TUGAS AKHIR – EE 184801

PENDARATAN PRESISI PADA *DRONE* BERBASIS KOMPUTER VISI

Petra Matatias Situmorang
NRP 07111540000119

Dosen Pembimbing
Astria Nur Irfansyah, S.T, M.Eng., Ph.D.
Muhammad Attamimi B.Eng., M.Eng., Ph.D.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – EE 184801

PENDARATAN PRESISI PADA *DRONE* BERBASIS KOMPUTER VISI

Petra Matatias Situmorang
NRP 07111540000119

Dosen Pembimbing
Astria Nur Irfansyah, S.T, M.Eng., Ph.D.
Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

Halaman ini sengaja dikosongkan

FINAL PROJECT – EE 184801

PRECISION LANDING FOR DRONE BASED ON COMPUTER VISION

Petra Matatias Situmorang
NRP 07111540000119

Supervisor
Astria Nur Irfansyah, S.T, M.Eng., Ph.D.
Muhammad Attamimi B.Eng., M.Eng., Ph.D.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical Technology
Sepuluh Nopember Institute of Technology
Surabaya 2019

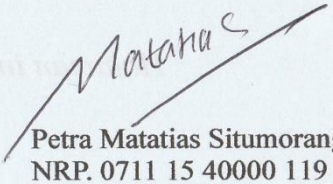
Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Pendaratan Presisi pada *Drone* Berbasis Komputer Visi" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya,



Petra Matatias Situmorang
NRP. 0711 15 40000 119

Halaman ini sengaja dikosongkan

PENDARATAN PRESISI PADA *DRONE* BERBASIS KOMPUTER VISI

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I,	Dosen Pembimbing II,
A. Nur Irfansyah, ST, M.Eng., Ph.D.	M. Attamimi, B.Eng., M.Eng., Ph.D.
NIP. 198103252010121002	NIP. 1985201711039

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO

SURABAYA
JULI, 2019

Halaman ini sengaja dikosongkan

PENDARATAN PRESISI PADA *DRONE* BERBASIS KOMPUTER VISI

Nama : Petra Matatias Situmorang
Pembimbing I : Astria Nur Irfansyah, ST, M.Eng., Ph.D.
Pembimbing II : Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

ABSTRAK

Unmanned Aerial Vehicle (UAV) atau *drone* adalah wahana yang tidak membutuhkan manusia untuk mengendarainya. Pesawat tampa awak sudah banyak digunakan untuk membantu pekerjaan manusia, contohnya mengantarkan paket (*delivery kit),* pemantauan dan pemetaan suatu area *(mapping and monitoring)* dan fotografi. Salah satu penelitian berlanjut adalah tentang pendaratan otomatis sebuah *drone* multi-copter. Metode navigasi yang berkembang saat ini yaitu sistem navigasi GPS dan sistem naviasi GPS/Visual yang terintegrasi.

Dalam tugas akhir ini, komputer visi akan diaplikasikan pada sebuah *drone quadcopter*. Kamera akan membantu wahana untuk menemukan landasan pendaratan. Sistem visual dapat memperoleh posisi relatif wahana terhadap target. Maka dapat dirancang kontrol untuk mencapai titik pendaratan. Stabilitas dan akurasi pendaratan adalah fokus dan kesulitan yang dihadapi. Dengan pendaratan yang presisi, *drone* dapat diaplikasikan dengan baik pada beberapa misi seperti *drone* pengantar paket yang membutuhkan akurasi pendaratan yang presisi.

Hasil pengujian yang didapatkan dalam tugas akhir ini adalah sistem pendaratan pada *drone* yang dapat mendarat dengan otomatis pada landasan yang ditentukan. Bantuan visual adalah komponen utama dalam menentukan posisi pendaratan *drone*. Drone berhasil mendarat dengan rata-rata error 45 cm dengan error minimum adalah 21cm.

Kata kunci: komputer visi, *drone*, pendaratan presisi

Halaman ini sengaja dikosongkan

PRECISION LANDING FOR DRONE BASED ON COMPUTER VISION

Name : Petra Matatias Situmorang
Supervisor : Astria Nur Irfansyah, ST, M.Eng., Ph.D.
Co-Supervisor : Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

ABSTRACT

Unmanned Aerial Vehicle (UAV) or a drone is a vehicle that does not require humans to drive it. A plane without a crew is already widely used to help human work, for example delivering kits, monitoring and mapping an area (mapping and monitoring) and photography. One ongoing study is about the automatic landing of a multi-copter drone. The current navigation method is the GPS navigation system and integrated GPS / Visual navigation system.

