	Waktu (s)	Tingkat Ketepatan jarak (mm)	Tingkat akurasi (%)
1.5	1	797	1.7	2278	0
	2	489	2.20	2618	
	3	481	2.15	2501	
	4	917	1.61	2054	
	5	949	1.66	2031	
Rata-Rata		726.6	1.864	2296.4	

Pada tabel 6.2 merupakan pengujian kecepatan dengan jarak 1.5m tanpa menggunakan *Linear Quadratic Regulator* yang dilakukan pengujian sebanyak lima kali. Pengujian yang dilakukan memperoleh hasil yang tidak sesuai dengan keinginan user, yaitu adanya perbedaan kecepatan setiap pengujian yang dilakukan dan untuk tingkat ketepatan jarak yang tempuh oleh *quadcopter* melebihi batas yang telah digunakan yaitu 1.5m. Untuk rata-rata jarak yang dicapai oleh *quadcopter* pada pengujian ini adalah 2296.4 milimeter atau 2,2964 meter dan kecepatan rata-rata untuk kecepatan adalah 726.6 mm/s

c. Jarak 2 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 2 meter, dalam pengujian ini *quadcopter* mencapai jarak 2 meter dengan waktu yang dibutuhkan adalan 2.12 detik. Pengujian tanapa metode *LQR*, kecepatan yang pada pengujian ini terdapat ketidak stabilan dan untuk tingkat ketepatan jarak yang ditempuh tidak sesuai dengan keinginan user. Ketika *quadcopter* telah mencapai jarak yang ditentukan, maka *quadcopter* akan landing secara otomatis. Untuk garafiknya dapat dilihat pada gambar 6.9 berikut ini.



Gambar 6. 9. Grafik kecepatan dengan jarak 2m

Untuk hasil pengujian yang telah dilakukan sebanyak 5 kali pada jarak yang digunakan adalah 2m dapat dilihat pada tabel 6.3 berikut ini:

Tabel 6. 3. Pengujian Kecepatan dengan jarak 2 m tanpa *LQR*

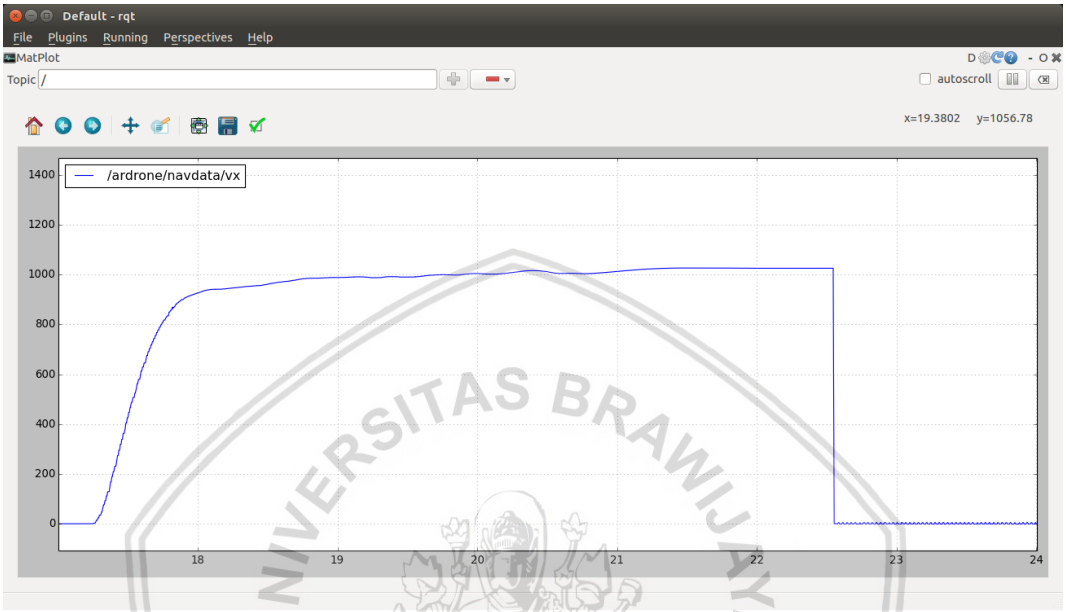
Jarak (m)	Percobaan	Kecepatan (mm/s)	Waktu (s)	Tingkat Ketepatan jarak (mm)	Tingkat akurasi (%)
2	1	952	2.03	3002	0
	2	978	2.12	3004	
	3	486	2.38	3075	
	4	668	2.14	3064	
	5	916	1.93	3122	
Rata-Rata		800	2.12	3053.4	

