

**PERANCANGAN SISTEM INSTRUMENTASI DAN KONTROL  
PENDARATAN OTONOM BERBASIS CITRA PADA  
WAHANA NIRAWAK QUADROTOR**

**LAPORAN TUGAS AKHIR**



Oleh  
Ihsan M. Fauzan NIM: 13320014

**PROGRAM STUDI TEKNIK FISIKA  
FAKULTAS TEKNOLOGI INDUSTRI  
INSTITUT TEKNOLOGI BANDUNG  
2024**

## ABSTRAK

### PERANCANGAN SISTEM INSTRUMENTASI DAN KONTROL PENDARATAN OTONOM BERBASIS CITRA PADA WAHANA NIRAWAK QUADROTOR

Oleh  
**Ihsan M. Fauzan NIM: 13320014**  
**(Program Studi Teknik Fisika)**

Quadrotor merupakan kendaraan yang dapat beroperasi secara otonom yang bisa digunakan untuk berbagai aplikasi. Salah satu tantangan dalam pengembangan quadrotor adalah algoritma pendaratan. Salah satu metode pendaratan yang umum digunakan adalah pendaratan berbasis citra, dengan menggunakan penanda marka, gambar, atau pola yang terletak pada lokasi pendaratan. Penelitian ini ditargetkan untuk menghasilkan rancangan sistem pendaratan quadrotor yang membuat quadrotor dapat melakukan proses pendaratan dengan menggunakan hasil tangkapan citra yang diperoleh dari kamera. Pada penelitian ini, digunakan marka ArUco yang cukup banyak digunakan sebagai penanda lokasi pendaratan pada penelitian sebelumnya. Proses pendaratan yang dilakukan diharapkan dapat dilakukan pada lingkungan dengan dukungan *Global Positioning System* (GPS) maupun tidak.

Penelitian dimulai dengan tahap perancangan sistem instrumentasi yang terdiri dari sistem estimasi marka ArUco serta sistem stabilisasi kamera yang menggunakan motor servo sebagai penggerak. Perancangan dilanjutkan dengan membuat sistem kontrol kaskade dengan 3 lapisan untuk mengontrol posisi. Hasil dari penelitian ini adalah sistem estimasi posisi menggunakan citra telah berhasil diujikan serta memberikan hasil yang baik secara korelasi serta stabil terhadap gangguan berupa orientasi quadrotor. Sistem pengontrolan juga bekerja dengan baik, dibuktikan dengan pendaratan yang berhasil dilakukan baik dengan bantuan GPS atau tidak. Quadrotor dapat mendarat dengan kriteria jarak dari pusat marka ArUco sebesar 30 cm dengan nilai MAE posisi (dalam cm) sebesar (38,6422; 31,4564; 54,5046) untuk pendaratan tanpa GPS dan (10,8672; 32,6929; 35,9229) untuk pendaratan dengan GPS. Untuk pendaratan tanpa GPS, improvisasi dilakukan dengan melakukan penalaan menggunakan model yang didapatkan dari data terbang. Aplikasi hasil penalaan pada pengujian pendaratan mendapatkan hasil pendaratan yang lebih stabil secara posisi, tapi membuat sinyal kontrol yang dihasilkan dan estimasi posisi quadrotor mengalami fluktuasi.

Kata kunci: quadrotor, instrumentasi, kontrol, pendaratan, stabilisasi, citra.

## ***ABSTRACT***

### ***DESIGN OF AN IMAGE-BASED AUTONOMOUS LANDING INSTRUMENTATION AND CONTROL SYSTEM FOR UNMANNED QUADROTOR VEHICLES***

*By*  
***Ihsan M. Fauzan NIM: 13320014***

***(Engineering Physics Study Program)***

*A quadrotor is a vehicle capable of operating autonomously and can be used for various applications. One of the challenges in quadrotor development is the landing algorithm. One commonly used landing method is image-based landing, utilizing markers, images, or patterns located at the landing site. This research aims to design a quadrotor landing system that enables the quadrotor to perform the landing process using image captures obtained from a camera. In this study, the ArUco marker, which is widely used as a landing location marker in previous research, is employed. The landing process is expected to be performed in environments with or without Global Positioning System (GPS) support.*

*The research began with the design phase of the instrumentation system, consisting of an ArUco marker estimation system and a camera stabilization system using servo motors as actuators. The design phase continued with the development of a cascade control system with three loops to control the position. The results of this research demonstrated that the position estimation system using images was successfully tested and provided good results in terms of correlation and stability against disturbances in the quadrotor's orientation. The control system also performed well, as evidenced by successful landings both with and without GPS assistance. The quadrotor was able to land within a distance of 30 cm from the center of the ArUco marker, with a mean absolute error (MAE) of position (in cm) of (38,6422; 31,4564; 54,5046) for landings without GPS and (10,8672; 32,6929; 35,9229) for landings with GPS. For landings without GPS, improvisation was done by tuning the model obtained from flight data. Applying the tuning results during landing tests resulted in more stable landings in terms of position, although it caused fluctuations in the generated control signals and the quadrotor's position estimation.*

*Keywords:* quadrotor, instrumentation, control, landing, stabilization, image.

**PERANCANGAN SISTEM INSTRUMENTASI DAN KONTROL  
PENDARATAN OTONOM BERBASIS CITRA PADA  
WAHANA NIRAWAK QUADROTOR**

**HALAMAN PENGESAHAN**

Oleh

**Ihsan M. Fauzan NIM: 13320014**

**(Program Studi Teknik Fisika)**

Institut Teknologi Bandung

Menyetujui

Tim Pembimbing

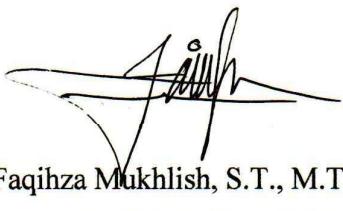
Tanggal 12 Juni 2024

Pembimbing 1

Pembimbing 2

  
Ir. Adhitya Gandaryus Saputro, Ph.D.

NOPEG. 117110025

  
Faqihza Mukhlish, S.T., M.T., Ph.D.

NOPEG. 121110001

## KATA PENGANTAR

Dengan rasa syukur, penulis ingin memanjatkan syukur kepada Allah Swt. karena telah memberikan rahmat dan karunia-Nya sehingga penulis berhasil menyelesaikan tugas akhir berjudul "Perancangan Sistem Instrumentasi dan Kontrol Pendaratan Otonom Berbasis Citra Pada Wahana Nirawak Quadrotor." Tugas akhir ini ditulis sebagai salah satu syarat kelulusan penulis dari Program Sarjana di Teknik Fisika ITB.

Penulis menyadari sepenuhnya bahwa penyusunan tugas akhir ini tidak mungkin terwujud tanpa dukungan, semangat, dan bimbingan dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

1. Faqihza Mukhlish, S.T., M.T., Ph.D. dan Ir. Adhitya Gandaryus Saputro, Ph.D. atas segala bimbingan dan dukungan serta banyak waktu yang diluangkan selama penggerjaan tugas akhir ini.
2. Fadjar Fathurrahman S.T., M.Si., Ph.D. sebagai dosen wali yang sudah memberikan dukungan selama keberjalanan perkuliahan.
3. Orang tua penulis, Dr. Muhammad Fauzan, M.A. dan Ermalinda, S.H.I. atas seluruh bimbingan dan dukungannya sehingga penulis dapat menempuh pendidikan di Institut Teknologi Bandung tanpa memiliki kendala sedikitpun.
4. Rekan seperjuangan quadrotor, Lutfi Rais Ismail dan Deedat Fatahillah yang telah berjuang bersama serta saling mendukung dari awal penggerjaan tugas akhir ini.
5. Teman-teman seluruh anggota Pusat Teknologi Instrumentasi dan Otomasi (PTIO) yang telah memberikan dukungan serta bersama-sama penulis selama mengerjakan tugas akhir.
6. Seluruh Dosen dan Tenaga Pendidik Program Studi Teknik Fisika yang telah memberikan ilmunya serta membantu dalam banyak aspek selama menempuh pendidikan di teknik fisika.
7. Teman-teman Teknik Fisika 2020 yang telah meneman dan berjuang bersama penulis sejak awal perkuliahan di teknik fisika, baik secara daring maupun luring.

8. Seluruh teman-teman dari GAMAIS, Unit Robotika ITB, Unit MTQ ITB serta berbagai unit kegiatan lain yang telah memberikan pengalaman berharga selama berkuliah di Institut Teknologi Bandung.
9. Seluruh pihak yang membantu yang tidak dapat disebutkan namanya satu persatu.

Penulis menyadari tugas akhir ini masih sangat jauh dari kata sempurna. Penulis sangat mengharapkan adanya saran dan kritik sehingga tugas akhir serta pelaksanaan tugas akhir nantinya dapat berjalan dengan baik. Penulis juga berharap hasil kerja keras yang dikerjakan ini nantinya dapat bermanfaat bagi bangsa, negara, serta seluruh masyarakat yang tinggal di dalamnya.

Bandung, 12 Juni 2024

Penulis

## DAFTAR ISI

<b>ABSTRAK .....</b>	<b>i</b>
<b>ABSTRACT .....</b>	<b>ii</b>
<b>HALAMAN PENGESAHAN .....</b>	<b>iii</b>
<b>KATA PENGANTAR.....</b>	<b>iv</b>
<b>DAFTAR ISI.....</b>	<b>vi</b>
<b>DAFTAR GAMBAR.....</b>	<b>viii</b>
<b>DAFTAR TABEL.....</b>	<b>x</b>
<b>DAFTAR SINGKATAN.....</b>	<b>xi</b>
<b>BAB I PENDAHULUAN.....</b>	<b>1</b>
1.1 Latar Belakang .....	1
1.2 Permasalahan.....	3
1.3 Tujuan dan Sasaran .....	3
1.4 Batasan dan Asumsi .....	4
1.5 Ide Solusi.....	4
1.6 Sistematika Penulisan.....	5
<b>BAB II TINJAUAN PUSTAKA.....</b>	<b>6</b>
2.1 Wahana Quadrotor .....	6
2.1.1 Model Kinematika Quadrotor .....	6
2.1.2 Model Dinamika Quadrotor .....	7
2.2 Pengukuran Berbasis Citra .....	9
2.2.1 <i>Perspective-N-Point Pose Estimation</i> .....	9
2.2.2 Distorsi pada Kamera .....	12
2.2.3 Marka ArUco.....	13
2.3 Pengontrol Umpang Balik dan Umpang Maju .....	15
2.4 Pengontrol <i>Proportional-Integral-Derivate</i> .....	15
2.5 Pengontrol Kaskade.....	17
2.6 <i>Raspberry Pi</i> .....	18
2.7 Pixhawk .....	20
2.8 Dynamixel XL-320 .....	21
2.9 <i>Inertial Stabilized Platform</i> .....	24
2.10 <i>Software</i> .....	26

2.10.1 OpenCV .....	26
2.10.2 Dronekit.....	26
<b>BAB III METODOLOGI DAN PERANCANGAN SISTEM .....</b>	<b>28</b>
3.1 Sistem Instrumentasi Pendaratan .....	28
3.1.1 Sistem Pengukuran Berbasis Citra .....	29
3.1.2 Sistem Stabilisasi Kamera .....	33
3.1.3 Integrasi Sistem Instrumentasi .....	35
3.2 Sistem Pengontrolan Pendaratan .....	37
3.2.1 Pengontrol Dengan Dukungan GPS .....	37
3.2.2 Pengontrolan Tanpa Dukungan GPS.....	38
3.2.3 Berbagai Pembatasan dan Pengaturan Parameter .....	41
3.3 Integrasi Sistem Secara Keseluruhan .....	42
<b>BAB IV HASIL DAN ANALISIS .....</b>	<b>44</b>
4.1 Performansi Sistem Instrumentasi.....	44
4.1.1 Sistem Pengukuran Koordinat Marka .....	44
4.1.2 Sistem Gimbal .....	48
4.2 Performansi Pengontrolan .....	49
4.2.1 Pengontrol Dengan Dukungan GPS .....	49
4.2.2 Pengontrol Tanpa Dukungan GPS .....	51
4.2.3 Improvisasi Pengontrol Tanpa Dukungan GPS.....	53
<b>BAB V KESIMPULAN DAN SARAN .....</b>	<b>56</b>
5.1 Kesimpulan.....	56
5.2 Saran.....	56
<b>DAFTAR PUSTAKA .....</b>	<b>58</b>
<b>Lampiran A Improvisasi Desain Gimbal.....</b>	<b>62</b>
A.1 Gimbal Versi Pertama .....	62
A.2 Gimbal Versi Kedua.....	62
A.3 Gimbal Versi Ketiga.....	63
<b>Lampiran B Diagram Pengontrol <i>Attitude Quadrotol</i>.....</b>	<b>64</b>

## DAFTAR GAMBAR

Gambar 1.1 Pendaratan Berbasis Visual [3] .....	1
Gambar 1.2 Marka ArUco dengan ukuran 5x5 (a) 6x6 (b) dan 8x8 (c).....	2
Gambar 1.3 Ide solusi .....	5
Gambar 2.1 Pengaturan Kecepatan Rotor untuk Mengatur Arah Gerak [13].....	6
Gambar 2.2 Sudut euler pada quadrotor [14].....	7
Gambar 2.3 Hubungan Antara <i>World Frame</i> dan <i>Camera Frame</i> .....	10
Gambar 2.4 Hubungan Geometri Antara Objek 3 Dimensi dan 2 Dimensi [16]...11	11
Gambar 2.5 Ilustrasi dari Distorsi Radial (a) dan Tangensial (b) [17].....	12
Gambar 2.6 Beberapa contoh <i>fiducial marker</i> [9] .....	13
Gambar 2.7 Urutan Proses Pendekripsi Marka ArUco [9] .....	14
Gambar 2.8 Diagram Blok Pengontrol Umpan Maju (atas) dan Umpan Balik (bawah) [19] .....	15
Gambar 2.9 Diagram Blok Pengontrol PID Paralel [21] .....	16
Gambar 2.10 Diagram Blok Pengontrol Kaskade [23].....	17
Gambar 2.11 Tampilan Fisik Raspberry Pi 4B [27] .....	18
Gambar 2.12 Diagram Pinout dari Raspberry Pi 4B [28] .....	19
Gambar 2.13 Tampilan Fisik dari Pixhawk 1 3DR [30] .....	20
Gambar 2.14 Pengontrolan Posisi Oleh Pixhawk [32].....	21
Gambar 2.15 Tampilan Fisik Servo Dynamixel XL-320.....	22
Gambar 2.16 Pengontrol PID pada Servo .....	22
Gambar 2.17 Urutan Bit Masuk dan Keluar dalam Dynamixel Protocol 2.0 .....	23
Gambar 2.18 Diagram Koneksi Servo XL-320.....	24
Gambar 2.19 Rangkaian Tambahan Komunikasi Dua Arah.....	24
Gambar 2.20 ISP 1 Sumbu [36] .....	25
Gambar 2.21 Diagram Blok Pengontrolan ISP [36] .....	26
Gambar 3.1 Arsitekur Sistem Instrumentasi .....	28
Gambar 3.2 Pi Camera OV5647 dengan IR.....	29
Gambar 3.3 Papan Catur Ukuran 7x10 .....	30
Gambar 3.4 Titik Potong yang Didetksi di Papan Catur.....	31
Gambar 3.5 Marka ArUco Bertumpuk.....	32
Gambar 3.6 Ilustrasi FOV Tanpa Gimbal .....	33
Gambar 3.7 Desain Mekanikal dari Gimbal .....	34
Gambar 3.8 Diagram Blok Pengontrolan Gimbal.....	34
Gambar 3.9 FTDI <i>USB to TTL Converter</i> .....	35
Gambar 3.10 Skematik Rangkaian Instrumentasi .....	36
Gambar 3.11 Board PCB Rangkaian yang Digunakan .....	36
Gambar 3.12 Diagram Blok Pengontrol Dengan Dukungan GPS .....	38
Gambar 3.13 Diagram Blok Pengontrol Tanpa Dukungan GPS.....	39
Gambar 3.14 Integrasi Seluruh Perangkat dan Rangkaian.....	42
Gambar 3.15 Diagram Alir Keseluruhan Program dengan 3 <i>Thread</i> Berbeda .....	43
Gambar 4.1 Tampilan Estimasi Jarak .....	44
Gambar 4.2 Tren Hasil Estimasi dan Pengukuran Aktual Koordinat x .....	45

Gambar 4.3 Tren Hasil Estimasi dan Pengukuran Aktual Koordinat y .....	46
Gambar 4.4 Tren Hasil Estimasi dan Pengukuran Aktual Koordinat z .....	46
Gambar 4.5 Pendektsian yang Gagal Karena Pantulan .....	47
Gambar 4.6 Marka ArUco Modifikasi .....	47
Gambar 4.7 Deteksi yang Berhasil di Siang Hari .....	47
Gambar 4.8 Kondisi Citra Pendaratan Sebelum (a) dan Sesudah Pergerakan (b) .	48
Gambar 4.9 Lintasan Pendaratan Dengan GPS Dalam 3 Dimensi (a) dan 2 Dimensi untuk Sumbu XY (b), Sumbu XZ (c) dan YZ (d).....	50
Gambar 4.10 Plot Radius Terhadap Waktu Pendaratan Dengan GPS .....	51
Gambar 4.11 Lintasan Pendaratan Tanpa GPS Dalam 3 Dimensi (a) dan 2 Dimensi untuk Sumbu XY (b), Sumbu XZ (c) dan YZ (d).....	52
Gambar 4.12 Plot Radius Terhadap Waktu Pendaratan Tanpa GPS .....	53
Gambar 4.13 Diagram Blok Simulasi Menggunakan Simulink.....	54
Gambar 4.14 Respons Sistem Pada Simulasi.....	54
Gambar 4.15 Grafik Setpoint Roll dan Pitch Terhadap Waktu .....	55
Gambar 4.16 Grafik Pengukuran X dan Y Terhadap Waktu .....	55

## DAFTAR TABEL

Tabel 2.1 Kelebihan dan Kekurangan Raspberry Pi .....	19
Tabel 3.1 Spesifikasi Kamera yang Digunakan .....	29
Tabel 3.2 Parameter Intrinsik Kamera .....	31
Tabel 4.1 Perbandingan Jarak Aktual dan Hasil Estimasi Dalam cm .....	45
Tabel 4.2 Parameter Pengontrol Dengan Kecepatan.....	49
Tabel 4.3 MAE Posisi Quadrotor Pendaratan Dengan GPS .....	50
Tabel 4.4 Parameter Pengontrol Dengan Sudut .....	51
Tabel 4.5 MAE Posisi Quadrotor Pendaratan Tanpa GPS.....	52
Tabel 4.6 Parameter PID Hasil Penalaan Visual.....	54

## DAFTAR SINGKATAN

SINGKATAN	Nama	Pemakaian pertama kali pada halaman
CRC	<i>Cyclic Redundancy Check</i>	23
EKF	<i>Extended Kalman Filter</i>	22
ESC	<i>Electronic Speed Controller</i>	21
FOV	<i>Field of View</i>	25
GPS	<i>Global Positioning System</i>	2
id	<i>identifier</i>	14
IOT	<i>Internet of Things</i>	18
ISP	<i>Inertially Stabilized Platform</i>	25
LOS	<i>Line of Sight</i>	25
MAE	<i>Mean Absolute Error</i>	49
PID	<i>Proportional-Integral-Derivative</i>	15
PnP	<i>Perspective-n-Point</i>	9
LAMBANG	Arti	Pemakaian pertama kali pada halaman
$\phi, \theta, \psi$	Sudut orientasi quadrotor	6
$X_E, Y_E, Z_B$	Koordinat terhadap bumi	7
$X_B, Y_B, Z_B$	Koordinat terhadap quadrotor	7
$R$	Matriks transformasi kordinat	7
$C_A$	Koefisien angkat	8
$C_x, C_y, C_z$	Koefisien gesek di berbagai sumbu	8
$A$	Luas permukaan baling-baling	8
$m$	Massa quadrotor	8
$\rho$	Massa jenis udara	8
$r$	Jari-jari baling baling	8
$\Omega$	Kecepatan sudut baling-baling	8
$r$	Jari-jari baling baling	8
$I_{cm}$	Momen inersia terhadap pusat massa	8
$\omega_b$	Kecepatan sudut	8
$l$	Panjang lengan quadrotor	9
$\alpha, \beta$	Sudut lengan quadrotor	9
${}^c\mathbf{T}_w$	Matriks Transformasi koordinat <i>world</i> ke <i>camera</i>	10
${}^c\mathbf{R}_w$	Matriks Rotasi koordinat <i>world</i> ke <i>camera</i>	10
${}^c\mathbf{t}_w$	Matriks Translasi <i>koordinat</i> <i>world</i> ke <i>camera</i>	10
$p_x, p_y$	<i>Focal Length</i> Kamera	10
$u_0, v_0$	<i>Principal Point</i>	10
$x_d, y_d$	Koordinat terdistorsi	12
$x_u, y_u$	Koordinat ideal	12

$k_1, k_2 \dots$	Koefisien distorsi radial	12
$p_1, p_2, \dots$	Koefisien distorsi tangensial	21
$k_p, k_i, k_d$	Koefisien pengontrol propotional-integral-derivative	17
$e$	<i>Error</i> (perbedaan hasil pengukuran dengan setpoint)	17
$x_{sp}, y_{sp}, z_{sp}$	Setpoint posisi	38
$\dot{x}_{sp}, \dot{y}_{sp}, \dot{z}_{sp}$	Setpoint kecepatan	38
$\phi_{sp}, \theta_{sp}, T_{sp}$	Setpoint sudut <i>roll</i> , sudut <i>pitch</i> , dan <i>thrust</i>	38
$\bar{x}, \bar{y}, \bar{z}$	Estimasi posisi	38
$\bar{\dot{x}}, \bar{\dot{y}}, \bar{\dot{z}}$	Estimasi Kecepatan	38

## BAB I

### PENDAHULUAN

#### 1.1 Latar Belakang

Quadrotor merupakan salah satu kendaraan tanpa awak yang dapat digunakan untuk berbagai aplikasi. Oleh karena itu, penelitian tentang quadrotor merupakan topik yang sangat menarik, mengingat fungsionalitasnya yang sangat tinggi. Salah satu bagian yang menjadi tantangan tersendiri untuk pengembangan quadrotor adalah instrumentasi dan algoritma pendaratan otomatis yang merupakan bahasan utama pada tugas akhir ini.