In this final project, computer vision will be applied to a quadcopter drone. The camera will help the vehicle to find the landing pad. The visual system can get the relative position of the vehicle towards the target. Then controls can be designed to reach the landing point. The stability and accuracy of the landing is the focus and difficulty faced. With precise landings, the drones can be applied well to several missions such as package delivery drones that require precise landing accuracy.

The test results obtained in this final project are a landing system on a drone that can land automatically on a specified runway. Visual assistance is a major component in determining the position of a drone landing. The drone landed with an average error of 45 cm with the smallest error was 21cm.

Keywords: computer vision, drone, precision landing

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur kepada Tuhan YME atas berkat dan hikmat yang diberikan, penulis dapat menyelesaikan laporan tugas akhir dengan judul "**PENDARATAN PRESISI PADA *DRONE* BERBASIS KOMPUTER VISI**", sebagai salah satu persyaratan dalam menyelesaikan pendidikan program Strata-Satu di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.

Dalam penyusunan dan penyelesaian laporan Tugas Akhir ini penulis mendapatkan banyak sekali doa, bantuan dan dukungan dari berbagai pihak. Untuk itu penulis mengucapkan terima kasih yang sebesar – besarnya kepada:

1. Orang tua, yang tak henti-hentinya memberikan semangat dan kasih sayang yang luar biasa kepada penulis.
2. Seluruh keluarga terkhusus kepada abang saya Iman Situmorang yang menjadi panutan dan penyemangat untuk penyelesaian tugas akhir.
3. Bapak Astria Nur Irfansyah, S.T., M.Eng., Ph.D. selaku dosen pembimbing 1 atas gagasan topik tugas akhir serta bimbingan dan arahan untuk penulis selama mengerjakan tugas akhir.
4. Bapak Muhammad Attamimi, B.Eng., M.Eng., Ph.D. selaku dosen pembimbing 2 atas gagasan topik tugas akhir serta bimbingan dan arahan untuk penulis selama mengerjakan tugas akhir.
5. Bapak Dr. Ir. Djoko Purwanto M.Eng.; Bapak Dr. Ronny Mardiyanto S.T., M.T.; Bapak Ir. Tasripan M.T.; sebagai dosen penguji atas evaluasi, koreksi dan arahan yang diberikan pada tugas akhir ini.
6. Teman-teman bidang studi elektronika: Ayik, Andre, Maul, Luhung, Fauzi, Faishal, Adif, Abizhar yang telah membantu dan memberikan semangat kepada penulis selama menjalani perkuliahan di Departemen Teknik Elektro ITS.
7. Teman-teman tim Bayucaraka ITS, khususnya Soeromiber yang membantu ide dan komponen cadangan dalam menyelesaikan tugas akhir ini.
8. Teman-teman Etenier dan Rumah Yudas yang menjadi keluarga dan tempat pulang saat jenuh mengerjakan Tugas akhir.
9. Pahol selaku teman dan tempat bercerita untuk setiap masalah dalam penyusunan tugas akhir.

10. Teman-teman elektro: M. Farih dan Bagas yang membantu dalam memberikan ide dan masukan dalam mengerjakan tugas akhir ini.

Penulis menyadari bahwa masih banyak yang dapat dikembangkan pada tugas akhir ini. Oleh karena itu penulis menerima setiap masukan dan kritik yang diberikan. Semoga tugas akhir ini dapat memberikan manfaat bagi banyak pihak.

Surabaya, 27 juni 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

Tugas akhir merupakan suatu penelitian yang dilakukan sebagai persyaratan akademik untuk mendapatkan gelar sarjana teknik di Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Topik yang akan dibahas pada tugas akhir ini mengenai pendaratan secara presisi pada *drone* berbasis komputer visi.

Pada bab ini membahas mengenai hal-hal yang mendahului pelaksanaan tugas akhir. Hal tersebut meliputi latar belakang, perumusan masalah, tujuan penulisan, batasan masalah, metodologi penelitian, sistematika penulisan, dan relevansi.

1.1. Latar Belakang

Unmanned Aerial Vehicles (UAV) atau lebih dikenal dengan pesawat tampa awak banyak menarik perhatian peneliti untuk diekplorasi. *Drone* digunakan oleh akademisi dan peneliti untuk melakukan pengujian sistem yang dapat membantu pekerjaan manusia. Beberapa contohnya adalah *drone* untuk keperluan militer, pemantauan dan pemetaan, fotografi, dan lainnya.