Pada tabel 6.3 merupakan pengujian kecepatan dengan jarak 2 m tanpa menggunakan *Linear Quadratic Regulator* yang dilakukan pengujian sebanyak lima kali. Pengujian yang dilakukan memperoleh hasil yang tidak sesuai dengan keinginan user, yaitu adanya perbedaan kecepatan setiap pengujian yang dilakukan dan untuk tingkat ketepatan jarak yang tempuh oleh *quadcopter* melebihi batas yang telah digunakan yaitu 2m. Untuk rata-rata jarak yang dicapai oleh *quadcopter* pada pengujian ini adalah 3053.4 milimeter atau 3.0534 meter dan kecepatan rata-rata untuk kecepatan adalah 800 mm/s

d. Jarak 2.5 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 2.5 meter, dalam pengujian ini *quadcopter* mencapai jarak 2.5 meter dengan waktu yang

dibutuhkan adalan 2.56 detik. Pengujian tanapa metode *LQR*, kecepatan yang pada pengujian ini terdapat ketidak stabilan dan untuk jarak yang ditempuh tidak sesuai dengan keinginan user, yaitu selalu melebihi jarak yang ditentukan. *Quadcopter* akan landing secara otomaris jika telah mencapai jarak yang telah ditentukan. Untuk garafiknya dapat dilihat pada gambar 6.10 berikut ini.



Gambar 6. 10. Grafik kecepatan dengan jarak 2.5m

Untuk hasil pengujian yang telah dilakukan sebanyak 5 kali pada jarak yang digunakan adalah 2m dapat dilihat pada tabel 6.4 berikut ini:

Tabel 6. 4. Pengujian Kecepatan dengan jarak 2.5 m tanpa *LQR*

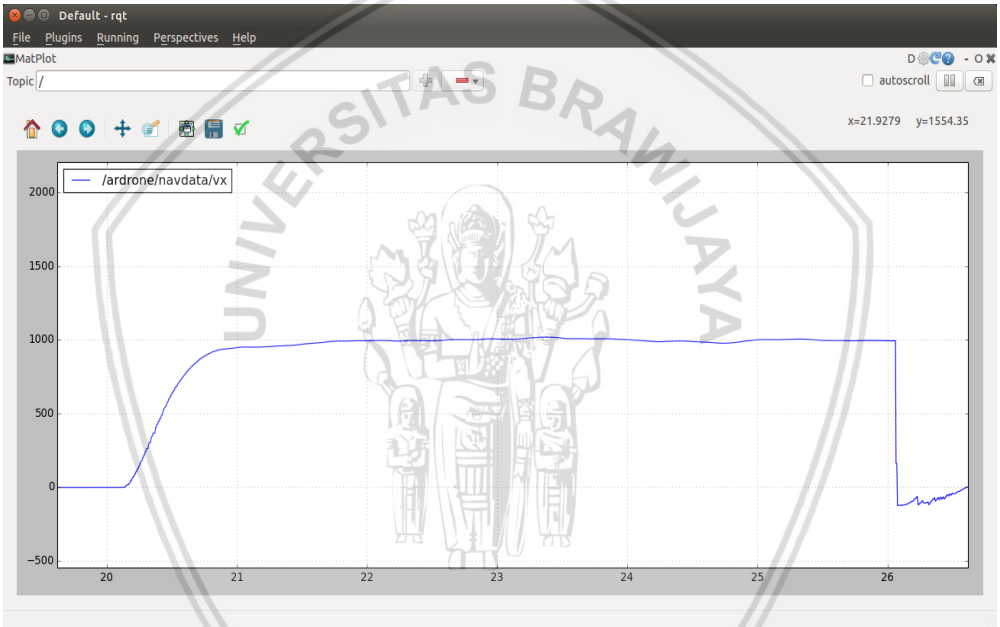
Jarak (m)	Percobaan	Kecepatan (mm/s)	Waktu (s)	Tingkat Ketepatan jarak (mm)	Tingkat akurasi (%)
2.5	1	987	2.90	3006	0
	2	655	2.15	3102	
	3	1005	2.73	3010	
	4	656	2.13	3066	
	5	979	2.92	3022	
Rata-Rata		856.4	2.56	3041.2	

Pada tabel 6.4 merupakan pengujian kecepatan dengan jarak 2.5 m tanpa menggunakan *Linear Quadratic Regulator* yang dilakukan pengujian sebanyak lima kali. Pengujian yang dilakukan memperoleh hasil yang tidak sesuai dengan keinginan user, yaitu adanya perbedaan kecepatan setiap pengujian yang

dilakukan dan untuk tingkat ketepatan jarak yang tempuh oleh *quadcopter* melebihi batas yang telah digunakan yaitu 2m. Untuk rata-rata jarak yang dicapai oleh *quadcopter* pada pengujian ini adalah 3041.2 milimeter atau 3.0412 meter dan kecepatan rata-rata untuk kecepatan adalah 856.4mm/s

e. Jarak 3 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 3 meter, dalam pengujian ini *quadcopter* mencapai jarak 3 meter dengan waktu yang dibutuhkan adalan 3.118 detik. Pengujian tanapa metode *LQR*, kecepatan yang pada pengujian ini terdapat ketidak stabilan. Dan ketika *quadcopter* telah mencapai jarak yang ditentukan, secara otomatis akan landing dan untuk tingkat ketepatan jarak yang ditempuh oleh *quadcopter* tidak sesuai dengan keinginan user yaitu selalu melebihi jarak yang diinputkan oleh user. Untuk garafiknya dapat dilihat pada gambar 6.11 berikut ini.