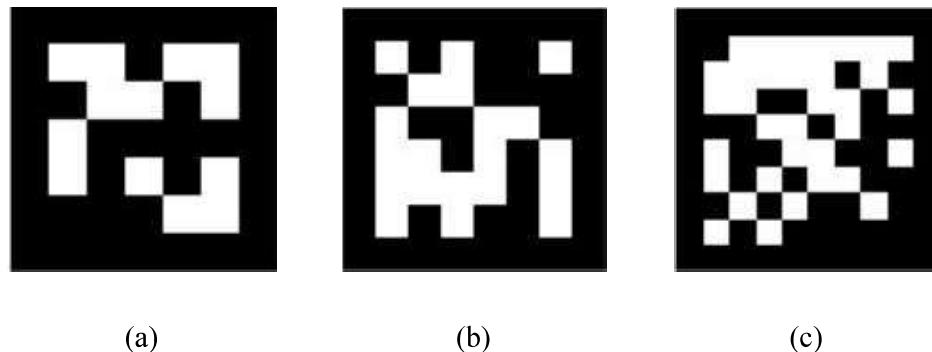
Berbagai penelitian telah dilakukan untuk melakukan peningkatan performansi pada tahapan pendaratan quadrotor. Penelitian yang telah dilakukan antara lain adalah untuk mengompensasi gangguan angin [1], pendaratan di atas platform yang bergerak [2], [3] atau pendaratan pada bidang miring [4]. Dengan adanya penelitian tersebut, diharapkan lebih banyak opsi pendaratan quadrotor yang bisa dilakukan sehingga quadrotor dapat beradaptasi untuk melakukan pendaratan di dalam situasi apapun.



**Gambar 1.1** Pendaratan Berbasis Visual [3]

Salah satu metode pendaratan yang cukup banyak digunakan adalah pendaratan berbasis citra. Pendaratan menggunakan metode ini menggunakan gambar yang berupa pola tertentu yang diletakkan pada lokasi pendaratan. Kamera yang

terpasang pada quadrotor kemudian digunakan untuk melakukan estimasi posisi pada proses pendaratan yang kemudian digunakan sebagai referensi bagi pengontrol untuk melakukan pendaratan. Penggunaan citra sebagai estimator posisi dapat menggantikan *global positioning system* (GPS) yang performansinya sangat bergantung pada satelit dan dapat cukup terganggu apabila berada dalam lingkungan perkotaan yang kompleks atau berada di dalam ruangan [5].



**Gambar 1.2** Marka ArUco dengan ukuran 5x5 (a) 6x6 (b) dan 8x8 (c)

Dari sekian jenis gambar atau pola yang digunakan dapat digunakan sebagai penanda lokasi pendaratan, marka fidusial ArUco merupakan marka yang cukup banyak digunakan dalam perancangan sistem pendaratan berbasis visual [6], [7], [8]. Penggunaan marka ini dalam proses pendekripsi gambar memiliki kelebihan dari sisi algoritma yang digunakan. Marka ArUco dideteksi menggunakan algoritma *edge detection* [9], sehingga tidak lagi diperlukan proses *training* data sebagaimana yang diperlukan jika menggunakan gambar yang lain sebagai penanda lokasi pendaratan. Marka ArUco juga memiliki performa yang cukup baik untuk mengestimasi posisi sebagaimana yang telah dilakukan pada penelitian sebelumnya [10].

Pada sistem fotografi/pencitraan, stabilisasi pada kamera untuk kendaraan nirawak udara juga merupakan topik yang cukup banyak untuk diteliti. Mateo Gašparović dan Luka Jurjević [11] melakukan penelitian tentang pengaruh gimbal 2 sumbu dalam parameter eksterior citra yang diambil dari citra yang didapatkan oleh kamera yang dipasang pada quadrotor. Penelitian lainnya tentang topik ini dilakukan oleh Chin E. Lin dan Shen-Kai Yang [12] yang bertujuan untuk membuat

sistem stabilisasi kamera yang dapat membuat kamera dapat mengikuti dan fokus kepada target. Kedua penelitian ini menghasilkan hasil yang lebih baik dibandingkan tanpa sistem stabilisasi. Performa seperti ini pun diharapkan dapat dicapai dalam penelitian ini.

Pada tugas akhir ini, akan dirancang sebuah sistem instrumentasi dan kontrol untuk pendaratan otonom pada quadrotor. Sistem ini dapat mendeteksi marka ArUco yang terletak pada lokasi pendaratan serta dapat mengestimasi posisi relatif dari lokasi pendaratan terhadap quadrotor menggunakan kamera dengan sistem stabilisasi yang menjaga orientasi kamera agar tetap mengarah ke arah bawah. Posisi ini kemudian digunakan oleh algoritma pengontrolan untuk mendaratkan quadrotor pada lokasi yang ditandai menggunakan marka. Pada sistem yang dirancang ini, quadrotor diharapkan dapat mendarat dalam kondisi dukungan GPS yang tersedia maupun tidak, sehingga dapat lebih fleksibel untuk mendarat di berbagai kondisi lingkungan.

## 1.2 Permasalahan

Berdasarkan hal yang telah dibahas pada latar belakang, permasalahan yang ingin diselesaikan pada tugas akhir ini adalah :

1. Bagaimana merancang sistem instrumentasi pendaratan quadrotor berbasis citra?
2. Bagaimana merancang sistem stabilisasi kamera untuk mempertahankan orientasi kamera dari perubahan orientasi quadrotor?
3. Bagaimana merancang sistem pendaratan yang dapat beroperasi baik di lingkungan dengan GPS maupun tanpa GPS?

## 1.3 Tujuan dan Sasaran

Tujuan yang ingin dicapai pada penelitian ini adalah perancangan sistem instrumen pendaratan pada kendaraan nirawak quadrotor beserta algoritma pendaratannya. Sasaran yang ingin dicapai melalui pemenuhan tujuan tersebut adalah :

1. Kendaraan nirawak quadrotor dapat mendeteksi lokasi pendaratan dengan bantuan citra dari kamera yang dipasang.

2. Kendaraan nirawak quadrotor dapat melakukan pendaratan otonom pada lokasi yang ditentukan oleh marka ArUco.

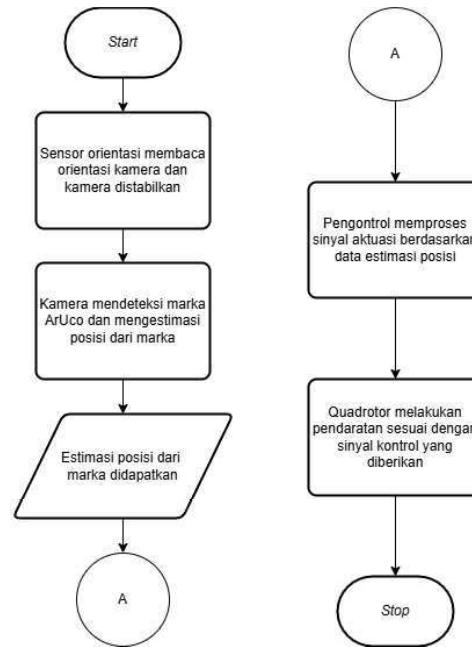
#### 1.4 Batasan dan Asumsi

Batasan dan asumsi pada pelaksanaan tugas akhir adalah sebagai berikut :

1. Landasan bergerak dan tidak berada pada bidang miring.
2. Mode terbang atau *cruise* dilakukan oleh pilot.
3. Sistem estimasi posisi tidak dirancang dengan akurasi tinggi, namun cukup memiliki korelasi yang baik.
4. Pengontrol pendaratan dirancang menggunakan pengontrol bawaan pada flight controller sebagai pengontrol bagian dalam, sehingga yang dirancang hanyalah pengontrol bagian luar.
5. Proses pendaratan ditargetkan untuk mendarat dengan toleransi terhadap pusat marka, tidak tepat harus berada di tepat di pusat marka.

#### 1.5 Ide Solusi

Untuk mencapai tujuan dari tugas akhir ini, dirancanglah ide solusi sesuai dengan yang dijelaskan pada diagram alir pada Gambar 1.3. Kendaraan nirawak quadrotor dapat mendeteksi lokasi pendaratan dengan marka ArUco dengan kamera yang telah distabilisasi. Data citra kemudian diproses dan digunakan untuk mengestimasi nilai posisi dari lokasi pendaratan. Pengontrol kemudian memberikan aksi kontrol kepada *flight controller* sesuai dengan kalkulasi *error*. Quadrotor kemudian mengikuti aksi kontrol yang diberikan serta melakukan proses pendaratan hingga selesai.



**Gambar 1.3** Ide solusi

## 1.6 Sistematika Penulisan

Proposal penelitian tugas akhir ini ditulis dengan sistematika sebagai berikut:

### BAB I: PENDAHULUAN

Bab ini berisikan latar belakang, rumusan masalah, sasaran dan tujuan, batasan dan asumsi, ide solusi dan sistematika penulisan.

### BAB II: STUDI LITERATUR

Bab tersebut berisi tentang penjelasan singkat mengenai literatur yang digunakan dalam penelitian ini.

### BAB III: METODOLOGI DAN PERANCANGAN SISTEM

Bab ini berisi tentang bagaimana sistem dirancang dan diintegrasikan sehingga dapat diujikan.

### BAB IV: HASIL DAN ANALISIS

Bab ini berisi tentang pengujian dari setiap sistem yang dirancang serta analisis dari performansinya.

### BAB V: KESIMPULAN DAN SARAN

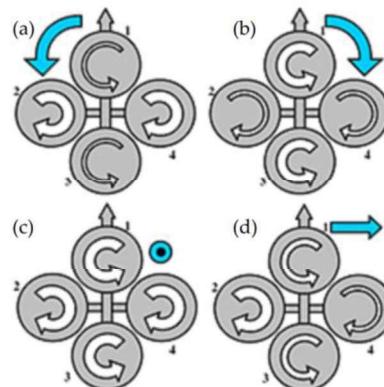
Bab ini berisikan kesimpulan dari penelitian dan saran yang berguna untuk penelitian ke depannya.

## BAB II

### TINJAUAN PUSTAKA

#### 2.1 Wahana Quadrotor

Quadrotor adalah wahana terbang yang memiliki 4 rotor yang mengontrol pergerakannya. 4 rotor tersebut terdiri dari 2 pasang rotor yang memiliki konfigurasi yang saling mengompensasi gerakan 1 sama lain. Untuk menggerakkan quadrotor, kecepatan keempat rotor diatur sedemikian rupa seperti yang diilustrasikan pada Gambar 2.1. Pada gambar tersebut, ketebalan garis panah bewarna abu-abu memiliki arti kecepatan dari rotor. Semakin tebal garis panah, maka semakin cepat pula kecepatan dari rotor.

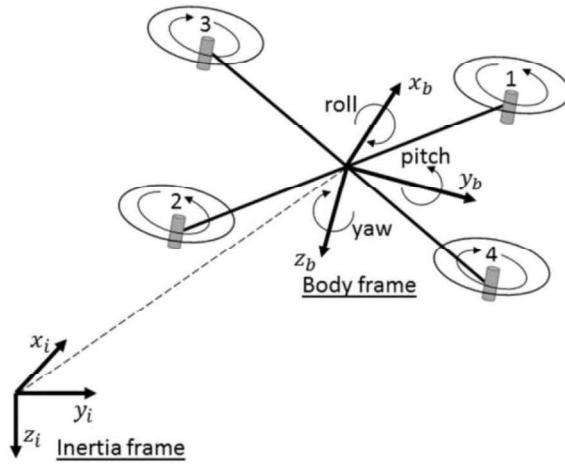


**Gambar 2.1** Pengaturan Kecepatan Rotor untuk Mengatur Arah Gerak [13]

Persamaan quadrotor dapat dibagi menjadi 2 bagian, yaitu kinematika dan dinamika. Persamaan kinematika dan quadrotor akan dijelaskan pada subbab berikut.

##### 2.1.1 Model Kinematika Quadrotor

Untuk membahas persamaan kinematika pada quadrotor, diperlukan adanya tinjauan untuk menyatakan hubungan antara koordinat dengan referensi bumi serta referensi dari quadrotor. Untuk menghubungkan kedua hal tersebut, diperlukan adanya tinjauan terkait 3 sudut euler yang dimiliki quadrotor yaitu sudut *roll* ( $\phi$ ), *pitch* ( $\theta$ ), dan *yaw* ( $\psi$ ). Sudut tersebut dapat dilihat pada Gambar 2.2.



**Gambar 2.2** Sudut euler pada quadrotor [14]

Hubungan kinematika quadrotor dapat dilihat pada persamaan (2.1).

$$\begin{bmatrix} X_E \\ Y_E \\ Z_E \end{bmatrix} = R \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} \quad (2.1)$$

Dengan R adalah sudut transformasi koordinat yang nilainya dapat dilihat pada persamaan :

$$R = \begin{bmatrix} c_\theta c_\psi & -c_\phi s_\psi - s_\phi s_\theta c_\psi & -c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & -s_\phi s_\theta s_\psi + c_\phi s_\psi & -c_\phi s_\theta c_\psi - s_\phi c_\psi \\ s_\theta & s_\phi c_\theta & c_\phi s_\theta \end{bmatrix} \quad (2.2)$$

Pada persamaan di atas dan setelahnya, notasi trigonometri cos dan sin disingkat menjadi c dan s untuk mempersingkat penulisan.

Hubungan antara koordinat berlaku tidak hanya pada posisi, akan tetapi pada turunannya juga, yaitu kecepatan dan percepatan. Hubungan antara kedua koordinat tersebut penting karena gaya dorong dari quadrotor hanya ada pada sumbu z quadrotor, sehingga penting untuk ditransformasi sehingga didapat ekuivalensinya untuk koordinat dengan referensi bumi.

### 2.1.2 Model Dinamika Quadrotor

Quadrotor menggunakan gaya dorong yang dihasilkan oleh baling-baling memanfaatkan gaya gesek yang ada pada udara. Gaya angkat tersebut proporsional

dengan kecepatan baling-baling quadrotor yang dinyatakan dengan persamaan berikut.

$$F_a = C_a \rho A r^2 \Omega^2 \quad (2.3)$$

Gaya angkat tersebut ada pada keempat baling-baling tergantung dengan kecepatan dari baling-baling tersebut. Selain gaya angkat, quadrotor juga dipengaruhi oleh gaya gravitasi serta gaya gesek dari udara. Gaya tersebut dirumuskan pada persamaan berikut.

$$F_g = mg \quad (2.4)$$

$$F_{sx} = \frac{1}{2} C_x A_c \rho \dot{x} |\dot{x}| \quad (2.5)$$

$$F_{sy} = \frac{1}{2} C_y A_c \rho \dot{y} |\dot{y}| \quad (2.6)$$

$$F_{sz} = \frac{1}{2} C_z A_c \rho \dot{z} |\dot{z}| \quad (2.7)$$

Untuk menyusun persamaan tersebut dan menghubungkan dengan percepatan dari quadrotor, digunakan persamaan Newton-Euler. Persamaan Newton-Euler memiliki bentuk sebagai berikut.

$$\begin{bmatrix} mI & 0 \\ 0 & I_{cm} \end{bmatrix} \begin{bmatrix} \dot{V} \\ \omega^b \end{bmatrix} + \begin{bmatrix} \omega^b \times mV \\ \omega^b \times I_{cm} \omega^b \end{bmatrix} = \begin{bmatrix} f \\ \tau^b \end{bmatrix} \quad (2.8)$$

Dengan bentuk akhir persamaan sebagai berikut setelah memasukkan sudut Euler yang sudah disebut pada bagian sebelumnya, memasukkan matriks transformasi koordinat serta mengasumsikan quadrotor berada pada keadaan kuasi-stasioner (keadaan posisi dan orientasi quadrotor hampir tidak berubah), diperoleh :

$$\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \tau^b \quad (2.10)$$

$$ma = -(F_{a1} + F_{a2} + F_{a3} + F_{a4}) \begin{bmatrix} c\phi s\theta c\psi + s\phi s\psi \\ c\phi s\theta s\psi - s\phi c\psi \\ c\phi c\theta \end{bmatrix} \quad (2.11)$$

$$+ \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - \frac{1}{2} \begin{bmatrix} C_x A_c \rho \dot{x} |\dot{x}| \\ C_y A_c \rho \dot{y} |\dot{y}| \\ C_z A_c \rho \dot{z} |\dot{z}| \end{bmatrix}$$

Dengan pengolah persamaan (2.4) hingga (2.11) diperoleh persamaan dinamika quadrotor sebagai berikut.

$$I_{xx}\ddot{\phi} = (I_{yy} - I_{zz})\dot{\theta}\dot{\psi} + (\Omega_{m1}^2 + \Omega_{m4}^2 - \Omega_{m2}^2 - \Omega_{m3}^2)l \sin(\alpha)C_a\rho Ar^2 \quad (2.12)$$

$$I_{yy}\ddot{\theta} = (I_{zz} - I_{xx})\dot{\phi}\dot{\psi} + (\Omega_{m1}^2 + \Omega_{m2}^2 - \Omega_{m3}^2 - \Omega_{m4}^2)l \sin(\beta)C_a\rho Ar^2 \quad (2.13)$$

$$I_{zz}\ddot{\psi} = (I_{xx} - I_{yy})\dot{\theta}\dot{\phi} + C_T\rho Ar^3 \sum_{i=1}^4 (-1)^i \Omega_{mi}^2 \quad (2.14)$$

$$m\ddot{x} = (-\cos\phi \sin\theta \cos\psi - \sin\phi \sin\psi) \cdot C_a\rho Ar^2 \cdot \sum_{i=1}^4 \Omega_{mi}^2 - \frac{1}{2} C_x A_c \rho \dot{x} |\dot{x}| \quad (2.15)$$

$$m\ddot{y} = (-\cos\phi \sin\theta \sin\psi + \sin\phi \cos\psi) \cdot C_a\rho Ar^2 \cdot \sum_{i=1}^4 \Omega_{mi}^2 - \frac{1}{2} C_y A_c \rho \dot{y} |\dot{y}| \quad (2.16)$$

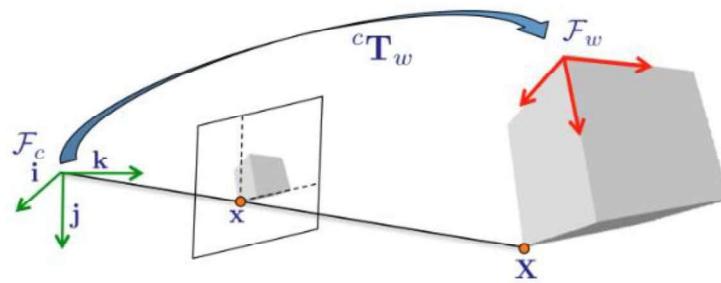
$$m\ddot{z} = -(\cos\phi \cos\theta) \cdot C_a\rho Ar^2 \cdot \sum_{i=1}^4 \Omega_{mi}^2 + mg - \frac{1}{2} C_z A_c \rho \dot{z} |\dot{z}| \quad (2.17)$$

## 2.2 Pengukuran Berbasis Citra

Pengukuran berbasis citra merupakan salah satu metode pengukuran yang menggunakan citra untuk mengetahui atau mengestimasi parameter atau besaran fisis. Pada tugas akhir ini, besaran yang ingin diestimasi adalah posisi. Metode yang digunakan dalam estimasi posisi adalah *Perspective-N-Point* (PnP) *Pose Estimation* dengan menggunakan sistem *fiducial marker* ArUco sebagai alat bantu. Dalam subbab ini juga akan dibahas bagaimana distorsi dapat mempengaruhi citra serta bagaimana formulasi untuk menghilangkan distorsi yang bisa digunakan agar metode *Perspective-N-Point* dapat digunakan.

### 2.2.1 Perspective-N-Point Pose Estimation

*Pose Estimation* adalah sebuah permasalahan yang membahas tentang bagaimana hubungan antara kerangka koordinat sebuah kamera ketika diketahui beberapa set korespondensi antara fiture sebuah gambar pada koordinat 3 dimensi atau koordinat dunia nyata dan representasinya pada kamera [15]. Tujuan dari permasalahan ini adalah mencari hubungan antara kerangka acuan kamera dan dunia nyata dalam 2 buah parameter yaitu vektor translasi dan matriks rotasi. Hubungan antara kedua kerangka acuan dapat dilihat pada Gambar 2.3.