Salah satu *drone* yang banyak dikembangkan saat ini adalah *quadcopter*. Kelebihan dari UAV jenis *copter* adalah kemampuan melakukan pendaratan (*landing)* dan *take-off* secara vertikal sehingga dikenal sebagai VTOL(*Vertical Take-Off and Landing)*

. Dengan kemampuan tersebut *quadcopter* dapat digunakan di berbagai lingkungan. Kemampuang *landing* merupakan salah satu yang penting untuk suatu *drone*. Contohnya pada aplikasi *drone* pengantar paket kesehatan, kemampuan pendaratan secara *autonomous* akan membantu misi terlaksana secara baik.

Sistem otonom menjadi hal yang penting dalam pengembangan pesawat tampa awak. Salah satunya adalah penerapan *quadcopter* yang mampu melakukan pendaratan otomatis dengan akurasi yang tinggi[2], [3]. Beberapa sistem yang telah ada yaitu dengan mengandalkan sensor GPS (*Global Positioning System)* [3] , ada juga yang menggunakan sensor kamera, dan menggunakan GPS dan Visual secara terintegrasi. Kelemahan dari sistem yang pertama adalah sensor GPS memiliki akurasi yang rendah dan hanya dapat digunakan pada ruangan terbuka. Dalam penggunaannya GPS digunakan pada objek yang besar dengan

radius delapan meter[4]. Untuk melakukan pendaratan dengann mengandalkan GPS sebagain besar *quadrotor* memiliki kesalahan 5 meter[5]. Kamera merupakan alat yang tepat dalam membantu mendeteksi posisi pendaratan.

Komputer visi pada *drone* akan membantu sebagai penentuan posisi relatif wahana terhadap landasan. Kelebihan sensor kamera adalah dapat digunakan baik di ruangan tertutup ataupun ruangan terbuka. Selain itu dengan menggunakan sistem visual pada *drone* dapat memperoleh banyak informasi lingkungan, memiliki adaptasi yang baik, ukuran kecil dan ringan [2]. Dengan mengintegrasikan GPS dengan kamera, navigasi *drone* akan lebih akurat. Hasil pengolahan citra akan dicari titik tengah landasan yang akan menjadi referensi untuk meningkatkan akurasi pendaratan *drone*. Diharapkan dengan mengintegrasikan kamera dan GPS navigasi dan pendaratan akan lebih presisi, sehingga sistem dari pendaratan dapat dimanfaatkan untuk membantu aktifitas manusia.

1.2. Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah:

1. Pengolahan citra untuk mendeteksi landasan pendaratan
2. Bagaimana cara *drone* mendarat dengan presisi pada landasan pendaratan.
3. Kontrol untuk memposisikan *drone* dengan mengandalkan komputer visi.

1.3. Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut:

1. Implementasi computer visi untuk pendaratan presisi pada *drone*.
2. Mengolah citra untuk mendeteksi landasan pendaratan yang sederhana
3. Kontrol pendaratan secara auto pada *drone*.

1.4. Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. *Drone* terbang dengan mode *position hold* mengandalkan GPS.
2. Sistem bekerja pada kondisi pencahayaan yang tepat
3. *Drone* diarahkan ke daerah landasan pendaratan secara manual.

1.5. Metodologi Penelitian

Langkah-langkah yang dikerjakan pada tugas akhir ini adalah sebagai berikut:

1. Studi literatur

Studi literatur bertujuan untuk mempelajari dan mengkaji teori dasar dan ataupun data-data penelitian sebelumnya yang relevan dengan tugas akhir yang akan dikerjakan. Literatur ini dapat diambil dari jurnal-jurnal, buku, artikel ataupun *paper* yang memiliki keabsahan sehingga dapat dijadikan referensi.

2. Perancangan dan Desain Awal

Merancang gambaran secara umum untuk *drone* yang akan dikembangkan. Mulai dari sisi mekanik, elektronik dan algoritma program yang akan digunakan. Pada bagian mekanik desain harus memiliki ukuran yang presisi dan stabil. Elektronik memfokuskan pada komponen-komponen dan sensor yang akan digunakan untuk menerbangkan *drone* secara stabil dan bisa mempertahankan posisi. Dalam tugas akhir ini *flight controller* akan diintegrasikan dengan raspberry pi yang berfungsi untuk mengolah gambar dari kamera dan mikrokontroller Arduino nano yang memberikan perintah kepada *drone*. Algoritma program untuk merancang sistem otonom pada *drone* sehingga dapat melakukan tugas pendaratan secara presisi.

3. Perakitan *Drone*

Setelah membuat rancagan mekanik dan elektronik, selanjutnya adalah merakit *drone* agar dapat diterbangkan. Sebelum menggunakan sistem otonom, *drone* harus dapat tebang secara stabil dan mempertahankan posisi (*hold position)*. Sehingga algortima dan program dapat diaplikasikan dengan baik.