Gambar 6. 11 Grafik kecepatan dengan jarak 3m

Untuk hasil pengujian yang telah dilakukan sebanyak 5 kali pada jarak yang digunakan adalah 2m dapat dilihat pada tabel 6.5 berikut ini:

Tabel 6. 5. Pengujian Kecepatan dengan jarak 3 m tanpa *LQR*

Jarak (m)	Percobaan	Kecepatan (mm/s)	Waktu (s)	Tingkat Ketepatan jarak (mm)	Tingkat akurasi (%)
3	1	968	3.52	4041	0
	2	995	3.19	3095	
	3	992	3.15	4963	

	4	993	2.85	4520	
	5	994	2.88	4240	
Rata-Rata		988.4	3.118	4171.8	

Pada tabel 6.5 merupakan pengujian kecepatan dengan jarak 3 m tanpa menggunakan *Linear Quadratic Regulator* yang dilakukan pengujian sebanyak lima kali. Pengujian yang dilakukan memperoleh hasil yang tidak sesuai dengan keinginan user, yaitu adanya perbedaan kecepatan setiap pengujian yang dilakukan dan untuk tingkat ketepatan jarak yang tempuh oleh *quadcopter* melebihi batas yang telah digunakan yaitu 2m. Untuk rata-rata jarak yang dicapai oleh *quadcopter* pada pengujian ini adalah 4171.8 milimeter atau 4.1718 meter dan kecepatan rata-rata untuk kecepatan adalah 988.4 mm/s

6.3. Pengujian kecepatan dengan menggunakan *LQR*

6.3.1. Tujuan Pengujian

Pada pengujian ini bertujuan untuk mengetahui apakah penggunaan *lqr* dapat menstabilkan kecepatan *quadcopter* dengan jarak yang digunakan oleh user adalah 1m, 1.5m, 2m, 2.5m, dan 3m serta mengetahui waktu tempuh *quadcopter* untuk setiap jarak yang ada.

6.3.2. Pelaksanaan Pengujian

Pelaksanaan pengujian ini dilakukan dengan kecepatan maksimum untuk mencapai jarak yang ditentukan user yaitu 1m, 1.5m, 2m, 2.5m, dan 3m. Ketika jarak sudah mencapai jarak yang ditentukan maka *quadcopter* akan landing otomatis, dan untuk kecepatan *quadcopter* terlihat stabil yaitu sebesar 1000mm/s atau 1m/s.

6.3.3. Prosedur Pengujian

Pengujian ini dilakukan dengan prosedur sebagai berikut:

1. Siapkan *quadcopter*, pasang baterai yang sudah terisi, dan tunggu hingga *quadcopter* terhubung. *Quadcopter* dapat digunakan ketika terdengar suara beep sebanyak empat kali.
2. Menghubungkan *PC* dengan *hotspot Wi-Fi* yang ada pada *quadcopter.*
3. Membuka *terminal* pada *ubuntu* dan masuk ke direktori dengan menggetik “cd tum_simulator_ws”, kemudian mengetik “source devel/setup.bash”. setelah itu mengetik “roslaunch ardrone_autonomy ardrone.launch” untuk menghubungkan ROS dengan *quadcopter*.
4. Membuka terminal baru lagi pada Ubuntu dan masuk ke direktori dengan mengetikan “cd tum_simulator_ws”, kemudian mengetik “source devel/setup.bash”. Setelah itu mengetikan rqt plot untuk menampilkan grafik kecepatan pada *quadcopter*.

5. Membuka *terminal* lagi pada *ubuntu* dan masuk ke direktori dengan menggetik "cd tum_simulator_ws", kemudian mengetik "source devel/setup.bash". Setelah itu mengetik kode program yang telah ada yaitu dengan perintah "rosrun nama direktori nama program". Pada hal ini perintah yang saya menjalankan perintah berikut "rosrun *lqr* velocity*lqr*.py".
6. Mengamati *output* pada *terminal* untuk mengetahui waktu tempuh *quadcopter* untuk mencapai jarak yang telah ditentukan berdasarkan kecepatan yang digunakan dan grafik pada rqt untuk mengetahui kestabilan *quadcopter*.
7. Untuk nilai jarak dapat dilihat dari pengukuran yang telah dilakukan yaitu dengan menggunakan meteran sesuai dengan nilai jarak yang digunakan.
8. Ulangi pengujian sebanyak 5 kali untuk setiap jarak yang ditentukan user.