**Gambar 2.3** Hubungan Antara *World Frame* dan *Camera Frame*

Hubungan antara kedua kerangka acuan dihubungkan oleh matriks  ${}^c\mathbf{T}_w$  yang dapat dinyatakan dalam :

$${}^c\mathbf{T}_w = \begin{bmatrix} {}^c\mathbf{R}_w & {}^c\mathbf{t}_w \\ 0_{3 \times 1} & 1 \end{bmatrix} \quad (2.18)$$

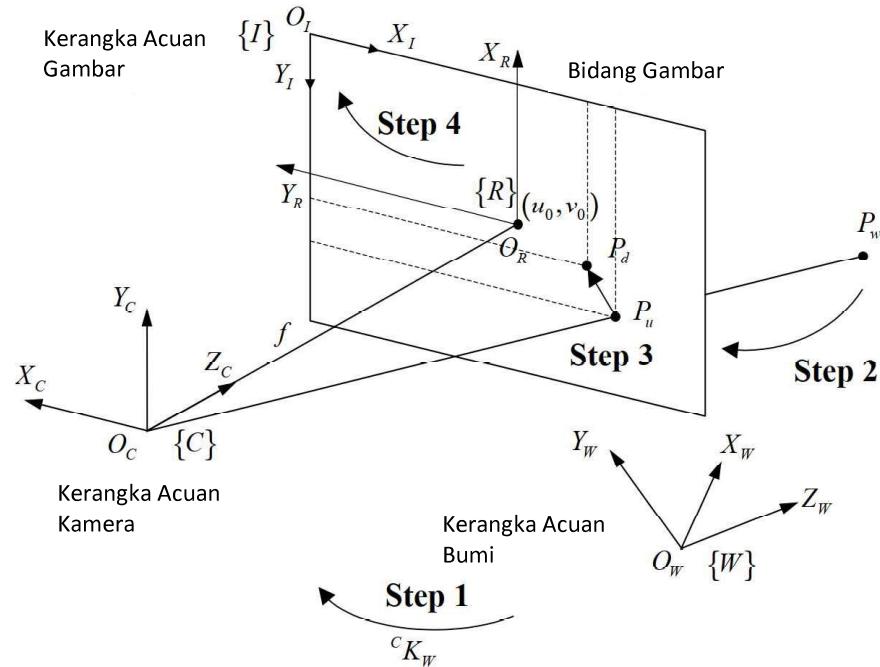
${}^c\mathbf{R}_w$  adalah matriks rotasi berukuran 3x3 dan  ${}^c\mathbf{t}_w$  adalah vektor translasi berukuran 3x1. Dengan mengetahui persamaan transformasi koordinat ini, dapat diketahui hubungan antara koordinat piksel yang terbaca pada citra dan koordinat benda aslinya pada dunia nyata adalah :

$$\bar{x} = K \Pi {}^c\mathbf{T}_w {}^wX \quad (2.19)$$

K di dalam persamaan ini adalah parameter intrinsik kamera yang didefinisikan sebagai :

$$K = \begin{bmatrix} p_x & 0 & u_0 \\ 0 & p_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.20)$$

Dimana  $p_x$  dan  $p_y$  adalah rasio antara *focal length* f dan lebar piksel pada sumbu x dan y ( $l_x$  dan  $l_y$ ), sedang  $u_0$  dan  $v_0$  didefinisikan sebagai *principal point* (titik perpotongan antara sumbu optik dan bidang gambar). Ilustrasi untuk *focal length* dan *principal point* dapat dilihat pada Gambar 2.4.



**Gambar 2.4** Hubungan Geometri Antara Objek 3 Dimensi dan 2 Dimensi [16]

$\Pi$  sendiri adalah matriks proyeksi yang dirumuskan sebagai berikut :

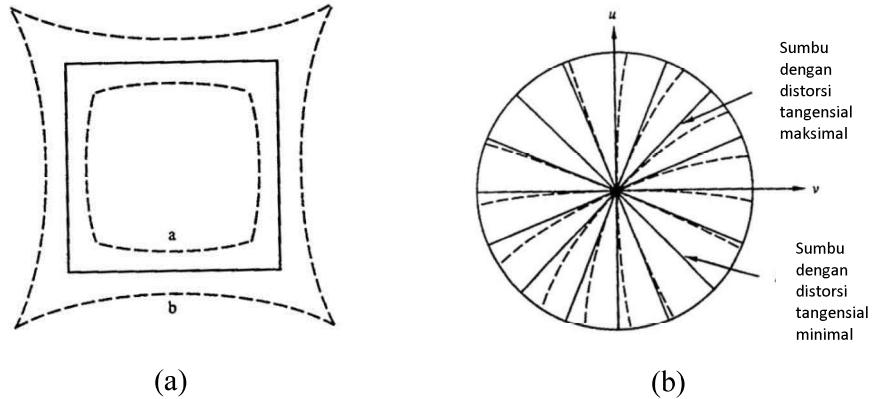
$$\Pi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.21)$$

PnP *pose estimation* dapat diselesaikan dengan menggunakan serangkaian dari persamaan yang telah dijelaskan sebelumnya. Jika ada  $n$  titik yang diketahui, maka setiap koordinat dalam kamera dinyatakan dalam  $\bar{x}_i = (\bar{x}_i, \bar{y}_i, 1)^T$  dan dalam dunia nyata dinyatakan dalam  ${}^w X_i = ({}^w X_i, {}^w Y_i, {}^w Z_i, 1)^T$  sehingga persamaan akhirnya dapat dinyatakan dalam :

$$\bar{x}_i = K \Pi {}^c T_w {}^w X_i \quad (2.22)$$

Dari persamaan tersebut, dapat diambil kesimpulan bahwa akan terdapat total persamaan sejumlah banyak titik yang diketahui korespondensinya. Dengan menyelesaikan persamaan tersebut, pose dari kamera yang dimuat dalam matriks  ${}^c T_w$  akan bisa dikalkulasikan.

## 2.2.2 Distorsi pada Kamera



**Gambar 2.5** Ilustrasi dari Distorsi Radial (a) dan Tangensial (b) [17]

Titik-titik yang ada di kamera bisa saja tidak sesuai dengan aslinya. Peristiwa ini disebut dengan distorsi. Umumnya, efek dari distorsi dapat dibagi menjadi dua yaitu distorsi radial dan distorsi tangensial. Ilustrasi dari kedua tipe distorsi tersebut dapat dilihat pada Gambar 2.5. Persamaan hubungan antara koordinat terdistorsi dan ideal dapat dimodelkan oleh persamaan distorsi Brown [18] yang dinyatakan dalam persamaan berikut :

$$x'_u = x'_d + \delta x \quad (2.23)$$

$$y'_u = y'_d + \delta y \quad (2.24)$$

$$\begin{aligned} \delta x = x'_d (k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6 + \dots) + [p_1(r_d^2 + 2x'_d^2) + 2p_2 y'_d x'_d] (1 \\ + p_3 r^2 d + \dots) \end{aligned} \quad (2.25)$$

$$\begin{aligned} \delta y = y'_d (k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6 + \dots) + [p_1(r_d^2 + 2y'_d^2) + 2p_2 y'_d x'_d] (1 \\ + p_3 r^2 d + \dots) \end{aligned} \quad (2.26)$$

Jika diukur dari pusat optik  $(x_c, y_c)$ , kita mendapatkan

$$x'_u = x_u - x_c \quad (2.27)$$

$$y'_u = y_u - y_c \quad (2.28)$$

$$x'_d = x_d - x_c \quad (2.29)$$

$$y'_d = y_d - y_c \quad (2.30)$$

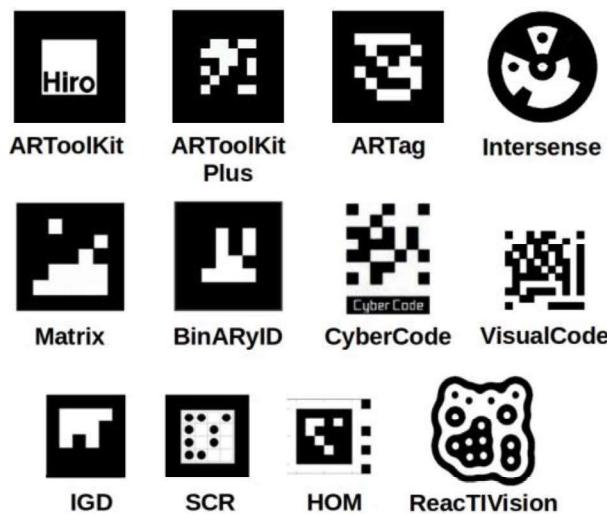
$$r_d = \sqrt{x'_d^2 + y'_d^2} \quad (2.31)$$

Pada persamaan di atas, subskrip  $d$  menyatakan koordinat terdistorsi dan  $u$  menyatakan kordinat ideal. Parameter  $k_i$  dan  $p_i$  menyatakan koefisien distorsi radial dan tangensial secara berurutan. Dengan persamaan koreksi di atas, maka titik koordinat yang terlihat di kamera akan bisa ditemukan bagaimana kondisinya idealnya, yaitu ketika distorsi tidak terjadi.

### 2.2.3 Marka ArUco

Marka ArUco adalah salah satu *fiducial marker* yang dikembangkan dalam penelitian Garrido-Jurado *et al.* [9]. Marka ini ditargetkan aplikasinya untuk *augmented reality*, robot *localization* dan lainnya. Sebelum ArUco, sudah ada banyak sistem *fiducial marker* yang ada seperti yang dapat dilihat pada Gambar 2.6, hanya saja, sistem *fiducial marker* pada ArUco diklaim memiliki beberapa kelebihan, antara lain:

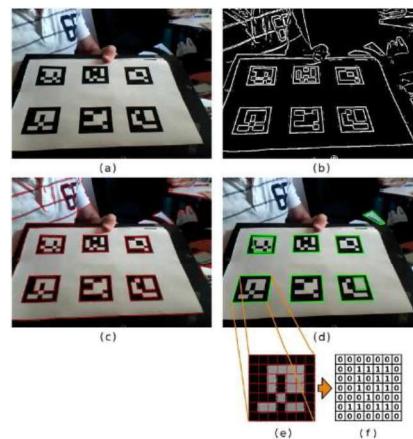
1. Jumlah *dictionary marker* yang lebih banyak dikarenakan *dictionary* yang dimiliki oleh ArUco dibuat menggunakan algoritma *marker generation*, bukan menggunakan *set marker* yang sudah ada.
2. Mempunyai algoritma untuk mendeteksi dan mengoreksi *error* yang dapat mendeteksi dan mengoreksi jumlah *error* yang lebih besar.



**Gambar 2.6** Beberapa contoh *fiducial marker* [9]

Langkah mendeteksi ArUco dalam sebuah citra dapat dilihat pada Gambar 2.7. Detail proses yang dilakukan untuk mengekstraksi ArUco dari sebuah citra adalah sebagai berikut:

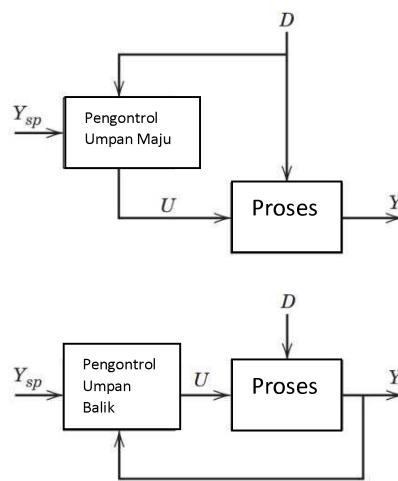
1. Citra asli dikonversi menjadi *grayscale*, kemudian dilakukan *adaptive thresholding* untuk membuat citra menjadi biner. Citra yang sudah dilakukan *thresholding* kemudian dilakukan pula *edge detection* sehingga menghasilkan citra seperti pada Gambar 2.7(b).
2. Gambar yang sudah dilakukan *edge detection* kemudian diekstraksi konturnya sehingga yang diambil hanyalah bagian gambar yang memiliki kontur segi empat seperti yang terlihat pada Gambar 2.7(d).
3. Kontur segiempat yang sudah dideteksi kemudian dihilangkan efek proyeksinya sehingga menjadi seperti Gambar 2.7(e). Kontur ini kemudian diubah menjadi biner dan dibagi menjadi *grid*. Setiap elemen dari *grid* yang sudah dibagi dinyatakan dalam bilangan biner 1 atau 0 tergantung mayoritas piksel yang berada pada *grid* tersebut seperti pada Gambar 2.7(f). Kontur yang dideteksi keseluruhan hanyalah kontur dengan nilai sisi terluar bernilai nol untuk semua elemen.
4. Setelah nilai untuk tiap elemen sudah berhasil dideteksi, tahap selanjutnya adalah pencocokan dengan *dictionary* yang memuat *identifier* (id) dari marka ArUco. Jika nilainya cocok, maka akan dinyatakan valid. Jika tidak, maka akan dicobakan variasi yang lain dengan melakukan rotasi 90° sampai nilai dinyatakan cocok. Jika tidak, maka kontur tadi akan ditolak.



**Gambar 2.7** Urutan Proses Pendeksteksian Marka ArUco [9]

### 2.3 Pengontrol Umpang Balik dan Umpang Maju

Berdasarkan hal yang mempengaruhi aksi kontrol, pengontrol dapat dibagi menjadi dua, yaitu pengontrol umpan balik dan pengontrol umpan maju. Pengontrol umpan balik memberikan aksi kontrol bergantung dengan besaran *error* atau selisih terhadap keadaan yang diinginkan dan keadaan sistem saat ini, sedangkan pengontrol umpan maju memberikan aksi kontrol dengan mengukur serta mengalkulasikan gangguan yang bekerja pada sistem. Perbandingan diagram blok dari kedua sistem pengontrol dapat dilihat pada Gambar 2.8.



**Gambar 2.8** Diagram Blok Pengontrol Umpang Maju (atas) dan Umpang Balik (bawah)  
[19]

Pengontrol umpan balik memerlukan waktu untuk membuat keadaan sistem mencapai *setpoint* dikarenakan aksi yang dilakukan adalah berupa aksi korektif. Solusi yang ditawarkan untuk hal ini adalah adanya pengontrol umpan maju. Hal ini disebabkan pengontrol umpan maju melakukan aksi preventif yang dilakukan berdasarkan gangguan yang ada pada sistem. Walaupun memiliki kelebihan seperti yang disebutkan sebelumnya, terdapat beberapa kekurangan pengontrol umpan maju antara lain diperlukannya pengukuran gangguan secara *online* serta perlunya mengetahui model dari sistem [19].

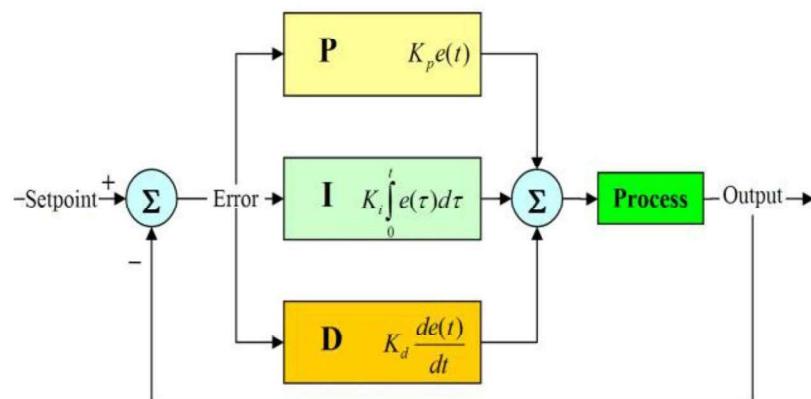
### 2.4 Pengontrol *Proportional-Integral-Derivative*

Salah satu pengontrol umpan balik yang populer digunakan adalah pengontrol (*proportional-integral-derivative*) PID. Pengontrol PID adalah salah satu tipe

pengontrol *feedback* yang berarti pengontrol PID menggunakan nilai *error* atau selisih dari *setpoint* atau nilai yang diinginkan dengan nilai aktual yang dimiliki keluaran sistem. Pengontrol PID terkenal dengan realibilitasnya dikarenakan pengontrol ini dapat berfungsi dengan baik walaupun model matematika dari sistem tidak diketahui atau ketika pendekatan analitik untuk mengontrol sistem tidak dapat digunakan [20].

Pengontrol PID sesuai dengan namanya terdiri dari 3 komponen pengontrol yaitu:

1. Pengontrol proporsional: Pengontrol bagian ini memberikan aksi kontrol tergantung dengan seberapa besar nilai *error* yang dimiliki oleh sistem. Pengontrol tipe ini terkadang tidak dapat berdiri sendiri (tergantung dari dinamikas sistem) karena dapat menyebabkan *steady-state error*.
2. Pengontrol integral: Pengontrol bagian ini memberikan aksi kontrol berdasarkan akumulasi *error*. Dapat digunakan untuk mengeliminasi *steady-state error* dan mempercepat mencapai *setpoint*.
3. Pengontrol derivatif: Pengontrol yang bertindak ketika terjadi perubahan nilai *error*. Berfungsi untuk memperkecil fluktuasi yang terjadi pada sinyal kontrol yang diberikan, atau dengan kata lain dapat melakukan pengereman pada aksi kontrol.



**Gambar 2.9** Diagram Blok Pengontrol PID Paralel [21]

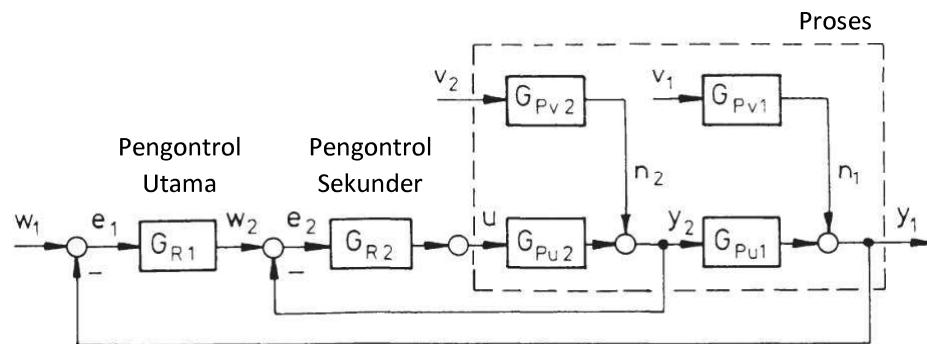
Pengontrol PID yang biasanya digunakan adalah pengontrol PID paralel seperti yang dapat dilihat pada Gambar 2.9. Pengontrol tipe ini menjumlahkan seluruh

komponen untuk diberikan kepada *final element controller*. Persamaan akhir dari PID tipe paralel ini dapat dinyatakan sebagai:

$$U = k_p e + k_i \int e dt + k_d \frac{de}{dt} \quad (2.32)$$

## 2.5 Pengontrol Kaskade

Pengontrol kaskade atau yang dapat dikatakan sebagai pengontrol bertumpuk adalah pengontrol yang melibatkan 2 pengontrol atau lebih dalam 1 lop kontrol utama. Pengontrol tipe ini mempunyai performa yang sangat baik untuk meningkatkan performa pengontrolan apabila gangguan yang ada pada sistem juga mempengaruhi *manipulated variable* [22]. Pengontrolan kaskade memiliki 2 komponen utama, yaitu *main controller* yang berfungsi untuk mengontrol *controlled variable* utama sedangkan sisanya disebut sebagai *auxiliary controller* yang berguna untuk mengontrol *value* dari *manipulated variable* yang *setpoint*-nya ditentukan oleh *main controller* [23].



**Gambar 2.10** Diagram Blok Pengontrol Kaskade [23]

Diagram blok dari pengontrol kaskade dapat dilihat pada Gambar 2.10. Berdasarkan gambar tersebut, dapat dilihat bahwa cara kerja dari pengontrol kaskade adalah sebagai berikut:

1. Dalam lop utama, variabel yang dikontrol adalah  $y_1$  di mana nilainya diatur agar mencapai *setpoint*  $w_1$ . Selisih dari  $w_1$  dan  $y_1$  dianggap sebagai *error* dan diteruskan kepada *main controller*.
2. *Main controller* mengatur berapa nilai *manipulated variable* yang seharusnya agar nilai dari *controlled variable*  $y_1$  dapat menuju nilai *setpoint*  $w_1$ . Akan tetapi, karena nilai *manipulated variable* tidak bisa langsung

dicapai atau adanya gangguan yang menyebabkan nilainya menjadi bukan nilai yang sebenarnya diinginkan, nilai ini dijadikan *setpoint*  $w_2$ .

3. *Auxiliary controller* berfungsi untuk mengatur *manipulated variable* yang lain sehingga setelah melewati blok  $G_{pu2}$ , *manipulated variable* yang dibutuhkan oleh lup utama mencapai nilai *setpoint*  $w_2$ .
4. Nilai keluaran dari lup *auxiliary* akan digunakan untuk menjadi *input* bagi blok  $G_{pu2}$  yang akan menghasilkan keluaran akhir bagi keseluruhan sistem.

Dengan cara kerja tersebut, dapat diambil kesimpulan terkait fungsi transfer akhir sistem yaitu untuk lup *auxiliary*:

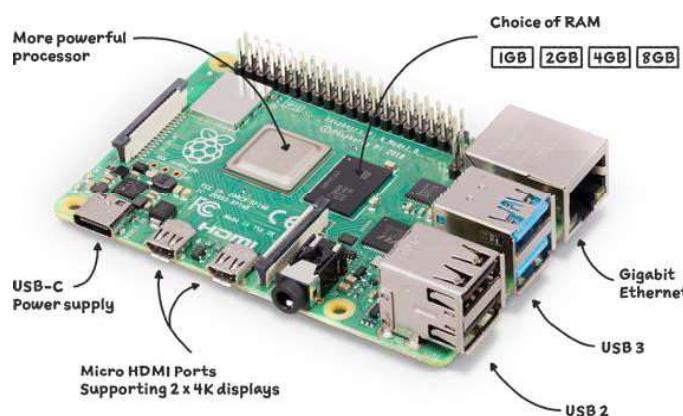
$$G_{aux} = \frac{G_{r2}G_{pu2}}{1 + G_{r2}G_{pu2}} \quad (2.33)$$

Dan untuk lup utama :

$$G_{main} = \frac{G_{r1}G_p}{1 + G_{r1}G_p} = \frac{G_{r1} \frac{(G_{r2}G_{pu2})}{1 + G_{r2}G_{pu2}} G_{pu1}}{1 + G_{r1} \frac{(G_{r2}G_{pu2})}{1 + G_{r2}G_{pu2}} G_{pu1}} \quad (2.33)$$

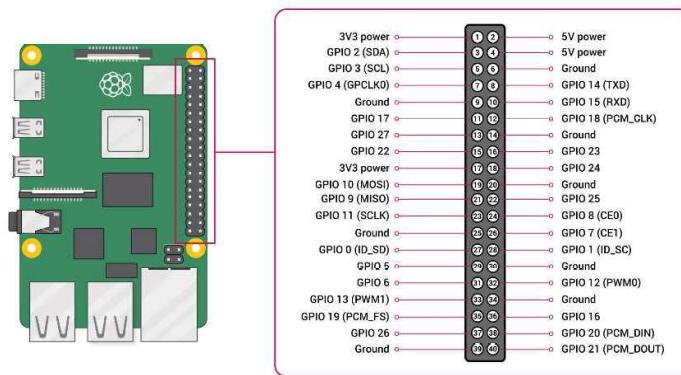
## 2.6 Raspberry Pi

Raspberry Pi merupakan perangkat mini komputer yang beroperasi dengan sistem operasi raspbian. Raspberry Pi memiliki pemakaian yang sangat luas yaitu pada bidang *Internet of Things* (IOT) [24], sistem tertanam [25], robotik [26], dan lainnya. Salah satu model yang dari Raspberry Pi yang digunakan dalam tugas akhir ini adalah Raspberry Pi 4B yang penampilkannya dapat dilihat pada Gambar 2.11.