4. Pengembangan Sistem

Sistem algortima dari pendaratan presisi pada *drone* dikembangkan dengan meninjau penelitian yang terdahulu. Pada tinjauan pustaka beberapa metode yang telah uji dan dikembangkan para peneliti akan diimplementasikan.

5. Pengujian

Pada tahap ini akan diuji sistem yang telah dirancang. Apakah sesuai dengan ekspektasi. *Drone* akan diuji melakukan pendaratan dengan menggunakan sistem visual dan tampa meggunakan pengolaha citra. Sehingga dapat dibandingka tingkat akurasi dari pendaratan sistem secara keseluruhan.

6. **Analisa dan Evaluasi**

Analisa dilakukan untuk mengetahui karakteristik dari *drone* dalam proses pendaratan. Jika belum sesuai maka dilakukan evaluasi terhadap sistem yang telah diuji.

7. **Penulisan Laporan Tugas Akhir**

Tahap akhir adalah penulisan laporan. Proses pengerjaan dan data-data hasil pengujian sistem dilaporkan dalam bentuk tulisan yang merupakan kesimpulan dari tugas akhir ini. Kesimpulan tersebut adalah jawaban dari topik permasalahan yang dianalisa.

1.6. Sistematika Penulisan

Dalam buku tugas akhir ini, pembahasan mengenai sistem yang dibuat terbagi menjadi lima bab dengan sistematika penulisan sebagai berikut:

- **BAB I: Pendahuluan**
  Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.
- **BAB II: Tinjauan Pustaka**
  Bab ini berisi tentang teori-teori penunjang dari penelitian yang pernah dilakukan. Selain itu juga menjelaskan setiap komponen baik perangkat keras atau perangkat lunak yang digunakan dalam tugas akhir ini.
- **BAB III: Perancangan Sistem**
  Bab peranacngan sistem akan membahas sistem yang digunakan pada realisasi tugas akhir. baik perangkat lunak ataupun perangkat keras.
- **BAB IV: Pengujian dan Analisis**
  Pada bab ini menguraikan tentang pengujian alat pada *drone* dan analisa hasil pengujian.
- **BAB V: Penutup**
  Bab ini berisi tentang kesimpulan yang diperoleh dari pembuatan alat serta saran untuk pengembangan lebih lanjut.

1.7. Relevansi

Hasil dari tugas akhir ini diharapkan mampu membantu pekerjaan manusia bidang transportasi khususnya untuk mengantarkan paket kedaerah bencana[6], [7]. Selain itu *drone* juga dapat digunakan untuk keperluan komersial[8]. Dengan pendaratan presisi paket dapat diantarkan dengan aman.

Halaman ini sengaja dikosongkan

BAB 2
TINJAUAN PUSTAKA DAN TEORI PENUNJUANG

Pada bab ini akan memaparkan terori-teori pendukung. Teori-teori ini menjadi dasar dalam penyusunan laporan tugas akhir ini. Tinjauan pustaka memuat terori dan penelitian yang telah dialakukan sebelumnya. Terori tersebut digunakan untuk menemukan solusi dalam permasalahan pada tugas akhir ini.

2.1. Tinjauan Pustaka

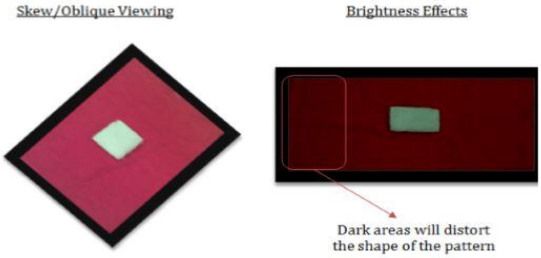
Unmanned Aerial Vehicle (UAV) atau *drone* adalah wahana yang tidak membutuhkan manusia untuk mengendarainya[19]. Salah satu contohnya adalah *quadcopter*, dimana dua motor bergerak searah jarum jam (*clockwise*) dan dua yang lain bergerak berlawanan arah jarum jam (*counter clockwise*) untuk menyeimbangkan posisi wahana [19],[10]. Aplikasi *drone* sudah sangat banyak , contohnya adalah mengantarkan logistik ke daerah berbahaya yang tidak dapat dicapai oleh manusia atau *dead zone* [11]. Daerah yang dituju tidak selalu dapat dicapai dengan mengandalkan sensor GPS saja. Pengolahan citra dapat membantu fungsi GPS menentukan landasauntuk pendaratan otomatis[2], [3], [11]–[13].