6.3.4. Hasil Pengujian

Setelah dilakukan pengujian maka, dihasilkan output seperti pada Tabel 6.6 berikut:

Tabel 6. 6. Pengujian Kecepatan Menggunakan *LQR*

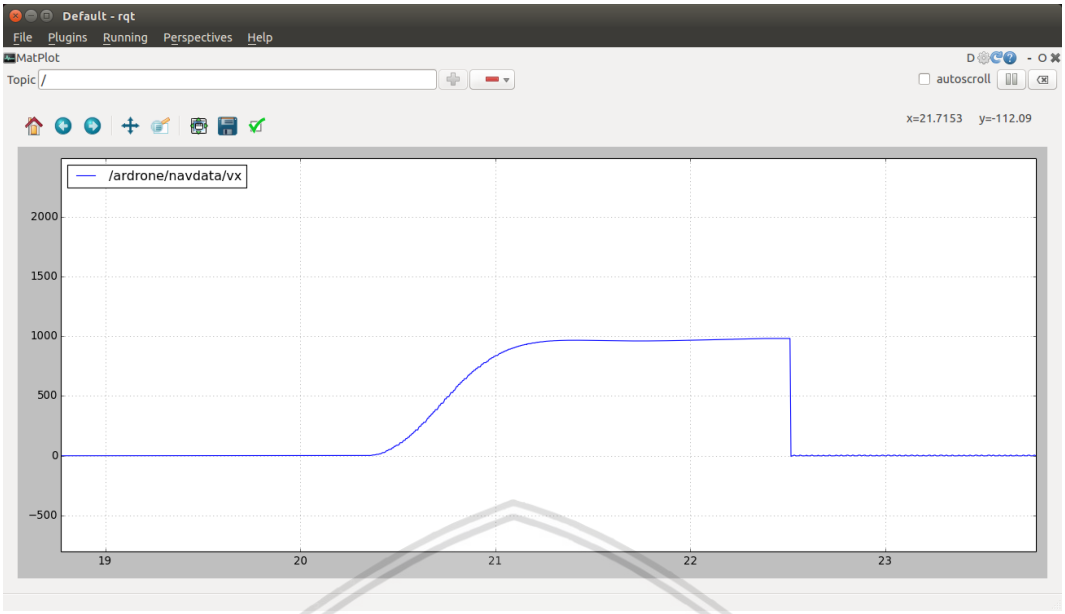
NO	Jarak	Kecepatan (mm/s)	Waktu	Tingkat Ketepatan Jarak (mm)	Tingkat akurasi (%)
1	1 Meter	1000	1.22	1000	100
2	1.5 Meter	1000	2.436	1500	100
3	2 Meter	1000	2.706	2000	100
4	2.5 Meter	1000	3.354	2500	100
5	3 Meter	1000	4.076	3000	100

6.3.5. Analisis Hasil Pengujian

Setelah dilakukan pengujian sebanyak 5 kali pada jarak yang berbeda-beda didapatkan hasil tidak sesuai dengan yang diharapkan user. Untuk hasil setiap pengujian yang telah dilakukan pada tiap jarak yang sudah ditentukan seperti dibawah ini:

a. Jarak 1 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 1 meter, dalam pengujian ini *quadcopter* mencapai jarak 1 meter dengan waktu rata-rata yang dibutuhkan adalan 1.22 detik. Pengujian menggunanakan metode *LQR*, kecepatan yang pada pengujian ini terlihat stabil yaitu selalu dalam kecepatan maksimal yaitu 1000mm/s, untuk hasilnya dapat dilihat dari gambar 6.12 dibawah ini:



Gambar 6. 12. Grafik kecepatan dengan jarak 1m

Untuk hasil pengujian yang telah dilakukan sebanyak 5 kali berturut-turut dengan jarak 1m dapat dilihat pada tabel 6.6 sebagai berikut:

Tabel 6. 7. Pengujian kecepatan dengan jarak 1m

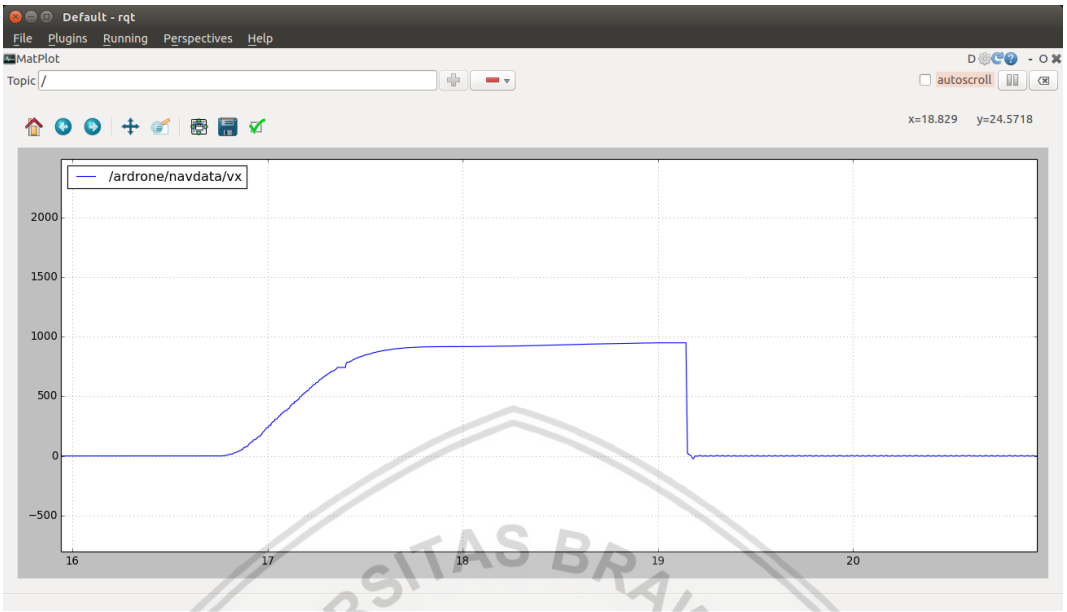
Jarak (meter)	Percobaan	Kecepatan(mm/s)	Waktu(s)	Ketepatan Jarak yang ditempuh(mm)	Tingkat akurasi
1	1	1000	1.1	1000	100%
	2	1000	1.3	1000	
	3	1000	1.4	1000	
	4	1000	1.2	1000	
	5	1000	1.1	1000	
Rata-Rata		1000	1.22	1000	

Pada tabel 6.7 diatas dapat kita lihat bahwa tingkat ketepatan yang dicapi oleh *quadcopter* adalah sesuai dengan yang ingin kan oleh user, yaitu 1m dan untuk tingkat ketepatan kecepatan yang ada juga selalu stabil yaitu 1000 mm/s. dan ketika *quadcopter* telah mencapai jarak yang ditentukan, maka secara otomatis *quadcopter* akan secara langsung melakukan landing otomatis.

b. Jarak 1.5 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 1.5 meter, dalam pengujian ini *quadcopter* mencapai jarak 1.5 meter dengan waktu rata-rata adalah 2,436 detik. Pengujian pada jarak ini menggunanakan metode *LQR* sama hal nya dengan jarak sebelumnya dengan kecepatan yang pada pengujian ini terlihat stabil

yaitu sebesar 1000mm/s dan untuk ketepatan jarak yang ditempuh yaitu sebesara 1500 milimeter atau 1,5 meter, seperti pada gambar 6.13 dibawah ini.



Gambar 6. 13. Grafik kecepatan dengan jarak 1m

Pada gambar 6.13 diatas terlihat bahwa kecepatan *quadcopter* tetap stabil, dan setelah *quadcopter* mencapai jarak yang telah ditentukan user makan secara otomatis *quadcopter* akan landing secara otomatsi. Untuk percobaan yang telah dilakukan oleh penulis seperti Tabel 6.8 dibawah ini:

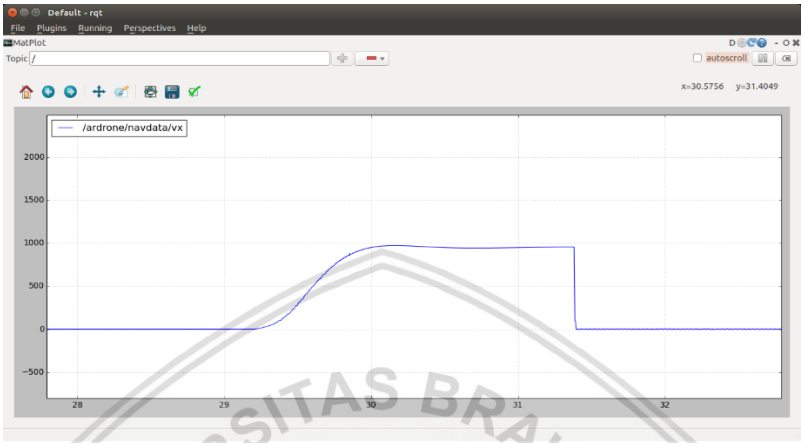
Tabel 6. 8. Pengujian Kecepatan dengan jarak 1,5 m menggunakan *LQR*