**Gambar 2.11** Tampilan Fisik Raspberry Pi 4B [27]

Raspberry Pi memiliki banyak perangkat komunikasi yang bisa digunakan untuk berkoneksi dengan berbagai perangkat. Sebuah perangkat Raspberry Pi 4 B dilengkapi dengan masing-masing 2 port USB 2&3, Gigabit *ethernet*, 2 buah *port display micro* HDMI serta 40 pin dengan berbagai kebutuhan. 40 pin tersebut memiliki konfigurasi seperti yang dapat dilihat pada Gambar 2.12. 40 pin tersebut terdiri dari gabungan antara pin *power supply* (3.3V dan 5V), pin *ground* serta pin *General Purpose Input Output* (GPIO) yang dapat berfungsi sebagai pin masukan dan keluaran dengan level tegangan 3.3V. Pin GPIO pada Raspberry Pi juga dapat digunakan untuk protokol komunikasi lain seperti I2C, SPI, PWM, dan juga komunikasi serial [28].



**Gambar 2.12** Diagram Pinout dari Raspberry Pi 4B [28]

Raspberry Pi sendiri memiliki kelebihan dan kekurangan seperti yang dapat dirangkum pada Tabel 2.1 [29].

**Tabel 2.1** Kelebihan dan Kekurangan Raspberry Pi

No.	Kelebihan	Kekurangan
1	Harga yang terjangkau	Penggunaan SD card sebagai penyimpanan internal membuatnya sangat lambat
2	Mendukung penggunaan banyak periferal	Tidak mempunyai unit pemrosesan grafis
3	Dapat menggunakan banyak sensor di waktu yang sama	Mudah mengalami <i>overheating</i>

4	Mendukung pemrograman dengan banyak bahasa pemrograman	Tidak dapat menjalankan sistem operasi windows
5	Mempunyai prosesor berkecepatan tinggi (dibandingkan dengan mikrokontroller jenis lain)	Tidak mempunyai <i>Real-Time Clock</i> (RTC)
6	Dapat digunakan sebagai komputer kecil	

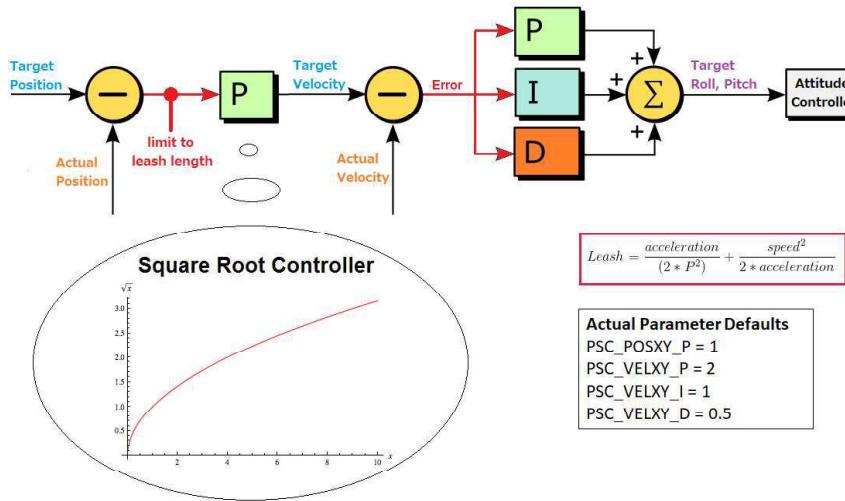
## 2.7 Pixhawk

Pixhawk merupakan perangkat keras yang populer digunakan sebagai *flight controller*. Pixhawk memiliki fungsionalitas gabungan dari perangkat PX4FMU dan PX4IO [30]. Perangkat keras Pixhawk memiliki tampilan seperti yang dapat dilihat pada Gambar 2.13. Perangkat ini memiliki banyak sensor seperti giroskop untuk menentukan arah, akselerometer untuk memperhitungkan pengaruh dari luar, kompas, dan barometer untuk menentukan ketinggian [31]. Pixhawk juga memiliki banyak koneksi yang dapat digunakan untuk terhubung ke sensor atau perangkat lainnya seperti I2C, CAN, ADC, UART, Console, PWM, dan SBUS [30].



**Gambar 2.13** Tampilan Fisik dari Pixhawk 1 3DR [30]

Pixhawk memiliki kemampuan untuk mengontrol besaran posisi dan kecepatan dari wahana terbang. Pengontrol yang digunakan adalah pengontrol tipe kaskade dengan total 3 pengontrol. Pengontrol pertama yaitu pengontrol posisi yang memberikan nilai *setpoint* kecepatan yang diperlukan. Dengan *setpoint* tersebut, Pixhawk mengatur target *attitude* dari objek sehingga nilai kecepatan tersebut dapat dicapai. Diagram blok pengontrolan ini dapat dilihat pada Gambar 2.14.



**Gambar 2.14** Pengontrolan Posisi Oleh Pixhawk [32]

Diagram pengontrolan *attitude* dari quadrotor pula dapat dilihat pada **Lampiran B** Diagram Pengontrol Attitude Quadrotol. Proses pengontrolan juga terdiri dari pengontrol kaskade. Pengontrol *attitude* menerima target sudut yang diinginkan dan memberikan *setpoint* kecepatan putaran tiap motor pada pengontrol motor. Pengontrol motor pada lop kontrol terakhir memberikan sinyal aktuasi pada *Electronic Speed Controller* (ESC) untuk menggerakkan setiap motor sesuai dengan *setpoint* yang sudah dijelaskan sebelumnya.

Dalam hal pengukuran posisi, kecepatan serta orientasi angular, perangkat keras Pixhawk memiliki kemampuan untuk menggunakan *Extended Kalman Filter* (EKF). EKF digunakan untuk menggabungkan hasil dari seluruh pengukuran yang dapat menolak hasil pengukuran yang mempunyai galat yang besar. Dengan adanya algoritma ini, kendaraan menjadi kurang rentan terhadap kesalahan yang memengaruhi satu sensor [33].

## 2.8 Dynamixel XL-320

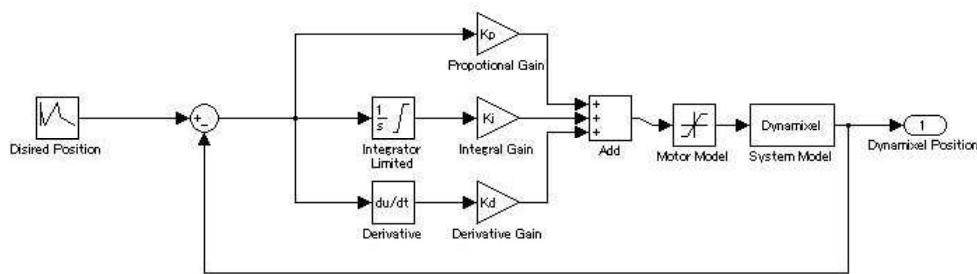
Dynamixel XL-320 merupakan perangkat keras motor servo yang dimanufaktur oleh Robotis. Perangkat ini merupakan gabungan dari motor DC, driver, *controller*, sensor, aktuator, dan gir reduksi. Penampilan dari servo ini dapat dilihat pada Gambar 2.15.



**Gambar 2.15** Tampilan Fisik Servo Dynamixel XL-320

Servo Dynamixel mempunyai performansi yang cukup mumpuni untuk digunakan dalam berbagai keperluan. Dengan sudut putar mencapai  $300^\circ$  (dalam mode *joint*) dengan resolusi mencapai  $0,29^\circ$ , servo ini dapat digunakan untuk aplikasi yang memerlukan sudut putar yang luas dan ketelitian yang sangat tinggi. Servo ini juga dapat beroperasi dengan mode *wheel* yang membuatnya dapat berputar  $360^\circ$  secara terus menerus. Kecepatan dari servo ini dapat mencapai 114 RPM (1,6 putaran per detik) saat tidak ada beban.

Servo Dynamixel XL320 tidak menggunakan sinyal PWM untuk mengatur posisi seperti perangkat servo lainnya [34], melainkan menggunakan pengontrol PID posisi yang diagram bloknya dapat dilihat pada Gambar 2.16. Perbedaan posisi yang diinginkan dengan posisi aktual akan menjadi sinyal *error* yang dikirim ke pengontrol. Sinyal *error* akan menjadi acuan bagi pengontrol dalam menggerakkan motor sehingga posisi yang diinginkan bisa tercapai.



**Gambar 2.16** Pengontrol PID pada Servo

Untuk memerintah servo seperti memberikan *setpoint* atau membaca parameter yang terukur seperti temperatur dan tegangan, servo dapat berkomunikasi dengan

mikrokontroller dengan memberikan paket data digital yang berisikan informasi serta perintah spesifik. Aturan mengenai bagaimana data dikirim beserta isi paket diatur dalam protokol Dynamixel 2.0 [35].

Paket yang dikirimkan ke servo atau yang diterima oleh *controller* terdiri dari struktur yang dapat dilihat pada Gambar 2.17. Struktur tersebut pada dasarnya terdiri dari bit pembuka yang disebut sebagai *header* dan bit penutup yang berupa *Cycling Redundancy Check* (CRC) yang berfungsi untuk memastikan paket yang diterima atau dikirim dalam kondisi tidak bermasalah. Bit yang berada di tengah berisikan informasi penting seperti ID servo yang menerima perintah, panjang bit total, perintah yang diberikan serta parameter berkaitan dengan perintah yang diberikan.

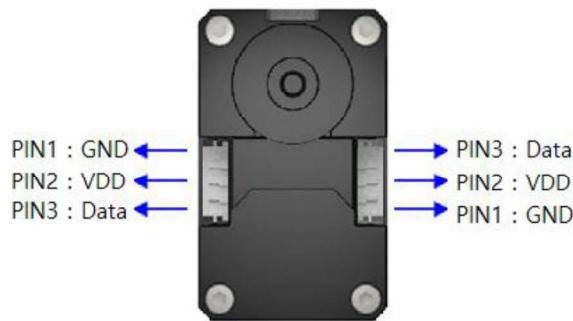
Header 1	Header 2	Header 3	Reserved	Packet ID	Length 1	Length 2	Instruction	Param	Param	Param	CRC 1	CRC 2
0xFF	0xFF	0xFD	0x00	ID	Len_L	Len_H	Instruction	Param 1	...	Param N	CRC_L	CRC_H

Header 1	Header 2	Header 3	Reserved	Packet ID	Length 1	Length 2	Instruction	ERR	PARAM	PARAM	PARAM	CRC 1	CRC 2
0xFF	0xFF	0xFD	0x00	ID	Len_L	Len_H	Instruction	>Error	Param 1	...	Param N	CRC_L	CRC_H

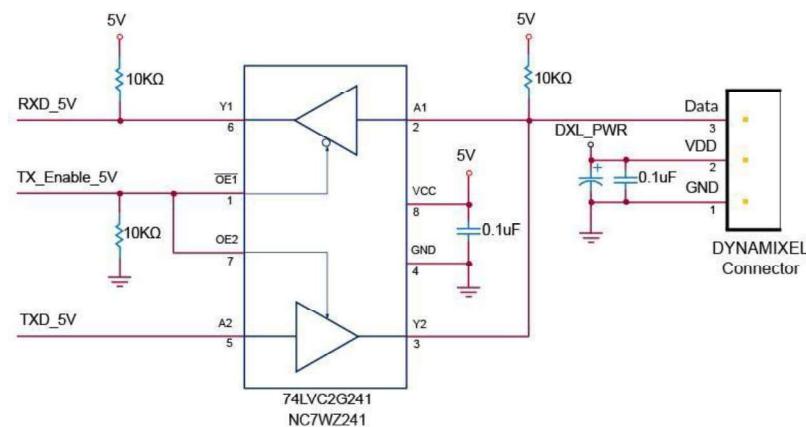
**Gambar 2.17** Urutan Bit Masuk dan Keluar dalam Dynamixel Protocol 2.0

Untuk koneksi sendiri, servo ini menggunakan protokol TTL Half Duplex Asynchronous Serial Communication dan koneksi fisik TTL Multidrop Bus 3 pin yang terdiri dari pin data, vcc, dan gnd yang dapat dilihat pada Gambar 2.18. Hal ini berarti servo ini hanya menggunakan satu kabel untuk menerima serta mengirim data, membuatnya tidak bisa menerima data dan mengirimkannya sekaligus. Alternatif solusi yang bisa digunakan untuk membuat komunikasi menjadi bisa mengirim dan menerima data adalah dengan menggunakan *flow control* yang salah satu rangkaianya dapat menggunakan contoh seperti Gambar 2.19. Pada rangkaian tersebut, *controller* yang digunakan menggunakan satu pin tambahan (tx\_enable) yang akan bernilai 1 jika ingin mengirim data dan akan bernilai 0 jika ingin menerima data (atau bisa sebaliknya, tergantung konfigurasi). Kehadiran tx\_enable ini akan membuat arus data hanya akan dibuka sesuai keperluan sehingga tidak akan ada tabrakan antara data yang ingin diterima serta data yang ingin dikirimkan.



**Gambar 2.18** Diagram Koneksi Servo XL-320

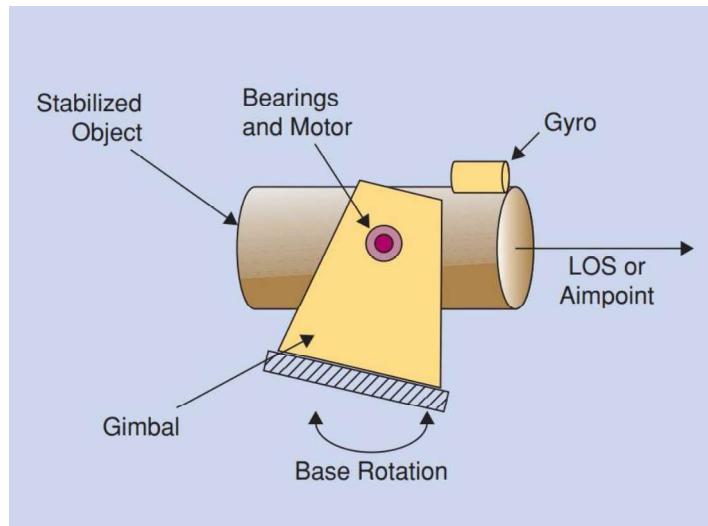
**TTL Communication Circuit**



**Gambar 2.19** Rangkaian Tambahan Komunikasi Dua Arah

## 2.9 Inertial Stabilized Platform

*Inertial Stabilized Platform* (ISP) merupakan sebuah teknologi untuk mempertahankan *Line of Sight* (LOS) dari perangkat yang butuh perargetan seperti kamera, teleskop, sensor inframerah, persenjataan dan lainnya. LOS tersebut bisa berupa *aimpoint* dari sebuah laser atau senjata, atau bisa berupa *field of view* (FOV) dari kamera atau teleskop. Ilustrasi dari sebuah ISP sederhana yang mengontrol hanya satu sumbu dapat dilihat pada Gambar 2.20.



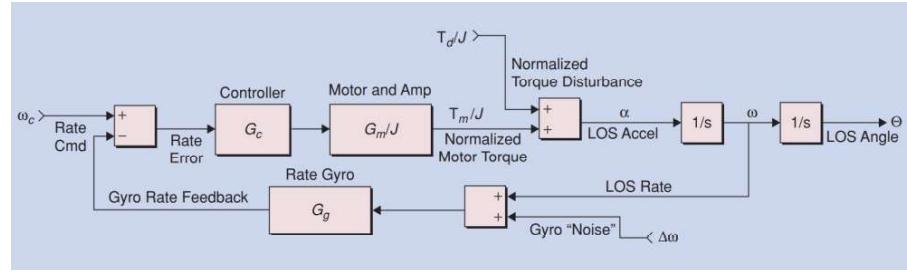
**Gambar 2.20** ISP 1 Sumbu [36]

Secara umum, persamaan dari pergerakan ISP dapat dinyatakan dalam persamaan berikut

$$\omega_{target/FOV} = V_{T\perp}/R - V_{i\perp}/R + \omega_M - \omega_i \quad (2.3)$$

$\omega_{target/FOV}$  adalah pergerakan yang diinginkan yang merupakan penjumlahan dari *parallactic motion*, baik dari sensor maupun target itu sendiri ( $V_{T\perp}/R - V_{i\perp}/R$ ), dan pergerakan yang disebabkan oleh distorsi media dan gerakan inersia dari sensor ( $\omega_M - \omega_i$ ).

Salah satu contoh diagram blok untuk pengontrolan ISP satu sumbu dapat dilihat pada Gambar Gambar 2.21. Pada gambar tersebut, terlihat bahwa pengontrol menggunakan umpan balik dari laju perubahan sudut LOS dan serta perubahan sudut yang disebabkan oleh faktor eksternal. Umpan balik tersebut digunakan oleh pengontrol untuk memberikan aksi pengontrolan berupa gerakan motor yang akan memutar sumbu tersebut sehingga laju perubahan sudut bisa sesuai dengan yang diinginkan pada *setpoint* sehingga pada akhirnya sudut LOS akan tetap sesuai dengan yang diinginkan.



**Gambar 2.21** Diagram Blok Pengontrolan ISP [36]

## 2.10 Software

Dalam tugas akhir ini, terdapat dua pustaka utama yang digunakan. Pustaka pertama adalah OpenCV yang digunakan untuk pemrosesan citra sedangkan yang kedua adalah Dronekit yang digunakan untuk berkomunikasi (memberikan perintah dan menerima informasi) dari quadrotor.

### 2.10.1 OpenCV

OpenCV [37] adalah pustaka sumber terbuka yang digunakan dalam visi komputer. OpenCV dirancang untuk efisiensi dari sisi komputasi dan aplikasi waktu nyata. Salah satu tujuan utama dari adanya OpenCV adalah menyediakan infrastruktur visi komputer yang mudah digunakan yang membantu proses untuk membangun aplikasi visi yang cukup canggih dengan cepat. OpenCV memiliki lebih dari 500 fungsi yang menyebar di banyak bidang seperti pabrik, inspeksi produk, pencitraan medis, keamanan, antarmuka pengguna, kalibrasi kamera, penglihatan stereo, dan robotika [38].

## 2.10.2 Dronekit

Dronekit [39] merupakan pustaka yang memperbolehkan pengembang untuk mengomunikasikan *companion computer* (komputer/perangkat yang ikut bergerak bersama kendaraan) dengan *flight controller* ardupilot menggunakan hubungan dengan latensi rendah. Adanya aplikasi *onboard* ini memungkinkan peningkatan pada kecerdasan perilaku dari kendaraan, meningkatkan kemampuan autopilot serta memungkinkan penyelesaian tugas yang memiliki komputasi yang intensif atau sensitif terhadap waktu.

Dronekit menghubungkan perangkat *companion computer* dengan *flight controller* menggunakan protokol MAVLink. Protokol MAVLink [40] sendiri adalah protokol pengiriman pesan yang dapat digunakan untuk komunikasi dengan drone dan komponen-komponennya. Fitur kunci dari protokol MAVlink ini adalah sebagai berikut.

1. Efisien, karena memiliki *overhead* yang rendah untuk setiap packet yang dikirimkan.
2. Telah digunakan sejak tahun 2009 dengan berbagai tantangan dalam komunikasi.
3. Dapat menggunakan banyak bahasa pemrograman.
4. Mendukung 255 sistem dalam jaringan secara bersamaan.
5. Dapat mendukung komunikasi *offboard* dan *onboard*.

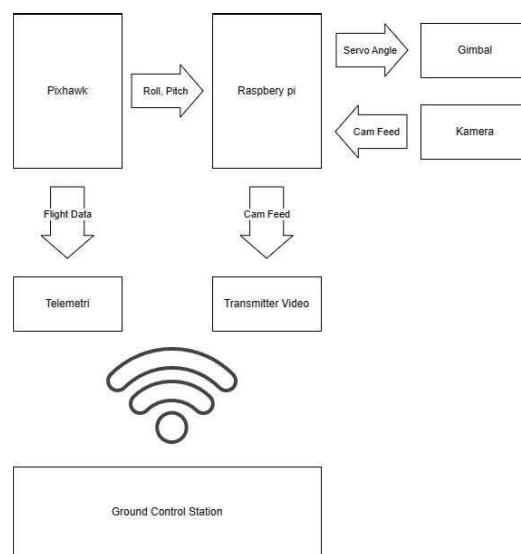
## BAB III

### METODOLOGI DAN PERANCANGAN SISTEM

Untuk mencapai tujuan dari tugas akhir ini, dibuatlah suatu sistem yang terdiri atas gabungan sistem instrumentasi dan sistem kontrol. Sistem instrumentasi terdiri dari sistem pengambilan citra menggunakan kamera dan stabilisasi dari kamera menggunakan perangkat keras motor servo. Sistem instrumentasi ini menghasilkan data koordinat yang kemudian akan diumpulkan kepada sistem kontrol untuk mengontrol pergerakan quadrotor ketika proses pendaratan dilakukan.

#### 3.1 Sistem Instrumentasi Pendaratan

Sistem instrumentasi terdiri dari kamera yang digunakan untuk menangkap citra dari marka ArUco untuk menentukan posisi dari quadrotor relatif terhadap lokasi pendaratan dan sistem stabilisasi kamera untuk menjaga orientasi kamera tetap ke arah bawah walaupun quadrotor bergerak yang menyebabkan perubahan sudut *roll* dan *pitch*. Semua sistem yang digunakan, baik kamera dan gimbal terintegrasi dengan perangkat Raspberry Pi dan Pixhawk untuk pembacaan sensor, penggerakan aktuator, dan pemrosesan gambar. Arsitektur dari sistem instrumentasi dapat dilihat pada Gambar 3.1.