Jarak	Percobaan	Kecepatan	Waktu	Ketepatan Jarak yang ditempuh	Tingkat akurasi
1.5	1	1000	2.25	1500	100%
	2	1000	2.51	1500	
	3	1000	2.61	1500	
	4	1000	2.42	1500	
	5	1000	2.39	1500	
Rata-Rata		1000	2.436	1500	

Pada table 6.8 diatas dapat dilihat bahwa telah dilakukan pengujian sebanyak 5 kali dengan jarak yang sama dan dihasilkan bahwa kecepatan nya selalu stabil dan ketika sudah mencapai jarak yang ditentukan *quadcopter* akan landing secara otomatis untuk ketepatan jarak sesuai dengan keinginan user dan dengan tingkat akurasi ketepatan jarak yaitu 100%.

c. Jarak 2 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 2 meter, dalam pengujian ini *quadcopter* mencapai jarak 2 meter dengan waktu yang dibutuhkan adalan 2.075 detik. Pengujian menggunanakan metode *LQR*, kecepatan yang pada pengujian ini terlihat stabil, dilihat dari grafik yang ada, dan ketika *quadcopter* telah mencapai jarak yang ditentukan, maka *quadcopter* akan landing secara otomatis. Untuk grafiknya kecepatan adalah seperti gambar 6.14 berikut ini:



Gambar 6. 14. Grafik Kecepatan pada jarak 2m

Untuk hasil pengujian yang dilakukan secara 5 kali berturut-turut dapat dilihat pada tabel 6.9 berikut ini:

Tabel 6. 9. Pengujian kecepatan dengan jarak 2m

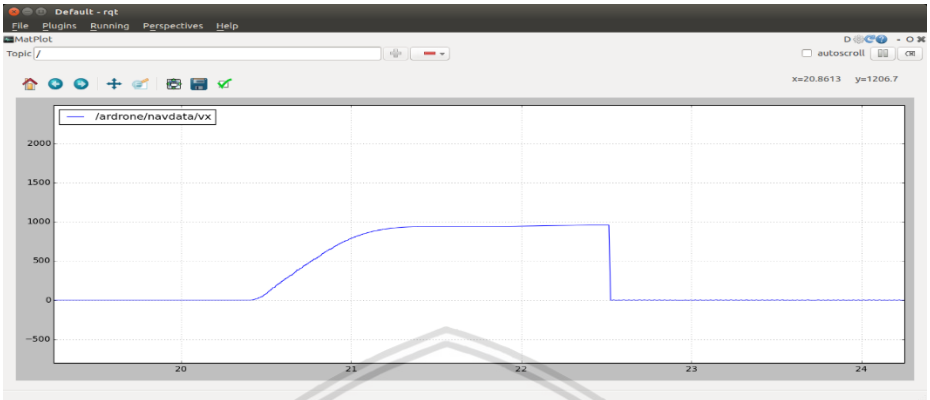
Jarak	Percobaan	Kecepatan	Waktu	Ketepatan Jarak yang ditempuh	Tingkat akurasi
2	1	1000	2.78	2000	100%
	2	1000	2.66	2000	
	3	1000	2.65	2000	
	4	1000	2.89	2000	
	5	1000	2.55	2000	
Rata-Rata		1000	2.706	2000	

Pada table 6.9 diatas dapat dilihat bahwa telah dilakukan pengujian sebanyak 5 kali dengan jarak yang sama dan dihasilkan bahwa kecepatan nya selalu stabil dan ketika sudah mencapai jarak yang ditentukan *quadcopter* akan landing secara otomatis untuk ketepatan jarak sesuai dengan keinginan user dan dengan tingkat akurasi ketepatan jarak yaitu 100%.

d. Jarak 2.5 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 2.5 meter, dalam pengujian ini *quadcopter* mencapai jarak 2.5 meter dengan waktu yang dibutuhkan adalan 3.354 detik. Pengujian menggunanakan metode *LQR*,

kecepatan yang pada pengujian ini terlihat stabil, dilihat dari grafik yang ada, dan ketika *quadcopter* telah mencapai jarak yang ditentukan, *quadcopter* akan melakukan landing secara otomatis. Untuk grafik kecepatan adalah seperti gambar 6.15 berikut ini:



Gambar 6. 15. Grafik Kecepatan dengan jarak 2.5 m

Untuk hasil pengujian yang telah dilakukan sebanyak 5 kali secara berturut-turut dapat dilihat pada tabel 6.10 berikut ini:

Tabel 6. 10. Hasil pengujian kecepatan dengan jarak 2.5 m

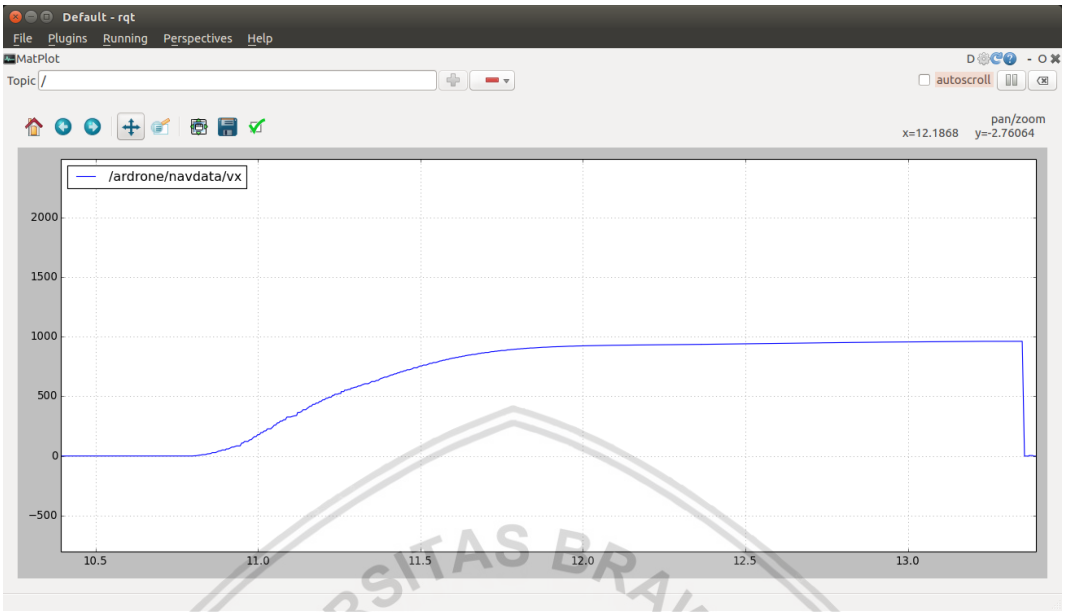
Jarak	Percobaan	Kecepatan	Waktu	Ketepatan Jarak yang ditempuh	Tingkat akurasi
2.5	1	1000	3.39	2500	100%
	2	1000	3.37	2500	
	3	1000	3.53	2500	
	4	1000	3.19	2500	
	5	1000	3.29	2500	
Rata-Rata		1000	3.354	2500	

Pada tabel 6.10 diatas dapat dilihat bahwa telah dilakukan pengujian sebanyak 5 kali dengan jarak yang sama dan dihasilkan bahwa kecepatan nya selalu stabil dan ketika sudah mencapai jarak yang ditentukan *quadcopter* akan landing secara otomatis untuk ketepatan jarak sesuai dengan keinginan user dan dengan tingkat akurasi ketepatan jarak yaitu 100%.

e. Jarak 3 Meter

Pada pengujian ini digunakan jarak yang digunakan user adalah 3 meter, dalam pengujian ini *quadcopter* mencapai jarak 3 meter dengan waktu yang dibutuhkan adalan 4.076 detik. Pengujian menggunanakan metode *LQR*, kecepatan yang pada pengujian ini terlihat stabil, dilihat dari grafik yang ada, dan ketika *quadcopter* telah mencapai jarak yang ditentukan, maka *quadcopter* akan melakukan landing

secara otomatis. Untuk grafik kestabila kecepatan adalah seperti gambar 6.16 berikut ini:



Gambar 6. 16. Grafik kecepatan dengan jarak 3m

Untuk hasil percobaan pada jarak 3m dengan pengujian sebanyak 5 kali dapat kita kita lihat pada tabel 6.11 berikut:

Tabel 6. 11. Hasil pengujian kecepatan dengan jarak 3m

Jarak	Percobaan	Kecepatan	Waktu	Ketepatan Jarak yang ditempuh	Tingkat akurasi
3	1	1000	4.41	3000	100%
	2	1000	4.06	3000	
	3	1000	3.605	3000	
	4	1000	4.361	3000	
	5	1000	3.945	3000	
Rata-Rata		1000	4.076	3000	

Pada table 6.11 diatas dapat dilihat bahwa telah dilakukan pengujian sebanyak 5 kali dengan jarak yang sama dan dihasilkan bahwa kecepatan nya selalu stabil dan ketika sudah mencapai jarak yang ditentukan *quadcopter* akan landing secara otomatis untuk ketepatan jarak sesuai dengan keinginan user dan dengan tingkat akurasi ketepatan jarak yaitu 100%.