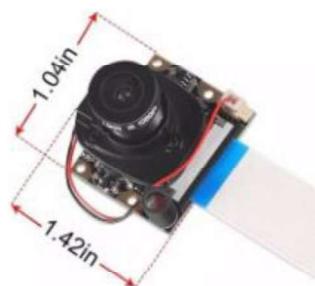


**Gambar 3.1** Arsitekur Sistem Instrumentasi

Pada sistem ini, Raspberry Pi digunakan untuk menangkap citra kamera dan mengolahnya sehingga mendapatkan koordinat lokasi pendaratan relatif terhadap posisi quadrotor. Raspberry Pi digunakan juga untuk mendapatkan informasi tentang sudut Euler (*roll* dan *pitch*) dari quadrotor yang dibaca serta dikalkulasi oleh Pixhawk. *Output* video dari Raspberry Pi juga dikoneksikan dengan *analog video transmitter* TS832. Hal ini dilakukan agar kondisi dari lokasi citra pendaratan dapat dipantau menggunakan perangkat lain seperti laptop atau *smartphone*, sehingga memastikan proses eksperimen dapat berjalan dengan aman.

### 3.1.1 Sistem Pengukuran Berbasis Citra

Untuk mendapatkan citra dari lokasi pendaratan, digunakan kamera yang kompatibel dengan Raspberry Pi 4 B yang digunakan dalam tugas akhir ini. Kamera yang akhirnya digunakan adalah kamera Raspberry Pi Camera OV5647 dengan tampilan seperti pada Gambar 3.2. Kamera ini memiliki spesifikasi seperti yang ditampilkan pada Tabel 3.1. Pada seluruh tahapan kamera, resolusi kamera dibatasi hanya pada 640x480 piksel menyesuaikan resolusi tampilan komposit pada Raspberry Pi.



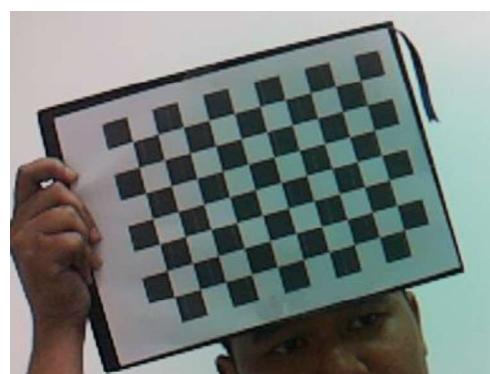
**Gambar 3.2** Pi Camera OV5647 dengan IR

**Tabel 3.1** Spesifikasi Kamera yang Digunakan

Spesifikasi	Nilai
Resolusi	2592 x 1944 Max
Resolusi sensor	5MP
Spesifikasi	Nilai

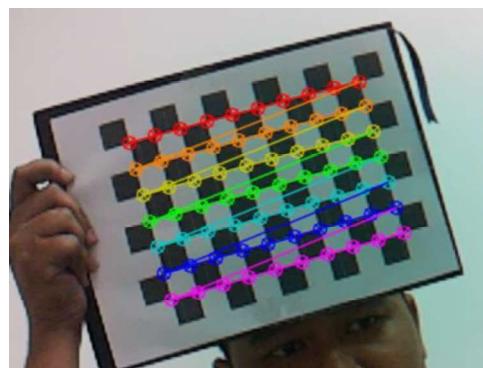
Ukuran CCD	¼ inci
Bukaan lensa (F)	1,8
FOV	75°
Focal length	3,6 mm
Jarak Pandang	Di atas 5 cm
Frame Rate Video	1080p di 30 <i>Frame per Second</i> (H.264); 90 <i>Frame per Second</i> di kualitas VGA
Ukuran	25mm x 24mm

Untuk mengetahui koordinat dari lokasi pendaratan yang ditandai dengan marka ArUco, dibuatlah sistem pengukuran berbasis citra lokasi pendaratan yang dapat mengukur posisi serta jarak lokasi pendaratan menggunakan satuan *centimeter*. Tahapan dalam perancangan sistem ini diawali dengan awalnya kalibrasi, kemudian dilanjutkan dengan pengukuran koordinat objek. Keduanya dilakukan dengan menggunakan algoritma PnP yang sudah dijelaskan pada bagian 2.2.1. Bedanya, kalibrasi digunakan untuk mendapatkan parameter intrinsik kamera berupa *focal length* serta *principal point*. Sedangkan pada proses pengukuran, hal yang diperlukan adalah untuk mencari matriks transformasi koordinat kamera ke pusat koordinat ArUco atau disebut parameter ekstrinsik kamera.



**Gambar 3.3** Papan Catur Ukuran 7x10

Kalibrasi dilakukan dengan menggunakan checkerboard berukuran 8x11 dengan titik potong sebanyak 7x10 seperti terlihat pada Gambar 3.3. Gambar yang diambil dalam proses kalibrasi diekstraksi seluruh titiknya dan diambil koordinat dari seluruh titiknya. Titik yang sudah diekstraksi dapat dilihat pada Gambar 3.4.



**Gambar 3.4** Titik Potong yang Dideteksi di Papan Catur

Titik-titik yang sudah diekstraksi dari 11 gambar berbeda kemudian dimasukkan ke dalam fungsi kalibrasi yang dimiliki oleh pustaka OpenCV. Hasil dari program tersebut menghasilkan parameter instrinsik kamera yang dapat dilihat pada Tabel 3.2. Parameter ini kemudian digunakan untuk estimasi posisi dari marka ArUco yang akan dijelaskan pada paragraf berikut.

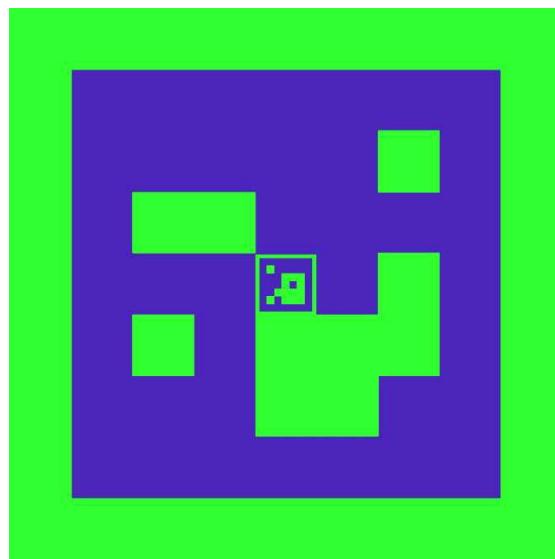
**Tabel 3.2** Parameter Intrinsik Kamera

Parameter	Satuan	Nilai
<i>Focal length</i>	Piksel	(736,97; 732,25)
<i>Optical center</i>	Piksel	(34,88; 208,13)
Koefisien distorsi radial	-	(-0,44; 0,9457; -3,14)
Koefisien distorsi tangensial	-	(0,0024; -0,0028)

Untuk bagian estimasi jarak, digunakan algoritma PnP dalam pustaka OpenCV untuk menentukan nilai matriks transformasi yang dimiliki oleh sistem pengukuran dengan menganggap pusat koordinat dalam *world frame* adalah pusat dari marka ArUco. Dengan anggapan ini, dapat diambil kesimpulan bahwa jarak dari kamera ke pusat marka ArUco adalah sama dengan parameter yang tertera dalam vektor translasi. Hal yang perlu dilakukan untuk bagian ini adalah mengetahui ukuran dari marka ArUco. Dengan mengetahui ukuran (panjang sisi) dari marka ArUco,

koordinat dari 4 titik sudut dari marka tersebut dapat diketahui relatif terhadap pusat marka sehingga memungkinkan untuk melakukan kalkulasi PnP.

Algoritma untuk mengestimasi jarak dari marka ArUco dilakukan dengan menggunakan salah satu fungsi yang terdapat dalam pustaka OpenCV yaitu `EstimatePoseSingleMarker()`. Algoritma ini dijalankan usai pendektsian ArUco sudah dilakukan sehingga koordinat titik sudut dari marka ArUco terhadap *camera frame* dapat diketahui. Fungsi ini membutuhkan masukan seperti yang terdapat pada persamaan (2.19) yaitu matriks parameter intrinsik kamera, koordinat pada *world frame* (yang diketahui melalui panjang marka), koordinat pada gambar yang sudah diketahui sebelumnya, serta koefisien distorsi yang bertujuan untuk mengoreksi posisi titik yang memiliki *error* akibat distorsi radial maupun tangensial. Algoritma ini pada akhirnya akan mendapatkan matriks rotasi dan vektor translasi untuk *world frame*. Nilai dari setiap elemen dari vektor translasi kemudian dijadikan nilai untuk posisi relatif marka terhadap posisi kamera, yang dalam kasus ini dapat dianggap sebagai koordinat lokasi pendaratan relatif terhadap quadrotor.



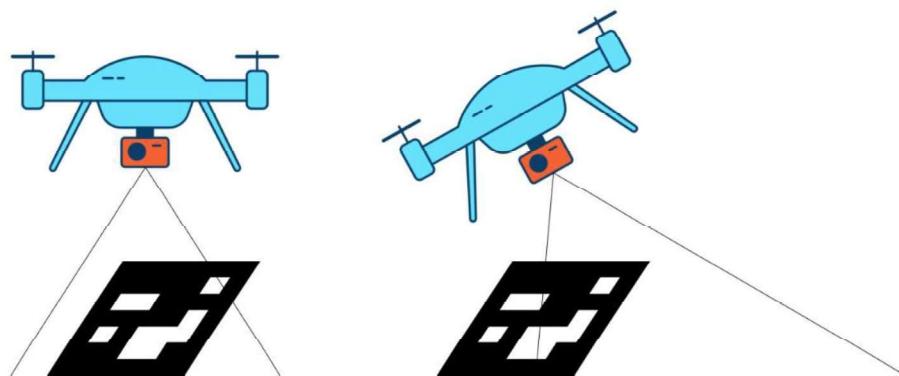
**Gambar 3.5** Marka ArUco Bertumpuk

Marka ArUco yang digunakan pada sistem ini berupa marka bertumpuk dengan id 1 dan 3 seperti yang dapat dilihat pada Gambar 3.5. Alasan pemilihan id tersebut adalah marka yang berada di bagian luar memiliki titik tengah yang berwarna hitam,

sehingga jika marka yang lebih kecil akan memiliki nilai pixel rata-rata rendah (tetap terbaca hitam). Penumpukan ini dilakukan agar marka dapat tetap terdeteksi walau quadrotor berada di dengan posisi yang sangat dekat, yang memungkinkan pengukuran posisi dapat terus berlangsung dan tidak ada permasalahan pada bagian kontrol nantinya.

### 3.1.2 Sistem Stabilisasi Kamera

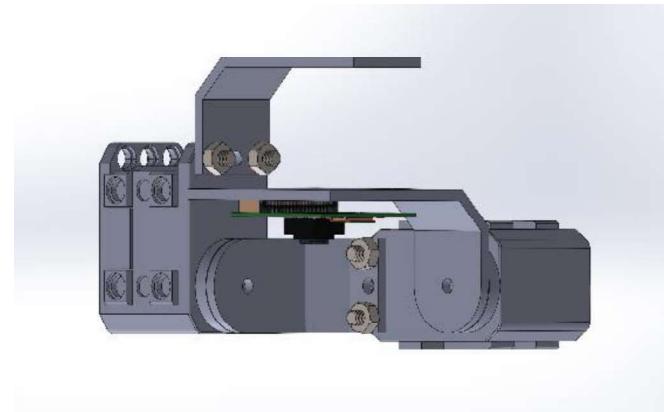
Selama bergerak, quadrotor mengubah sudut *roll* dan *pitch* untuk memindahkan gaya angkat yang semula ke atas menjadi sebagain ke arah horizontal. Akibat dari manuver ini, kamera yang berada pada bagian bawah quadrotor akan mengalami perubahan *field of view*, sehingga marka yang seharusnya terlihat menjadi tidak terlihat/terpotong seperti yang dapat dilihat pada Gambar 3.6. Prediksi dari jarak pun akhirnya tidak menjadi akurat akibat adanya perubahan dari rotasi kamera. Untuk memperbaiki masalah ini, dirancanglah sistem stabilisasi yang menjaga LOS dari kamera untuk tetap mengarah ke bawah sehingga FOV dari kamera bisa lebih luas untuk mendeteksi marka ArUco.



**Gambar 3.6** Ilustrasi FOV Tanpa Gimbal

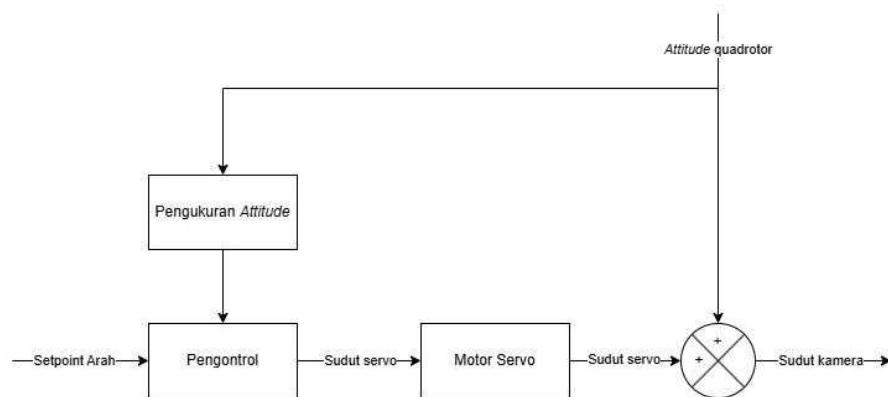
Dikarenakan ada 2 sudut yang perlu dikompensasi, sistem stabilisasi yang dirancang adalah sistem stabilisasi yang terdiri dari 2 sumbu. Desain mekanikal dari sistem ini dapat dilihat pada Gambar 3.7. Terdapat 2 motor servo yang digunakan dengan total 3 sambungan. Sambungan pertama digunakan untuk menghubungkan rangkaian servo dengan bagian bawah pada quadrotor. Sambungan kedua digunakan untuk menghubungkan servo 1 dan 2, yang akan mengompensasi sudut *roll*. Terakhir, sambungan ketiga menghubungkan servo dengan dudukan kamera

untuk mengompensasi sudut *pitch*. Sambungan didesain sedimikian rupa sehingga memiliki bentuk yang ringkas dan tidak terlalu panjang, sehingga defleksi dari gimbal tidak terlalu mempengaruhi pengukuran kamera.



**Gambar 3.7** Desain Mekanikal dari Gimbal

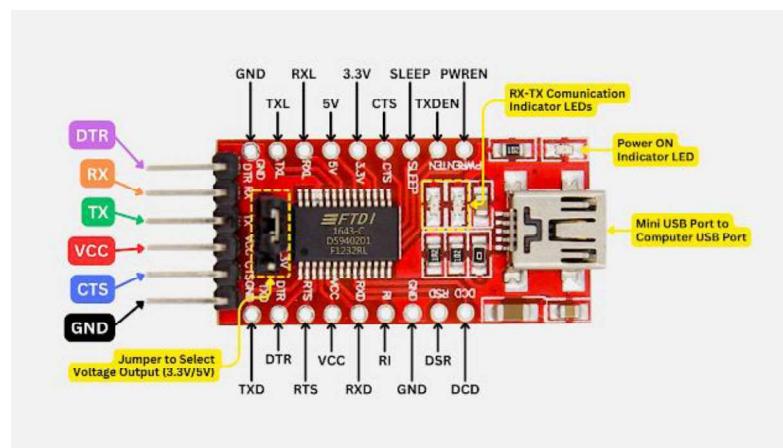
Pengontrolan sistem stabilisasi dilakukan secara umpan maju yang diagram bloknya dapat dilihat pada Gambar 3.8. Gangguan (dalam hal ini adalah sudut quadrotor) akan diukur melalui sistem pengukuran yang terdapat pada Pixhawk dibaca oleh pengontrol Raspberry Psi. Pengontrol tersebut kemudian memerintahkan motor servo untuk bergerak berlawanan dengan arah sudut dari quadrotor. Dengan melakukan hal ini, sudut dari kamera dapat selalu diarahkan ke bagian bawah dan permasalahan yang disebutkan sebelumnya dapat teratasi.



**Gambar 3.8** Diagram Blok Pengontrolan Gimbal

Pada aplikasi koneksinya, pengontrolan servo tidak menggunakan pin UART bawaan yang dimiliki oleh Raspberry Pi. Hal ini disebabkan perbedaan level

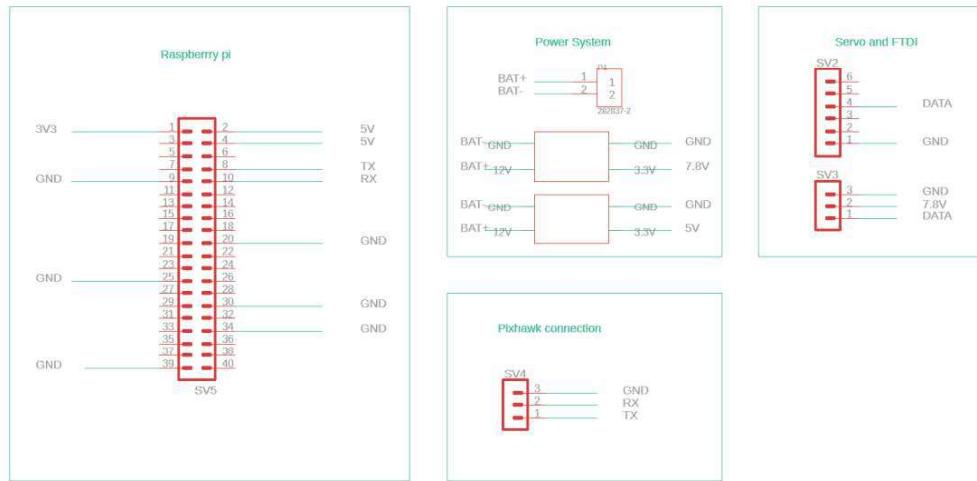
tegangan pada komunikasi keduanya. Raspberry Pi menggunakan komunikasi 3,3 V sedangkan servo yang digunakan mempunyai level tegangan komunikasi sebesar 5 V. Solusi dari masalah ini adalah dengan menggunakan perangkat tambahan berupa *USB to TTL converter* yang menggunakan *chip* dari FTDI seperti yang dapat dilihat pada Gambar 3.9. Perangkat ini dapat membuat Raspberry Pi dapat berkomunikasi secara UART dengan motor servo dengan tegangan 5V. Pada sistem ini juga, komunikasi juga dibiarkan *half duplex* dengan hanya menggunakan pin transmisi. Hal ini dilakukan untuk menghemat komponen dan juga karena data *feedback* dari servo tidak dibutuhkan.



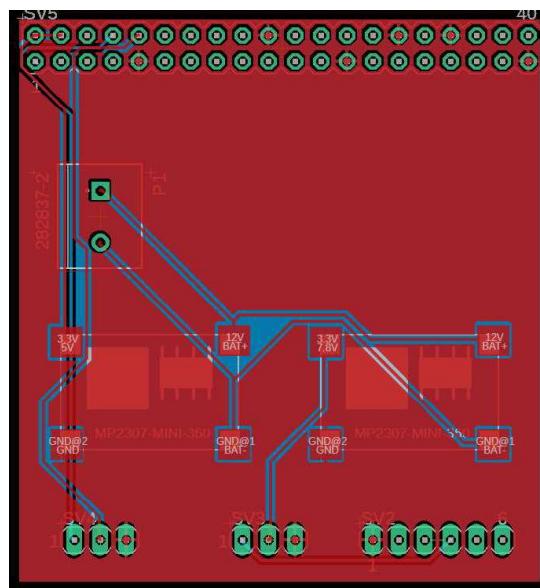
**Gambar 3.9 FTDI USB to TTL Converter**

### 3.1.3 Integrasi Sistem Instrumentasi

Untuk menggabungkan keseluruhan sistem menjadi satu kesatuan dan menggabungkannya dengan sistem quadrotor, dibuatlah sebuah *Printed Circuit Board* (PCB) yang menghubungkan seluruh rangkaian. Skematik dan *board* dari rangkaian tersebut dapat dilihat pada Gambar 3.10 dan Gambar 3.11.



**Gambar 3.10** Skematik Rangkaian Instrumentasi



**Gambar 3.11** Board PCB Rangkaian yang Digunakan

Rangkaian yang telah ditunjukkan memiliki komponen sebagai berikut:

1. 2 DC Step Down Mini 360 dengan 2 level tegangan berbeda. Level pertama di 5 V digunakan untuk memberikan daya pada Raspberry Pi. Komponen yang lain memiliki level tegangan 7,8 V, digunakan untuk memberi daya pada motor servo dan *video transmitter*.
2. 3 pin untuk menghubungkan Pixhawk dan Raspberry Pi.
3. 3 pin untuk memberi daya serta memberi data pada servo.

4. 6 pin untuk menghubungkan keluaran FTDI USB converter dengan servo. Sebagai catatan, FTDI USB converter dikoneksikan ke Raspberry Pi menggunakan kabel mini USB, kemudian keluarannya dihubungkan dengan pin yang ada di PCB.
5. 2 pin untuk menghubungkan keseluruhan rangkaian dengan baterai LiPo 3S. Penggunaan baterai 3S dipilih untuk memastikan tegangan berada cukup tinggi, namun tidak terlalu berat sehingga pergerakan quadrotor masih bisa stabil.