VII. KESIMPULAN DAN SARAN

7.1. Kesimpulan

Berdasarkan analisa dari hasil yang telah dilakukan pada penelitian ini dapat diambil kesimpulan sesuai dengan rumusan masalah yang telah di tentukan sebelumnya, yaitu :

1. Pada penelitian ini dapat dilihat bahwa kinerja sistem terhadap perubahaan kecepatan yang dibangun menggunakan metode *LQR* dapat berjalan dengan baik terhadap sistem. Penggunaan *LQR* pada penelitian ini menghasilkan bahwa kecepatan pada *quadcopter* yang menggunakan *LQR* dapat terlihat stabil pada setiap jarak yang telah diinputkan, yaitu kecepatan dengan 1m/s atau 1000mm/s. Pengujian yang telah dilakukan untuk setiap jarak menghasilkan tingkat akurasi sebesar 100% untuk semua jarak yang ada. Berbeda dengan sistem yang dibuat tanpa menggunakan metode *Linear Quadratic Regulator*, yaitu setiap pengujian yang telah dilakukan, dihasilkan bahwa kecepatan yang selalu berubah-ubah dalam setiap jarak yang digunakan. Untuk waktu tempuh setiap jarak 1m, 1.5m, 2m, 2.5m, 3m membutuhkan waktu 1.2 detik ,2.43 detik, 2.07 detik, 3.35 detik dan 4.07 detik.

2. Pada pengujian ketepatan jarak yang ada, dihasilkan bahwa setiap jarak yang digunakan user memiliki tingkat ketepatan yang sesuai dengan yang digunakan user, yaitu 1m,1,5m,2m,2,5m,dan 3m untuk sistem yang mengggunakan metode *Linear Quadratic Regulator*. Untuk sistem yang tidak menggunakan metode *Liner Quadratic Regulator* memiliki tingkat ketepatan jarak yang jauh dari keinginan user, yaitu jarak yang ditempuh dua kali lipat dari jarak yang digunakan oleh user. Pada jarak 1m *quadcopter* menempuh jarak dengan rata-rata 1.867 m,untuk jarak 1.5 m *quadcopter* menempuh jarak 2.286 m, untuk jarak 2 m *quacopter* menempuh jarak 3.053 m, dan jarak 2.5 m *quadcopter* menempuh jarak sebesar 3.041 m serta untuk jarak yang seharusnya 3m *quadcopter* menempuh jarak sebesar 4.171m.

7.2. Saran

Agar sistem dapat dikembangkan lebih lanjut dan dengan performa yang lebih baik lagi, maka diperlukan saran:

1. Penelitian selanjutnya seperti sistem kendali posisi dan kendali pergerakan *quadcopter* dengan metode yang lain.

2. Pada penelitian selanjutnya tidak menggunakan *trial and error* untuk mmeperoleh matriks *Q* dan matriks *R*

DAFTAR PUSTAKA

Ahmadi. 2015. *Sistem kestabilan robot inverted pendulum menggunakan metode linear quadratic regulator*. Politeknik Negri Malang

Anggraini,Novi. 2005. "*Desain Kontroler Menggunakan Metode Linear Quadratic Regulator (LQR) untuk Pengontrolan Suhu Uap pada Solar Boiler Once Through Mode"*. *Skripsi*. Jurusan Tekinik Elektro, Fakultas Teknik, Universitas Brawijaya Malang.

Barták, R., Hraško, A. & Obdržálek, D., 2014. A Controller for Autonomous Landing of AR.Drone. IEEE

Hamdani, CN. Rusdhianto, EAK. Iskandar, E. 2013. *Perancangan Autonomous Landing pada Quadcopter Menggunakan Behavior-Based Intelligent Fuzzy Control.* Jurnal Teknik Pomits. 2(2):E63-E68

Kardono, Rusdhianto, EAK. Fatoni, A. 2012. *Perancangan dan Implementasi Sistem Pengaturan Optimal LQR untuk Menjaga Kestabilan Hover pada Quadcopter*. JURNAL TEKNIK ITS. 1 (1):F7-F13.

Latif, M, 2014. *Perancangan Sistem Autonomous Quadcopter.* Universitas Trunojoyo Madura

Ogata, K., 1997. *Teknik Kontrol Automatik jilid 2.* Jakarta: Erlangga.

Parlina, Soraya. 2016. *Kontrol kecepatan pada robot pendulum terbalik beroda dua menggunakan kontroler fuzzy*. ITS. Surabaya

Piskorski, S. & Brulez, N., 2016. *Drone Developer Guide. [Online] Available at:* https://www.msh-tools.com/Ar Drone/Ar Drone Developer Guide.pdf/ [Diakses pada 16 Mei 2018]

Romero, Luis Et al., 2014. *Quadcopter stabilization by using PID CONTROLLERS* Quito, Equador, IEEE.

Sumanti, Juliana., 2014. *Kontrol optimal pada balancing robot menggunakan metode Linear Quadratic Regulator.* UNSRAT

Tommaso, B. 2008. *Modelling, Identification and Control of a Quadcopter Helicopter*. Thesis. Department of Automatic Control Lund University.