### 3.2 Sistem Pengontrolan Pendaratan

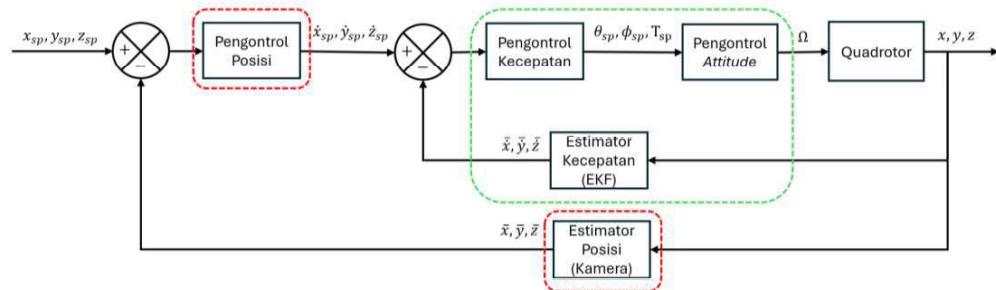
Sistem pengontrolan pendaratan digunakan untuk mengatur posisi dari quadrotor sehingga dapat mencapai lokasi pendaratan. Dalam tugas akhir ini, pengontrol dibuat sehingga pendaratan dapat dilakukan dalam dua kondisi berbeda, yaitu dalam kondisi dukungan GPS tersedia dan tidak. Oleh karena itu, dirancang 2 pengontrol berbeda yang bekerja menyesuaikan kehadiran dukungan GPS.

Perbedaan dari kedua sistem kontrol yang dirancang terletak pada keluaran *setpoint* yang diberikan pada flight controller. Pada skema pendaratan dengan GPS, sinyal kontrol yang diberikan berupa *setpoint* kecepatan. Hal ini dimungkinkan dikarenakan GPS bersamaan dengan *Inertial Measurement Unit* (IMU) dapat digunakan untuk mengestimasi kecepatan dari quadrotor menggunakan EKF, sehingga umpan balik dari kecepatan quadrotor dapat diterima sehingga kecepatan dari quadrotor dapat dikontrol. Pada skema pendaratan tanpa GPS, estimasi kecepatan tidak dapat dilakukan. Oleh karena itu, sinyal kontrol yang diberikan adalah berupa sudut Euler (*roll* dan *pitch*) serta fraksi dari thrust (gaya dorong) yang bekerja pada quadrotor. Detail dari perancangan sistem kontrol akan dijelaskan pada subbab berikutnya.

#### 3.2.1 Pengontrol Dengan Dukungan GPS

Dalam lingkungan dengan dukungan GPS, sistem kontrol pendaratan yang dirancang pada dasarnya cukup mirip dengan sistem kontrol posisi yang dimiliki oleh *flight controller* Pixhawk seperti yang ditunjukkan pada Gambar 2.14. Perbedaan yang terdapat pada sistem kontrol ini adalah pada sistem estimasi posisi. Pada pengontrol yang ada pada flight controller sebelumnya, estimasi posisi

dilakukan dengan gabungan pengukuran antara GPS dan IMU, sedangkan pada pengontrol yang dirancang saat ini, sistem estimasi posisi tersebut diganti menggunakan sistem estimasi berbasis citra yang telah dirancang sebelumnya.



**Gambar 3.12** Diagram Blok Pengontrol Dengan Dukungan GPS

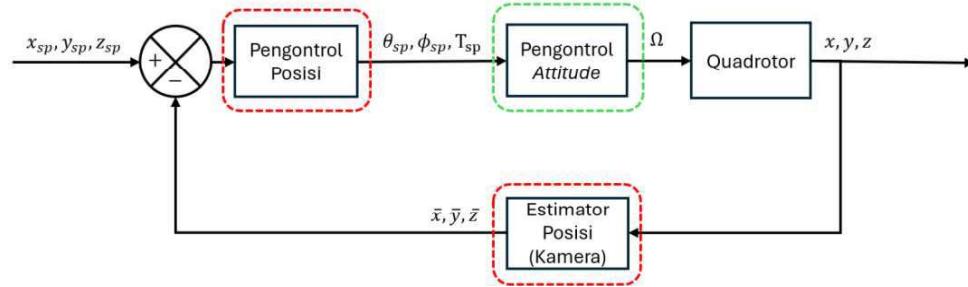
Diagram blok pengontrol dengan dukungan GPS dapat dilihat pada Gambar 3.12. Blok yang berada di dalam garis putus-putus berwarna merah adalah bagian yang bekerja pada perangkat Raspberry Pi, sedangkan bagian yang berada dalam garis putus-putus berwarna hijau merupakan bagian yang bekerja pada *flight controller* Pixhawk.

Cara kerja dari pengontrol untuk skema ini adalah dengan memberikan *setpoint* posisi pada pengontrol posisi. Dengan membandingkan perbedaan nilai dari hasil estimasi, pengontrol posisi memberikan setpoint kecepatan pada flight controller agar nilai setpoint posisi dapat tercapai. Pengontrol kecepatan pula akan memberikan setpoint sudut roll dan pitch serta thrust pada pengontrol attitude sehingga nilai kecepatan yang ditentukan oleh setpoint dapat tercapai. Keluaran akhir dari pengontrol ini adalah berupa kecepatan dari setiap rotor yang akhirnya membuat quadrotor bergerak mencapai setpoint posisi. Pengontrol ini akan terus bekerja sampai pada akhirnya proses pendaratan selesai dilakukan.

### 3.2.2 Pengontrolan Tanpa Dukungan GPS

Dalam skema pengontrolan tanpa GPS, estimator kecepatan tidak dapat digunakan, sehingga aksi kontrol berupa *setpoint* kecepatan tak lagi dapat digunakan. Sebagai alternatif, aksi kontrol keluaran dari pengontrol posisi yang digunakan adalah *setpoint* sudut *roll* dan *pitch* serta *thrust* yang diumpulkan langsung pada

pengontrol *attitude* quadrotor. Hal tersebut menghasilkan sistem kontrol seperti yang terlihat pada diagram blok pada Gambar 3.13.



**Gambar 3.13** Diagram Blok Pengontrol Tanpa Dukungan GPS

Pengontrol ini pada dasarnya mengontrol posisi dengan mengatur percepatan pada quadrotor. Hal ini ditunjukkan dari hubungan antara sudut *roll* dan *pitch* serta *thrust* (yang diwakilkan oleh kecepatan sudut quadrotor) dalam persamaan (2.15) hingga (2.17). Permasalahan yang timbul dalam perancangan ini adalah sudut *roll* dan *pitch* mempengaruhi percepatan di sumbu x, y, dan z jika dilihat dari persamaan tersebut. Hal ini membuat semua lup menjadi berkaitan dan akan mengurangi kestabilan sistem. Oleh karena itu, butuh adanya beberapa batasan agar sistem lup pada sistem pengontrolan ini tidak saling mempengaruhi.

Batasan yang pertama digunakan adalah terkait dengan arah *heading* dari quadrotor. Pada sistem pengontrolan ini, *heading* tidak diatur, atau dalam kata lain, pendaratan dibebaskan untuk dilakukan dalam orientasi *heading* apapun. Dengan hal ini, sudut *yaw* dari quadrotor dapat selalu dianggap nol sehingga persamaan tadi akan menjadi seperti sebagai berikut

$$m\ddot{x} = (-\cos \phi \sin \theta) \cdot C_a \rho A r^2 \cdot \sum_{i=1}^4 \Omega_{mi}^2 - \frac{1}{2} C_x A_c \rho \dot{x} |\dot{x}| \quad (3.1)$$

$$m\ddot{y} = (\sin \phi \cos \psi) \cdot C_a \rho A r^2 \cdot \sum_{i=1}^4 \Omega_{mi}^2 - \frac{1}{2} C_y A_c \rho \dot{y} |\dot{y}| \quad (3.2)$$

$$m\ddot{z} = -(\cos \phi \cos \theta) \cdot C_a \rho A r^2 \cdot \sum_{i=1}^4 \Omega_{mi}^2 + mg - \frac{1}{2} C_z A_c \rho \dot{z} |\dot{z}| \quad (3.3)$$

Batasan selanjutnya adalah batasan mengenai batasan *setpoint* sudut *roll* dan *pitch* yang diberikan. Agar persamaan dapat direduksi lagi, *setpoint* dibatasi pada sudut

kecil (di bawah  $10^\circ$ ) sehingga penyederhanaan deret Taylor dapat digunakan. Nilai sinus dan cosinus dinyatakan dalam deret Taylor dapat dinyatakan dalam persamaan:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots \quad (3.4)$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots \quad (3.5)$$

Jika sudut cukup kecil, maka orde kedua dan seterusnya dapat diabaikan, sehingga nilai sinus dan cosinus dapat dinyatakan sebagai:

$$\sin(x) = x \quad (3.6)$$

$$\cos(x) = 1 \quad (3.7)$$

Dengan persamaan tersebut, persamaan percepatan dapat disederhanakan lagi menjadi:

$$m\ddot{x} = -\theta \cdot C_a \rho A r^2 \cdot \sum_{i=1}^4 \Omega_{mi}^2 - \frac{1}{2} C_x A_c \rho \dot{x} |\dot{x}| \quad (3.8)$$

$$m\ddot{y} = \phi \cdot C_a \rho A r^2 \cdot \sum_{i=1}^4 \Omega_{mi}^2 - \frac{1}{2} C_y A_c \rho \dot{y} |\dot{y}| \quad (3.9)$$

$$m\ddot{z} = -C_a \rho A r^2 \cdot \sum_{i=1}^4 \Omega_{mi}^2 + mg - \frac{1}{2} C_z A_c \rho \dot{z} |\dot{z}| \quad (3.10)$$

Dari persamaan hasil reduksi di atas, dapat disimpulkan bahwa interaksi antar lop kontrol akhirnya minim. Interaksi yang disebabkan oleh sudut sudah tidak ada, menyisakan interaksi yang terjadi akibat pengaturan kecepatan tiap motor. Dari hasil perhitungan ini, didapatkan kesimpulan pengontrol berbasis sudut dapat digunakan untuk pengontrolan posisi dalam sistem pendaratan.

Perlu dijadikan catatan bahwa aksi kontrol yang bisa diberikan harus sesuai dengan aturan parameter yang dikirimkan pada *flight controller*. Oleh karena itu, dibuat normalisasi untuk variabel *thrust*. Dari dokumentasi pustaka Dronekit, dikatakan bahwa *thrust* 0,5 digunakan untuk menjaga posisi vertikal tetap diam, sehingga untuk mengontrol variabel *thrust* ini, setiap variabel kontrol ditambahkan dengan 0,5.

### 3.2.3 Berbagai Pembatasan dan Pengaturan Parameter

Dalam proses pendaratan, terdapat berbagai kemungkinan yang berpotensi membahayakan/merusak performa seperti:

1. Quadrotor sudah mencapai tanah, akan tetapi posisi lateral belum mencapai batas yang diinginkan sehingga ada kemungkinan *setpoint* pengontrol tetap diberikan sehingga aksi kontrol tetap dapat terjadi.
2. Quadrotor bergerak terlalu cepat/terlalu jauh sehingga FOV dari kamera tidak dapat mendeteksi keberadaan marka lagi. Hal ini membuat pembacaan posisi menjadi berhenti dan keluaran pengontrol menjadi bermasalah.

Untuk mengantisipasi permasalahan ini, dilakukan berbagai hal sebagai berikut:

1. Pendaratan dialihkan ke mode pendaratan bawaan apabila posisi sudah mencapai nilai tertentu. Hal ini akan memutus komunikasi antara Raspberry Pi dan Pixhawk sehingga quadrotor hanya akan bergerak turun. Nilai toleransi yang digunakan adalah  $\pm 30\text{ cm}$  untuk sumbu x dan y, serta  $110\text{ cm}$  untuk sumbu z.
2. *Setpoint* dari sumbu z tidak di set di ketinggian  $0\text{ cm}$ , akan tetapi dibuat di  $30\text{ cm}$  sehingga apabila sumbu x dan y belum mencapai batas toleransi, quadrotor akan tetap melayang sembari mengoreksi posisi pada sumbu x dan y.
3. Pemilihan pengontrol untuk kedua mode pengontrolan dibedakan. Pengontrol dengan sudut menggunakan pengontrol PD agar aksi kontrol dapat diperlambat ketika *setpoint* sudah dekat dengan nilai aktual. Hal ini dilakukan karena perbedaan orde sebanyak 2 antara percepatan dan posisi, sehingga butuh adanya penambahan *zero*/komponen derivatif untuk mengompensasi aksi kontrol. Untuk pengontrol dengan kecepatan, pengontrol yang diberikan cukup P karena ordenya hanya berbeda 1. Pengontrol I sangat dihindari penggunaannya karena diasumsikan tidak ada *steady state error* serta menghindari efek dari *integral windup*.
4. Aksi kontrol dibatasi pada nilai tertentu sehingga walaupun *error* cukup besar, tidak terjadi lonjakan yang tidak aman pada aksi kontrol. Nilai dari

pembatasan ini akan dilakukan dengan penalaan pada saat pengujian berlangsung.

5. Ketika pengukuran posisi berhenti, pembacaan tidak serta merta dijadikan nol, akan tetapi akan menyimpan data dari pengukuran terakhir. Harapannya agar walaupun marka tidak terdeteksi lagi, quadrotor dapat memiliki kecenderungan bergerak ke arah marka dan dapat memperbarui hasil pengukuran kembali.

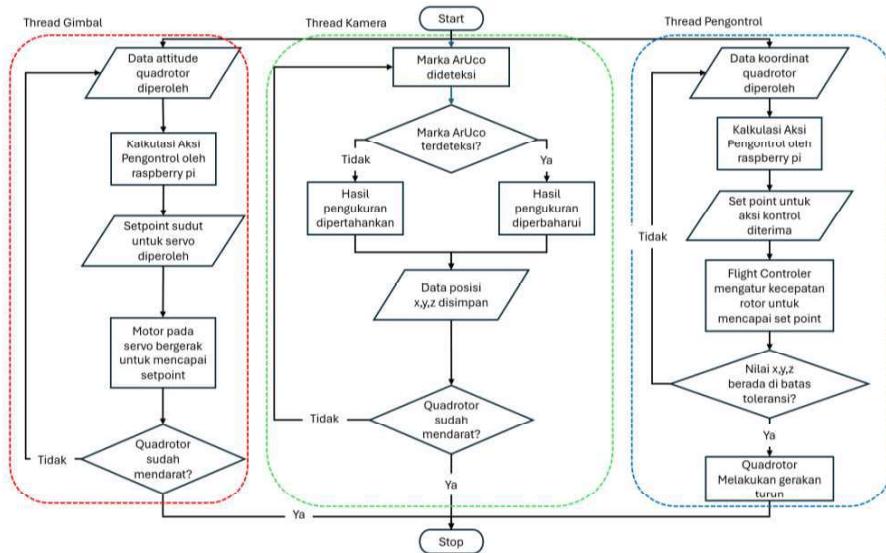
### 3.3 Integrasi Sistem Secara Keseluruhan

Setelah perancangan kedua sistem berhasil dilakukan, tahapan selanjutnya adalah menggabungkan semuanya menjadi 1 kesatuan. Integrasi dibagi menjadi 2 bagian, yaitu integrasi fisik yakni pemasangan semua komponen pada quadrotor, dan integrasi program yaitu pembuatan kode sumber gabungan yang dapat menjalankan keseluruhan sistem secara sekaligus.



**Gambar 3.14** Integrasi Seluruh Perangkat dan Rangkaian

Integrasi fisik dilakukan dengan memasang komponen Raspberry Pi beserta PCB, gimbal dan kamera, dan video *transmitter* serta menyambungkan semuanya menggunakan kabel. Hasil akhir dari penggabungannya dapat dilihat pada Gambar 3.14. Proses penggabungan ini haruslah memperhatikan posisi dari tiap komponen sehingga tidak mempengaruhi (misal: memberikan interferensi) komponen lain seperti modul GPS atau kompas.



**Gambar 3.15** Diagram Alir Keseluruhan Program dengan 3 *Thread* Berbeda

Untuk integrasi program, dilakukan pembagian program menjadi 3 *thread* berbeda sehingga program dapat berjalan secara independen tanpa harus menunggu proses eksekusi dari program lain selesai. Tampilan diagram alir dari program yang dibuat adalah seperti yang dapat dilihat pada Gambar 3.15. Ketika program dimulai, ketiga *thread* akan dimulai secara bersamaan dan memiliki lop proses sendiri. Walaupun berjalan secara sendiri-sendiri, variabel hasil dari tiap lop tetap akan disimpan secara global sehingga data akan dapat diakses oleh setiap *thread* yang lain. Program dikatakan selesai apabila quadrotor sudah mendarat secara sempurna.

## BAB IV

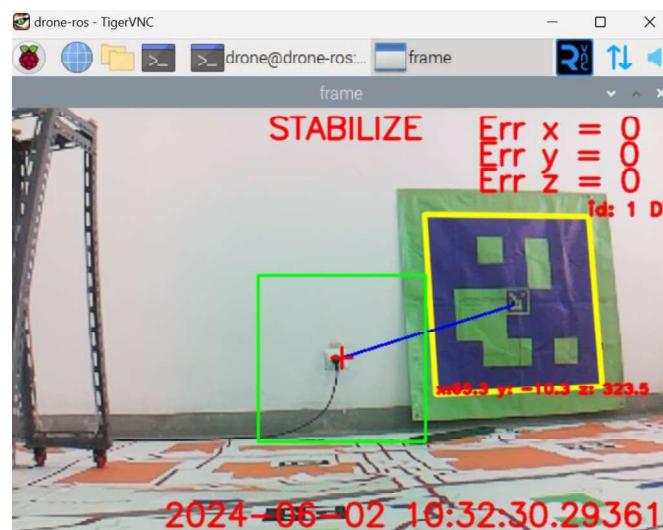
### HASIL DAN ANALISIS

#### 4.1 Performansi Sistem Instrumentasi

Pada bagian ini, terdapat dua hal yang akan diujikan, yaitu sistem pengukuran koordinat dan sistem stabilisasi dengan menggunakan gimbal.

##### 4.1.1 Sistem Pengukuran Koordinat Marka

Untuk sistem pengukuran koordinat, akurasi bukanlah tujuan utama dari sistem ini. Tujuan yang ingin dicapai adalah sistem pengukuran yang dapat mengestimasi posisi, sehingga keberhasilan pengukuran cukup dicapai ketika jarak estimasi yang dikeluarkan oleh program berkorelasi dengan jarak relatif yang aktual.



**Gambar 4.1** Tampilan Estimasi Jarak

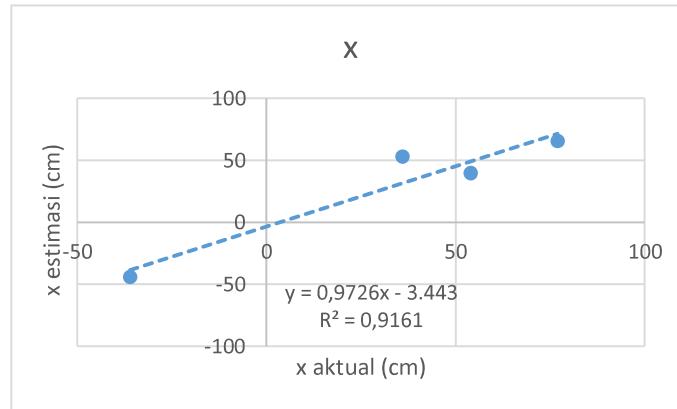
Pengujian dilakukan dengan mencoba mendeteksi marka ArUco dalam suatu ruangan. Mengukur koordinat marka ArUco dalam ruang 3 dimensi cukup sulit dikarenakan harus memastikan kesejajaran sumbu. Oleh karena itu, metode yang digunakan untuk mendapatkan koordinat adalah mencoba menandai sebuah titik yang sejajar dengan bidang ArUco, kemudian mengarahkan pusat kamera ke titik tersebut seperti yang dapat dilihat pada Gambar 4.1. Posisi x dan y dari marka ArUco didapat dengan mengukur jarak dari pusat marka ke titik yang

direferensikan, sedangkan jarak z didapat dengan mengukur jarak kamera ke bidang ArUco.

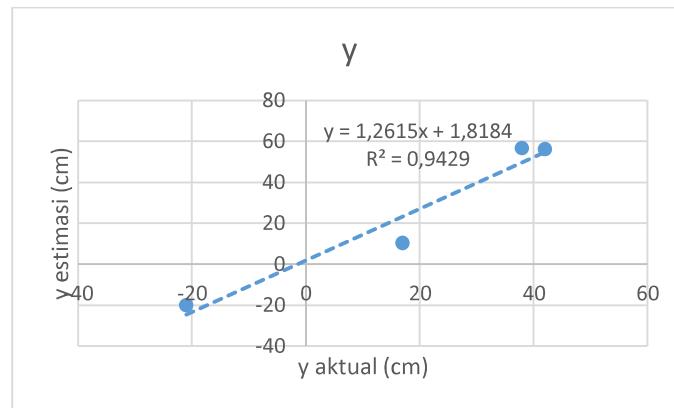
**Tabel 4.1** Perbandingan Jarak Aktual dan Hasil Estimasi Dalam cm

Jarak relatif aktual			Jarak estimasi			Error		
x(cm)	y(cm)	z(cm)	x(cm)	y(cm)	z(cm)	x(cm)	y(cm)	z(cm)
36	-21	327	52,98	-19,97	328,69	16,98	1,03	1,69
77	17	314	65,3	10,3	323,5	11,7	6,7	9,5
54	42	390	39,67	56,12	388,18	14,33	14,12	1,82
-36	38	363	-44,31	56,7	360,57	8,31	18,7	2,43

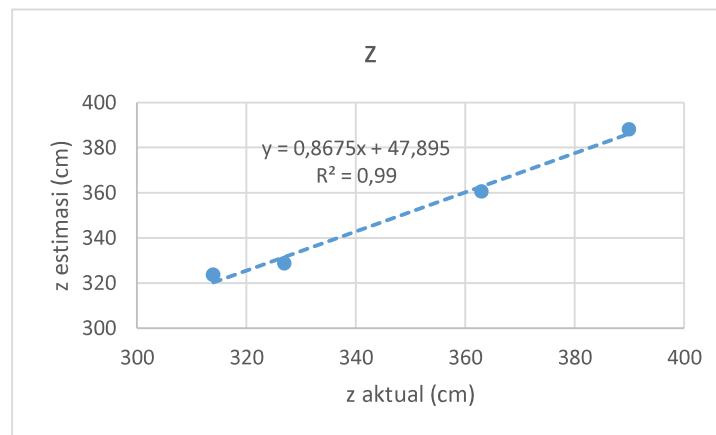
Percobaan dilakukan sebanyak empat kali dengan variasi koordinat di sumbu x, y, z. Hasil pengukuran dapat dilihat pada Tabel 4.1. Dari tabel tersebut, terlihat bahwa memang akurasi sistem pengukuran tidak begitu akurat, dengan *error* di beberapa variasi mencapai di atas 10 cm. Meskipun tidak akurat, estimasi dari marka ArUco masih mengikuti tren jarak aktual dengan nilai  $R^2$  masih cukup tinggi yang artinya hasil estimasi memang memiliki korelasi dengan pengukuran aktual. Tren ini dapat dilihat lebih baik pada grafik pada Gambar 4.2 hingga Gambar 4.4.



**Gambar 4.2** Tren Hasil Estimasi dan Pengukuran Aktual Koordinat x



**Gambar 4.3** Tren Hasil Estimasi dan Pengukuran Aktual Koordinat y



**Gambar 4.4** Tren Hasil Estimasi dan Pengukuran Aktual Koordinat z

Pengujian kemudian dilanjutkan dengan uji deteksi saat terbang. Terdapat permasalahan yang cukup mengganggu yaitu efek pantulan dari cahaya matahari dikarenakan marka dicetak menggunakan bahan spanduk yang cukup reflektif. Seperti yang dapat dilihat pada Gambar 4.5, secara kasat mata sebenarnya marka terlihat, akan tetapi dikarenakan efek tersebut, bagian yang harusnya terbaca gelap oleh kamera jadi terbaca terang. Masalah ini membuat pengujian hanya dapat dilakukan di pagi hari atau sore hari ketika matahari tidak terlalu terik.



**Gambar 4.5** Pendektsian yang Gagal Karena Pantulan

Untuk mengatasi permasalahan tersebut, marka dicetak kembali menggunakan bahan yang lebih tidak reflektif serta bagian berwarna ungu pada marka ditutupi dengan kertas karton berwarna hitam untuk menghilangkan pantulan seperti yang dapat dilihat pada Gambar 4.6. Hasilnya adalah pantulan berkurang drastis dan pendektsian dapat dilakukan walaupun di siang hari seperti yang dapat dilihat pada Gambar 4.7.



**Gambar 4.6** Marka ArUco Modifikasi



**Gambar 4.7** Deteksi yang Berhasil di Siang Hari

#### 4.1.2 Sistem Gimbal

Performa dari gimbal pada pengujian ini diukur secara kualitatif, yaitu dengan cara membandingkan perbedaan antara pengukuran dengan menggunakan gimbal dengan yang tidak menggunakan gimbal. Oleh karena itu, sistem servo akan diputuskan terlebih dahulu dari rangkaian sehingga orientasi dari gimbal akan selalu tetap. Quadrotor kemudian diterbangkan secara manual dan kemudian diambil kesimpulannya.

Pada Gambar 4.8, ditunjukkan citra yang diambil pada momen yang berurutan saat quadrotor sedang terbang dengan kondisi gimbal tidak menyala. Pada awalnya, quadrotor berada dalam kondisi mempertahankan posisi dengan keadaan datar. Quadrotor kemudian digerakkan ke samping yang otomatis mengubah sudut dari quadrotor. Pada urutan gambar tersebut, marka ArUco yang awalnya terlihat di tengah hampir sejajar dengan titik tengah kamera secara cepat berpindah ke sisi kanan hingga tidak terdeteksi lagi. Hal ini tentunya membuat pengukuran jadi tidak bisa diandalkan sehingga sistem kontrol tidak lagi bisa bekerja dengan baik.



**Gambar 4.8** Kondisi Citra Pendaratan Sebelum (a) dan Sesudah Pergerakan (b)

Dengan penggunaan gimbal, kejadian tersebut tidak terjadi lagi. Pergerakan marka ArUco pada kamera hanya terjadi ketika terjadi perpindahan posisi dari quadrotor. Hal ini dapat dibuktikan dengan adanya hasil pengukuran posisi yang cukup kontinu/tidak ada fluktuasi, serta jika dilihat dari video citra lokasi pendaratan, tidak terdapat guncangan lagi yang berarti. Keberhasilan pada sistem ini membuat pengujian selanjutnya yaitu pengujian performansi pengontrolan dapat dilakukan.

## 4.2 Performansi Pengontrolan

Untuk pengujian pengontrolan, uji coba dilakukan dengan 2 mode pada *flight controller* yaitu mode GUIDED dan mode GUIDED\_NOGPS sehingga pengontrolan dengan dukungan GPS maupun tidak dapat dilakukan serta dianalisis performanya. Parameter yang dianalisis adalah bagaimana lintasan (dari awal hingga akhir) pendaratan dari kedua pengontrol serta hubungan antara output dan input pada sistem pengontrol. Cara pengujian dilakukan adalah dengan membawa quadrotor ke ketinggian tertentu sampai marka ArUco terdeteksi, kemudian mode pada *flight controller* diganti sehingga proses pendaratan dapat langsung berjalan secara otomatis.

### 4.2.1 Pengontrol Dengan Dukungan GPS

Agar proses pendaratan berhasil dilakukan, dilakukan penalaan parameter dengan metode *trial* dan *error*. Penalaan dilakukan dengan cara mengubah parameter P pada pengontrol hingga respons pengontrol dirasa cukup baik (tidak terlalu cepat dan tidak terlalu lambat). Parameter yang didapatkan setelah penalaan dapat dilihat pada Tabel 4.2

**Tabel 4.2** Parameter Pengontrol Dengan Kecepatan

Parameter	P	Constraint
$v_x$	0,004	[-1;1]
$v_y$	0,004	[-1;1]
$v_z$	0,001	[-0,3;0,3]

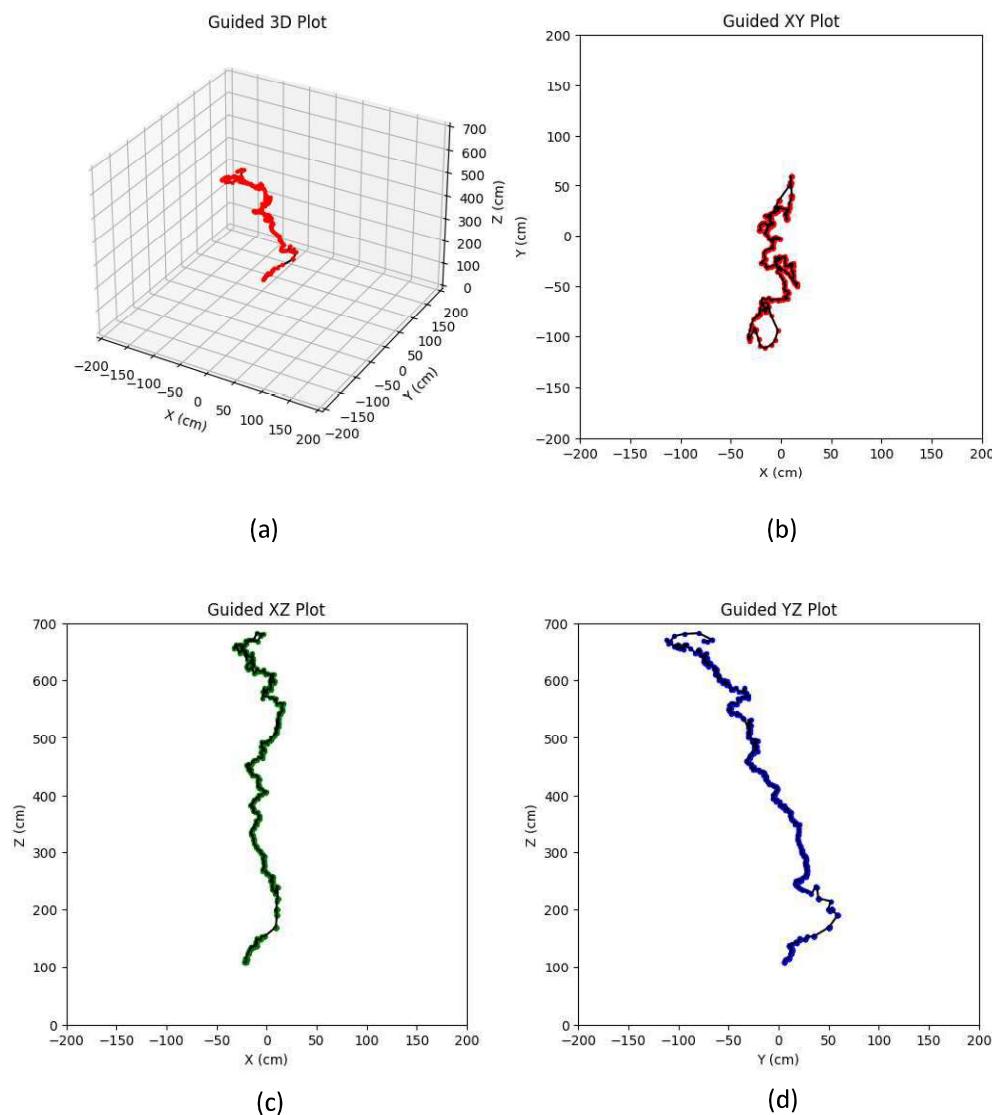
Metode yang dilakukan untuk menilai performansi dari pendaratan adalah dengan melihat grafik posisi selama proses pendaratan serta nilai *Mean Absolute Error* (MAE). Lintasan dari proses pendaratan ini dapat dilihat pada Gambar 4.9 serta radius posisi quadrotor terhadap pusat marka dapat dilihat pada Gambar 4.10. Nilai MAE dari posisi quadrotor selama proses pendaratan dapat dilihat pada Tabel 4.3.

Dari grafik posisi yang ditampilkan, algoritma yang dirancang juga mampu untuk membawa quadrotor ke pusat marka dari ketinggian 700 cm dan radius 100 cm ke 22 cm serta dapat mendarat di atas marka ArUco, sehingga algoritma ini dapat dikatakan memenuhi kriteria yang disebutkan sebelumnya yaitu di bawah 30 cm untuk koordinat x dan y. Nilai (MAE) dari proses pendaratan ini terlihat cukup

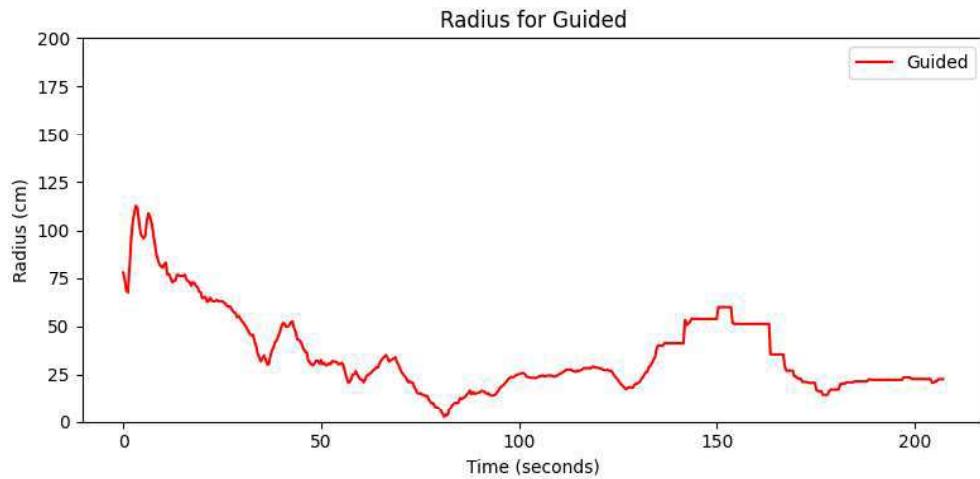
kecil, sehingga dapat dikatakan bahwa simpangan posisi quadrotor juga cukup kecil. Kurva posisi pendaratan juga terlihat stabil dengan sedikit osilasi.

**Tabel 4.3** MAE Posisi Quadrotor Pendaratan Dengan GPS

Koordinat	x	y	r
MAE (cm)	10,8672	32,6929	35,9229



**Gambar 4.9** Lintasan Pendaratan Dengan GPS Dalam 3 Dimensi (a) dan 2 Dimensi untuk Sumbu XY (b), Sumbu XZ (c) dan YZ (d)



**Gambar 4.10** Plot Radius Terhadap Waktu Pendaratan Dengan GPS

#### 4.2.2 Pengontrol Tanpa Dukungan GPS

Sebagaimana yang telah dilakukan pada percobaan sebelumnya dengan dukungan GPS, perlu dilakukan penalaan parameter pengontrolan agar proses pendaratan berhasil. Setelah sekian kali percobaan, didapatkan parameter pengontrol sebagai yang ditampilkan pada Tabel 4.4. Sebagai catatan, parameter di

**Tabel 4.4** Parameter Pengontrol Dengan Sudut

Parameter	P	D	Constraint
Roll	0,05	0,00004	[-10;10]
Pitch	0,05	0,00004	[-10;10]
Thrust	0,0005	-	[-0,45;0,55]

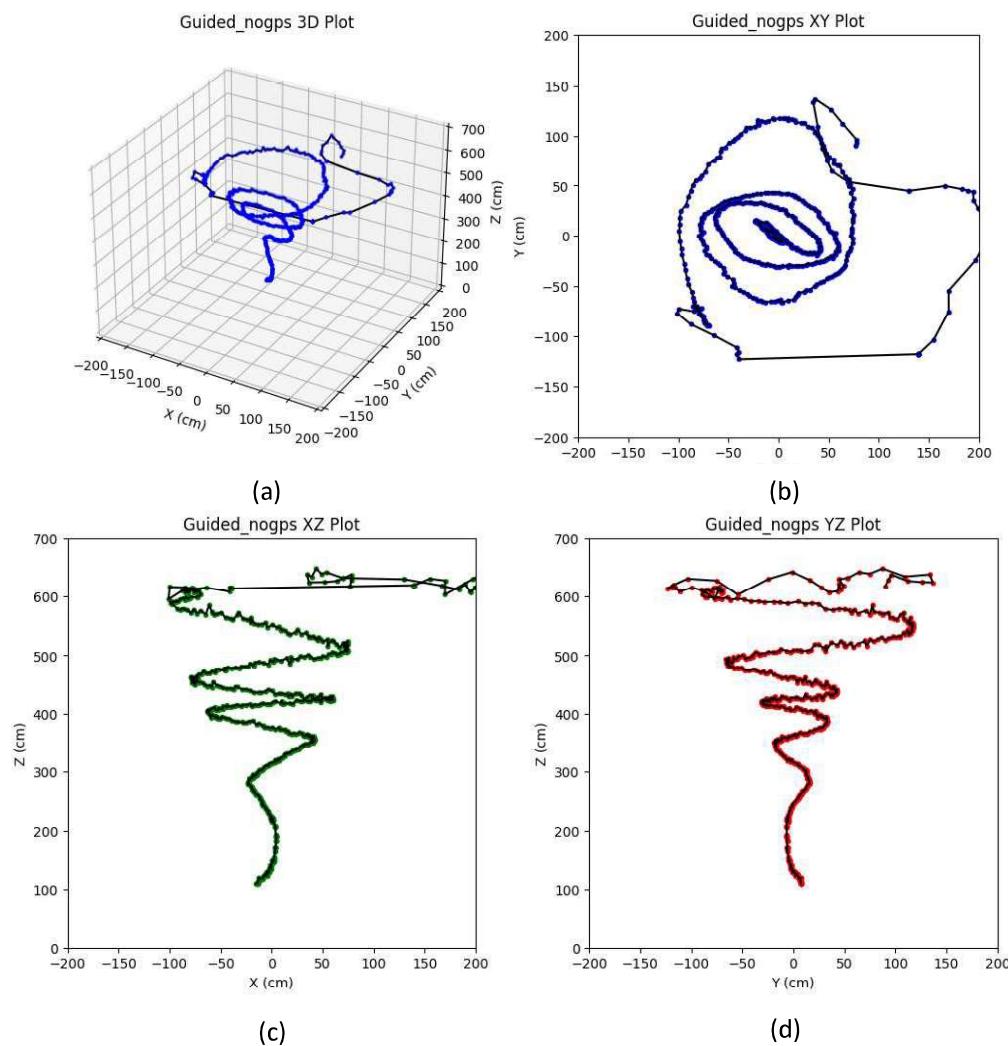
Lintasan dari proses pendaratan ini dapat dilihat pada Gambar 4.11 (abaikan lintasan pada bagian awal dikarenakan terdapat penyesuaian di bagian sudut *yaw*) dan Gambar 4.12 Plot Radius Terhadap Waktu Pendaratan Tanpa GPS untuk plot radius selama penerbangan. Dari percobaan juga, didapatkan nilai (MAE) posisi quadrotor terhadap pusat marka seperti yang dapat dilihat pada Tabel 4.5.

Dari grafik lintasan, terlihat bahwa algoritma pendaratan mampu untuk membawa quadrotor dari posisi awal 600 cm untuk ketinggian dan radius 200 cm dari pusat marka menjadi radius 17 cm dari pusat marka dan berhasil mendarat di permukaan marka, sehingga dapat dikatakan algoritma pendaratan ini juga dapat mencapai kriteria yang disebutkan pada bagian. Kekurangan dari pengontrolan dengan skema

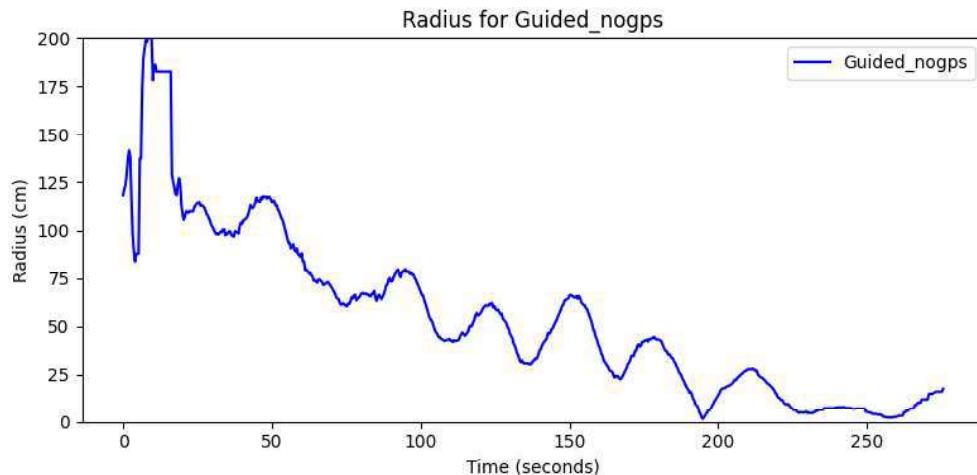
ini adalah osilasi yang terjadi pada posisi quadrotor cukup besar. Hal tersebut dapat dilihat pada grafik lintasan dan juga nilai MAE yang lebih besar jika dibandingkan dengan pendaratan dengan dukungan GPS.

**Tabel 4.5** MAE Posisi Quadrotor Pendaratan Tanpa GPS

Koordinat	x	y	r
MAE (cm)	38,6422	31,4564	54,5046



**Gambar 4.11** Lintasan Pendaratan Tanpa GPS Dalam 3 Dimensi (a) dan 2 Dimensi untuk Sumbu XY (b), Sumbu XZ (c) dan YZ (d)



**Gambar 4.12** Plot Radius Terhadap Waktu Pendaratan Tanpa GPS

#### 4.2.3 Improvisasi Pengontrol Tanpa Dukungan GPS

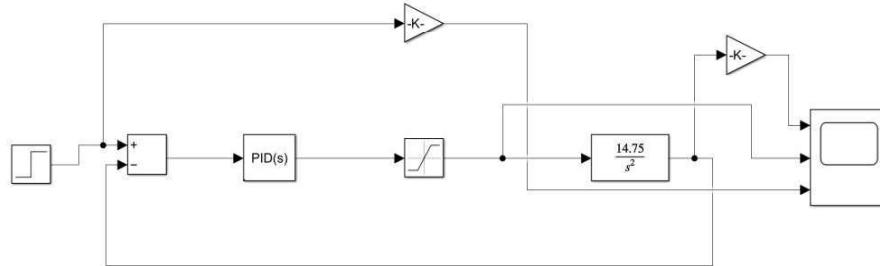
Untuk meningkatkan performa dari pengontrolan tanpa GPS, dilakukan pemodelan pada sistem quadrotor. Pemodelan dilakukan dengan menggunakan data terbang lurus pada sumbu x dan y sehingga hubungan antara posisi dan sudut roll atau pitch dapat diketahui. Pemodelan dilakukan dengan pemodelan secara *gray box* dengan memanfaatkan persamaan (3.8) dan (3.9) yang berada di dalam domain Laplace. Dengan menganggap koefisien gaya gesek cukup kecil dan menyatukan seluruh parameter yang konstan dengan parameter baru yaitu  $k$ , maka diperoleh fungsi transfer sistem :

$$\frac{X}{\Theta} = -\frac{k_1}{s^2} \quad (4.1)$$

$$\frac{Y}{\Phi} = \frac{k_2}{s^2} \quad (4.2)$$

Persamaan tersebut dimasukkan ke dalam algoritma *curve fitting* pada aplikasi Matlab untuk menemukan nilai  $k_1$  dan  $k_2$ . Setelah algoritma tersebut dijalankan, diperoleh nilai  $k_1 = 14,75$  dan  $k_2 = 13,108$ . Nilai tersebut kemudian digunakan untuk simulasi pengontrol dengan menggunakan Simulink dengan blok program yang dapat dilihat pada Gambar 4.13. Pada program tersebut, dilakukan penalaan secara visual terhadap grafik respons sistem sehingga pada akhirnya diperoleh parameter yang mengakibatkan respons yang tidak terlalu berosilasi namun tetap

dapat mencapai setpoint secara cepat. Pemodelan dan penalaan dilakukan hanya dalam 1 sumbu mengingat perbedaan nilai parameter  $k$  tidak terlalu signifikan.

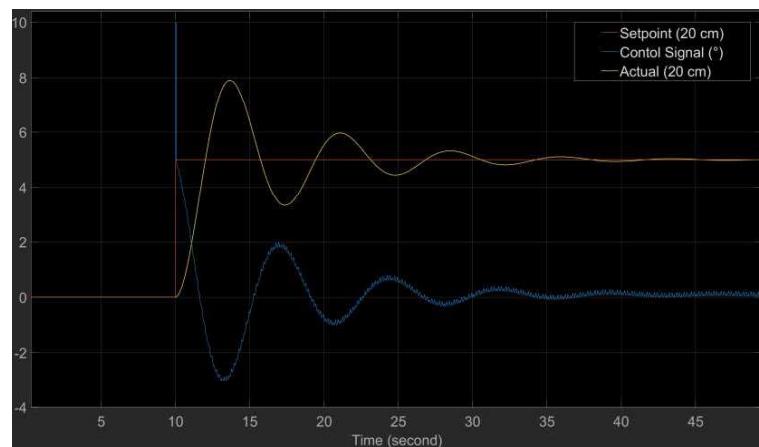


**Gambar 4.13** Diagram Blok Simulasi Menggunakan Simulink

Setelah melakukan percobaan, didapatkan parameter penalaan sebagaimana yang diperoleh pada Tabel 4.6. Simulasi dengan parameter tersebut menghasilkan grafik respons sebagaimana yang dapat dilihat pada Gambar 4.14. Terlihat bahwa meskipun terjadi osilasi, osilasi lebih cepat teredam dan juga cukup cepat untuk mencapai *setpoint*. Parameter ini kemudian diujikan pada pendaratan secara langsung untuk melihat peningkatan performansinya.

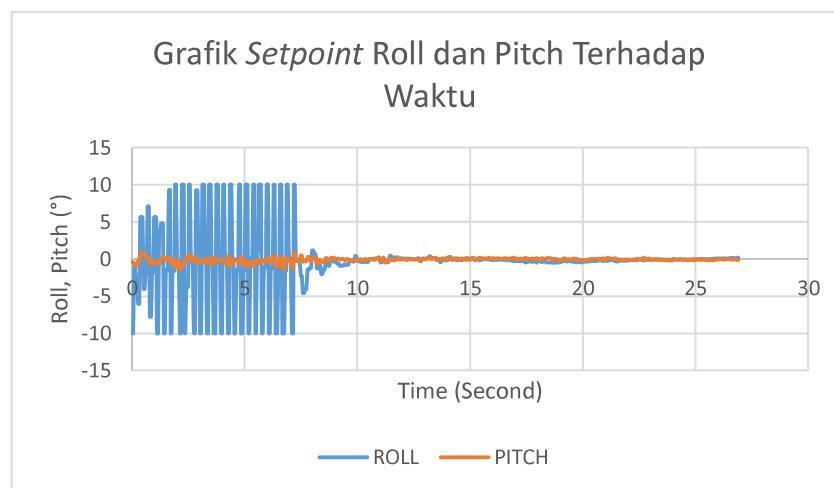
**Tabel 4.6** Parameter PID Hasil Penalaan Visual

Parameter	Value
Proportional	0,05
Integral	0
Derivative	0,02

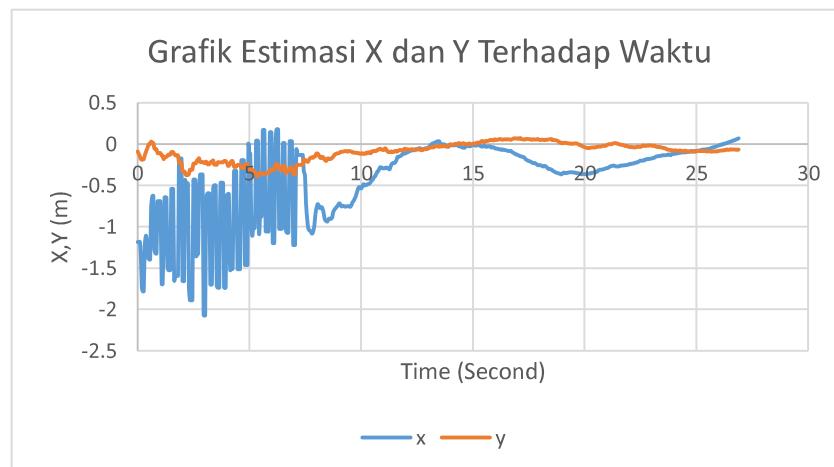


**Gambar 4.14** Respons Sistem Pada Simulasi

Pada pengujian aktual, terdapat permasalahan yaitu aksi kontrol derivatif yang membuat quadrotor tidak stabil. Hal ini dapat dilihat pada grafik aksi kontrol yang terdapat pada Gambar 4.15. Aksi kontrol yang fluktuatif secara cepat ini juga menyebabkan pembacaan dari kamera mengalami fluktuasi juga seperti yang dapat dilihat pada Gambar 4.16 yang terjadi karena respons gimbal yang kurang cepat untuk melakukan kompensasi pada perubahan sudut. Meskipun terjadi fluktuasi yang cukup parah pada aksi kontrol, hal tersebut hanya terjadi pada awal proses pendaratan (sekitar 8 detik pertama). Setelah 8 detik, sinyal kontrol dan hasil pembacaan posisi mulai stabil dan pendaratan dilanjutkan tanpa adanya osilasi seperti yang terlihat pada bagian sebelumnya.



**Gambar 4.15** Grafik Setpoint Roll dan Pitch Terhadap Waktu



**Gambar 4.16** Grafik Pengukuran X dan Y Terhadap Waktu

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Setelah melakukan perancangan dan pengujian dari sistem yang sudah dibuat. Didapatkan kesimpulan akhir sebagai berikut:

1. Sistem instrumentasi berbasis citra telah berhasil dibuat serta telah berhasil untuk mengestimasi posisi dari marka ArUco dengan hasil estimasi cukup berkorelasi dengan posisi aktual.
2. Sistem stabilisasi dari kamera telah berhasil membuat kamera memiliki orientasi yang tetap selama penerbangan walaupun terdapat gangguan berupa perubahan orientasi quadrotor.
3. Algoritma pendaratan yang dirancang mampu mendaratkan quadrotor, baik di skenario tanpa GPS yaitu dengan mengontrol sudut quadrotor, atau dengan GPS yaitu dengan mengontrol kecepatan quadrotor. Nilai MAE dari posisi quadrotor (x,y dan radius dalam cm) selama proses pendaratan adalah (38.6422, 31.4564, 54.5046) untuk pendaratan tanpa GPS dan (10.8672, 32.6929, 35.9229) untuk pendaratan dengan GPS.
4. Peningkatan performansi dari pendaratan tanpa GPS telah berhasil dilakukan dengan cara pemodelan dan penalaan secara simulasi yang menghasilkan gerakan pendaratan yang tidak berosilasi. Meskipun demikian, terdapat fluktuasi pada aksi kontrol yang menyebabkan ketidakstabilan dan fluktuasi pada hasil estimasi posisi.

#### **5.2 Saran**

Adapun saran untuk penelitian ini, telah penulis rangkum agar penelitian selanjutnya dapat dilakukan lebih baik. Saran tersebut antara lain:

1. Terdapat 2 opsi yang lebih baik untuk menghilangkan efek pantulan pada marka yaitu dengan *manual thresholding* atau dengan pemilihan bahan yang lebih baik.

2. Diperlukan peningkatan performansi dari sistem stabilisasi kamera agar estimasi posisi dapat dilakukan dengan lebih baik. Hal ini dapat dilakukan dengan meningkatkan frekuensi kontrol dari gimbal baik dengan mengubah konfigurasi pengambilan data dari *flight controller* atau dapat juga menggunakan IMU eksternal dengan frekuensi yang lebih tinggi.
3. Pendaratan tanpa bantuan GPS dapat menggunakan diagram blok yang sama dengan pendaratan dengan bantuan GPS, dengan estimasi kecepatan yang sebelumnya menggunakan GPS diganti dengan menggunakan *observer* dari hasil pembacaan posisi.

## DAFTAR PUSTAKA

- [1] A. Paris, B. T. Lopez, and J. P. How, “Dynamic Landing of an Autonomous Quadrotor on a Moving Platform in Turbulent Wind Conditions,” Sep. 2019, [Online]. Available: <http://arxiv.org/abs/1909.11071>
- [2] G. Gillini and F. Arrichiello, “Nonlinear model predictive control for the landing of a quadrotor on a marine surface vehicle,” in *IFAC-PapersOnLine*, Elsevier B.V., 2020, pp. 9328–9333. doi: 10.1016/j.ifacol.2020.12.2388.
- [3] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza, “Vision-based autonomous quadrotor landing on a moving platform,” in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, IEEE, Oct. 2017, pp. 200–207. doi: 10.1109/SSRR.2017.8088164.
- [4] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos, “Quadrotor landing on an inclined platform of a moving ground vehicle,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2015, pp. 2202–2207. doi: 10.1109/ICRA.2015.7139490.
- [5] T. Yang *et al.*, “Hybrid Camera Array-Based UAV Auto-Landing on Moving UGV in GPS-Denied Environment,” *Remote Sens (Basel)*, vol. 10, no. 11, p. 1829, Nov. 2018, doi: 10.3390/rs10111829.
- [6] M. F. Sani, M. Shoaran, and G. Karimian, “Automatic landing of a low-cost quadrotor using monocular vision and Kalman filter in GPS-denied environments,” *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 27, no. 3, pp. 1821–1838, 2019, doi: 10.3906/elk-1809-204.
- [7] P. Wang, C. Wang, J. Wang, and M. Q. H. Meng, “Quadrotor Autonomous Landing on Moving Platform,” in *Procedia Computer Science*, Elsevier B.V., 2022, pp. 50–57. doi: 10.1016/j.procs.2022.10.097.
- [8] X. Liu, S. Zhang, J. Tian, and L. Liu, “An Onboard Vision-Based System for Autonomous Landing of a Low-Cost Quadrotor on a Novel Landing Pad,” *Sensors*, vol. 19, no. 21, p. 4703, Oct. 2019, doi: 10.3390/s19214703.
- [9] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognit*, vol. 47, no. 6, pp. 2280–2292, Jun. 2014, doi: 10.1016/j.patcog.2014.01.005.
- [10] T. Tocci, L. Capponi, and G. Rossi, “ArUco marker-based displacement measurement technique: Uncertainty analysis,” *Engineering Research Express*, vol. 3, no. 3, Sep. 2021, doi: 10.1088/2631-8695/ac1fc7.

- [11] M. Gašparović and L. Jurjević, “Gimbal influence on the stability of exterior orientation parameters of UAV acquired images,” *Sensors (Switzerland)*, vol. 17, no. 2, Feb. 2017, doi: 10.3390/s17020401.
- [12] C. E. Lin and S.-K. Yang, “Camera gimbal tracking from UAV flight control,” in *2014 CACS International Automatic Control Conference (CACS 2014)*, IEEE, Nov. 2014, pp. 319–322. doi: 10.1109/CACS.2014.7097209.
- [13] Y. Naidoo, R. Stopforth, and G. Bright, “Quad-Rotor Unmanned Aerial Vehicle Helicopter Modelling & Control Regular Paper.” [Online]. Available: [www.intechopen.com](http://www.intechopen.com)
- [14] L. Hua, J. Zhang, D. Li, and X. Xi, “Fault-Tolerant Active Disturbance Rejection Control of Plant Protection of Unmanned Aerial Vehicles Based on a Spatio-Temporal RBF Neural Network,” *Applied Sciences*, vol. 11, no. 9, p. 4084, Apr. 2021, doi: 10.3390/app11094084.
- [15] E. Marchand, H. Uchiyama, and F. Spindler, “Pose Estimation for Augmented Reality: A Hands-On Survey,” *TO APPEAR*, vol. 22, no. 12, pp. 2633–2651, 2016, doi: 10.1109/TVCG.2015.2513408.
- [16] J. Salvi, X. Armanguà, and J. Batlle, “A comparative review of camera calibrating methods with accuracy evaluation,” 2002. [Online]. Available: [www.elsevier.com/locate/patcog](http://www.elsevier.com/locate/patcog)
- [17] J. Weng, P. Cohen, and M. Herniou, “Camera calibration with distortion models and accuracy evaluation,” *IEEE Trans Pattern Anal Mach Intell*, vol. 14, no. 10, pp. 965–980, 1992, doi: 10.1109/34.159901.
- [18] A. Nowakowski and W. Skarbek, “Analysis of Brown camera distortion model,” in *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2013*, R. S. Romaniuk, Ed., SPIE, Oct. 2013, p. 89030X. doi: 10.1117/12.2035447.
- [19] D. E. Seborg, T. F. Edgar, D. A. Mellichamp, and F. J. Doyle III, “Process Dynamics and Control Fourth Edition.”
- [20] Katsuhiko. Ogata, *Modern control engineering*. Prentice-Hall, 2010.
- [21] Nikunj Mehta, Dharmendra Chauhan, Sagar Patel, and Siddharth Mistry, “Design of HMI Based on PID Control of Temperature,” *International Journal of Engineering Research and*, vol. V6, no. 05, May 2017, doi: 10.17577/IJERTV6IS050074.
- [22] Y. Lee, S. Park, and M. Lee, “PID Controller Tuning to Obtain Desired Closed-Loop Responses for Cascade Control Systems,” *IFAC Proceedings Volumes*, vol. 31, no. 11, pp. 613–618, Jun. 1998, doi: 10.1016/S1474-6670(17)44994-9.

- [23] R. Isermann, “Cascade Control Systems,” in *Digital Control Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 294–301. doi: 10.1007/978-3-662-02319-8\_16.
- [24] C. Wai Zhao, J. Jegatheesan, and S. Chee Loon, “Exploring IOT Application Using Raspberry Pi,” *International Journal of Computer Networks and Applications*, vol. 2, no. 1, [Online]. Available: <http://www.digi.com>
- [25] R. I. S. Pereira, I. M. Dupont, P. C. M. Carvalho, and S. C. S. Jucá, “IoT embedded linux system based on Raspberry Pi applied to real-time cloud monitoring of a decentralized photovoltaic plant,” *Measurement (Lond)*, vol. 114, pp. 286–297, Jan. 2018, doi: 10.1016/j.measurement.2017.09.033.
- [26] M. Vanitha, M. Selvalakshmi, and R. Selvarasu, “Monitoring and controlling of mobile robot via internet through raspberry Pi board,” in *2016 Second International Conference on Science Technology Engineering and Management (ICONSTEM)*, IEEE, Mar. 2016, pp. 462–466. doi: 10.1109/ICONSTEM.2016.7560864.
- [27] “Buy a Raspberry Pi 4 Model B – Raspberry Pi.” Accessed: Jun. 03, 2024. [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>
- [28] “Raspberry Pi hardware - Raspberry Pi Documentation.” Accessed: Jun. 01, 2024. [Online]. Available: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>
- [29] H. Dipak Ghosh, L. Solanki, G. Sahu, and A. Professor, “A Review Paper on Raspberry Pi and its Applications,” *International Journal of Advances in Engineering and Management (IJAEM*, vol. 2, p. 225, 2008, doi: 10.35629/5252-0212225227.
- [30] “3DR Pixhawk 1 Flight Controller (Discontinued) | PX4 Guide (main).” Accessed: May 24, 2024. [Online]. Available: [https://docs.px4.io/main/en/flight\\_controller/pixhawk.html](https://docs.px4.io/main/en/flight_controller/pixhawk.html)
- [31] “Archived:PX4FMU Overview — Copter documentation.” Accessed: May 30, 2024. [Online]. Available: <https://ardupilot.org/copter/docs/common-px4fmu-overview.html>
- [32] “Copter Position Control and Navigation — Dev documentation.” Accessed: Jun. 03, 2024. [Online]. Available: <https://ardupilot.org/dev/docs/code-overview-copter-poscontrol-and-navigation.html>
- [33] “Extended Kalman Filter (EKF) — Copter documentation.” Accessed: Jun. 01, 2024. [Online]. Available: <https://ardupilot.org/copter/docs/common-apm-navigation-extended-kalman-filter-overview.html>

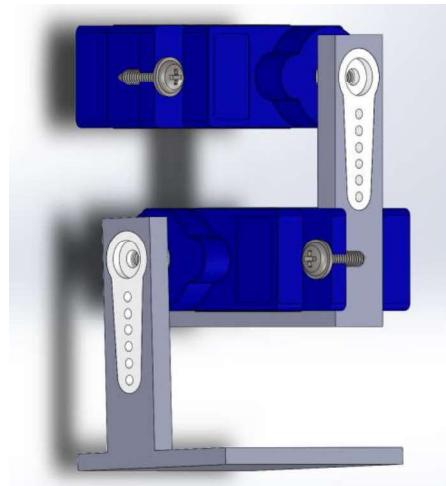
- [34] N. Pinckney, “Pulse-width modulation for microcontroller servo control,” *IEEE Potentials*, vol. 25, no. 1, pp. 27–29, Jan. 2006, doi: 10.1109/MP.2006.1635026.
- [35] “DYNAMIXEL Protocol 2.0.” Accessed: May 23, 2024. [Online]. Available: <https://emanual.robotis.com/docs/en/dxl/protocol2/>
- [36] J. M. Hilkert, “Inertially stabilized platform technology Concepts and principles,” *IEEE Control Syst*, vol. 28, no. 1, pp. 26–46, Feb. 2008, doi: 10.1109/MCS.2007.910256.
- [37] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [38] G. Bradski and A. Kaehler, *Learning OpenCV*, 1. ed. O’Reilly Media, Inc., 2008.
- [39] “About DroneKit.” Accessed: Jun. 01, 2024. [Online]. Available: <https://dronekit-python.readthedocs.io/en/latest/about/overview.html>
- [40] “Introduction · MAVLink Developer Guide.” Accessed: Jun. 01, 2024. [Online]. Available: <https://mavlink.io/en/>

## Lampiran A Improvisasi Desain Gimbal

Desain gimbal yang dibuat dalam tugas akhir ini beberapa kali mengalami perubahan. Hal tersebut dikarenakan performansi dari motor servo dan desain yang kurang memuaskan. Berikut dalam dilampirkan bagaimana perkembangan gimbal hingga menjadi seperti yang dipakai dalam tugas akhir ini.

### A.1 Gimbal Versi Pertama

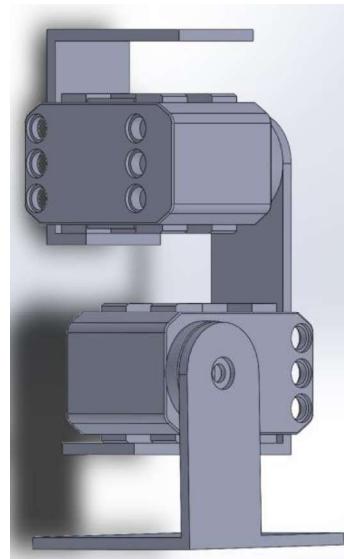
Gimbal pertama dibuat dengan menggunakan servo tower p90 yang dikontrol menggunakan sinyal PWM. Desain dari gimbal dapat dilihat pada Gambar A.1. Gimbal ini memiliki kelemahan di bagian motor servo yang hanya memiliki sudut  $180^\circ$  dan resolusi yang rendah. Oleh karena itu pada bagian berikutnya, servo diganti menjadi yang lebih baik yaitu servo tipe Dynamixel XL-320.



Gambar A.1 Gimbal Versi Pertama

### A.2 Gimbal Versi Kedua

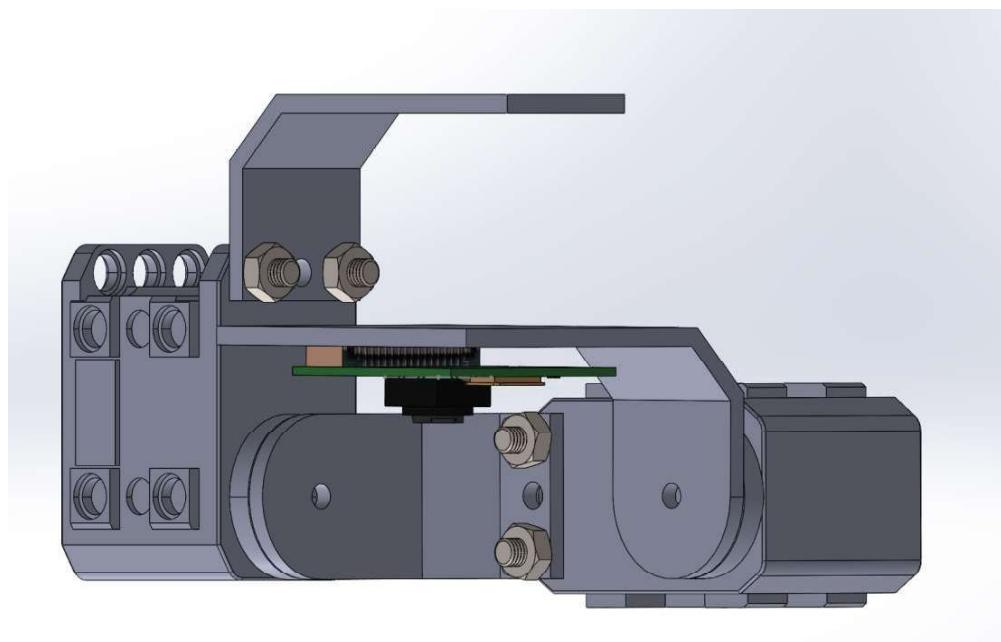
Desain gimbal setelah mengganti servo dapat dilihat pada Gambar A.2. Gimbal versi ini memiliki kemampuan yang lebih baik dalam merespons gangguan berupa perubahan sudut. Kekurangan yang dimilikinya adalah desain yang terlalu panjang yang menyebabkannya bisa dengan mudah menyentuh tanah serta punya defleksi yang lebih besar



**Gambar A.2** Gimbal Versi Kedua

### A.3 Gimbal Versi Ketiga

Setelah beberapa improvisasi pada desain, diperoleh bentuk gimbal seperti yang terlihat pada Gambar A.3. Desain sudah dijadikan lebih ringkas sehingga jarak ke bagian bawah tidak terlalu jauh. Hal ini membuat lebih banyak ruang kosong pada bagian bawah quadrotor untuk modifikasi.



**Gambar A.3** Gimbal Versi Ketiga

## Lampiran B Diagram Pengontrol *Attitude* Quadrotol