2.1.1. Image Processing pada *Drone*

Teknologi pengolahan citra pada *drone* sudah banyak dikembangkan. Teknologi ini sudah digunakan seperti keperluan navigasi [13], [14], membantu pendaratan otomatis [11], menghindari penghalang dan tabrakan , dan juga digunakan untuk pemetaan. Penerapan sistem pengolahan visual untuk *drone* juga sudah dikembangkan menjadi satu modul yang terintegrasi yaitu *optical flow.* Modul ini sangat membantu *drone* untuk mempertahankan posisinya saat terbang.

Pendaratan otamatis dengan mengandalkan GPS tidak menjamin untuk dapat mendarat pada landasan secara presisi. Akurasi yang dimiliki bisa error hingga 5 meter. Untuk mengatasi hal ini dapat dibantu dengan menggunakan sensor visual yaitu kamera[11]. Banyak ilmuan telah melakukan pengembangan teknologi *drone* berbasis visual, Venugopalan dan Taher [3] telah melakukan pengembangan teknologi pendaratan *drone* secara otomatis pada sebuah kendaraan laut otomatis. Algoritma

yang diadopsi terdiri dari dua bagian, yaitu *image processing* untuk mendeteksi landasan pendaratan, dan desain kontrol untuk navigasi. Pada algortima pengolahan citra mereka menggunakan *image processing toolbox* pada Matlab yang dipadukan dengan *C-code* untuk mengontrol *drone*. Metode yang digunakan dalam mendeteksi landasan pendaratan adalah memadukan antara pengenalan pola-pola pada landasan dan deteksi berdasarkan warna. Pola yang di deteksi adalah persegi panjang dari landasan pendaratan. Setelah mendeteksi warna, hal yang dipastikan adalah bentuk persegipanjang dari landasan, sehingga tidak terjadi kesalahan dalam mengenali landasan ketika menemui warna yang sama. Dengan mendeteksi bentuk persegi panjang, dapat ditentukan titik tengah yang menjadi acuan *drone*. Namun ada dua permasalahan dalam proses deteksi dengan metode ini yaitu kemiringan sudut pandang dan kecerahan. Seperti pada gambar 2.1 [3] permasalahan dalam pengenalan pola.

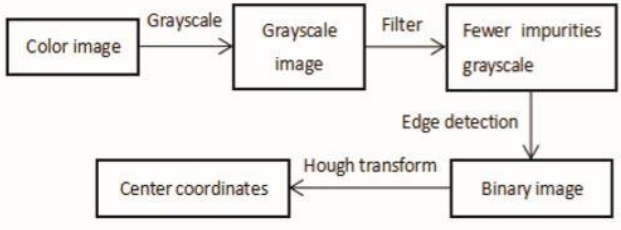


Gambar 2.1 Permasalahan pengenalan pola [3]

Navigasi menggunakan visual juga telah dikembangkan dengan mengintegrasikan dengan sensor inersia [12]. Dalam penelitian mereka pengolahan citra digunakan untuk mendeteksi pola ArUco. Prosesnya adalah gambar diubah menjadi *grayscale*, kemudian ke biner menggunakan adaptif *threshold* untuk mempersingkat waktu pengolahan gambar. Selanjutnya adalah mendektesi kontur gambar untuk mencari pola dari ArUco. Selain itu pengolahan visual juga digunakan untuk mengestimasi posisi 3D *drone*.

Jarak dan sudut pandang kamera terhadap objek sangat berpengaruh untuk pemrosesan visual. Nilai-nilai piksel dari gambar akan berbeda ketika objek yang sama ditangkap dengan posisi kamera yang berbeda. Untuk mengatasi hal tersebut dalam [2] mengembangkan sistem untuk mengatasi gangguan-ganguan tersebut. Bentuk dari landasan pendaratan

didesain berbentuk lingkaran konsentris dengan beberapa warna berbeda. Komputer visi digunakan untuk mencari titik koordinat dari landasan. Untuk mencari titik koordinatnya menggunakan *randomized Hough Transform.* Tahap-tahapnya secara singkat digambarkan dalam blok diagram pada gambar 2.2[2].



Gambar 2.2 Contoh Menentukan titik koordinat **[2]**

Proses deteksi berbasis bentuk objek yang digunakan adalah untuk mengabaikan informasi warna dengan mengubah kebentuk *grayscale* atau abu-abu. Proses pengolahannya adalah pertama gambar video diubah menjadi warna abu-abu, kemudian difilter. Setelah didapatkan gambar abu-abu, diperoleh informasi dengan deteksi tepi atau *edge* untuk mendapatkan gambar biner. Langkah terakhir adalah menentukan titik koordinat dengan *Hough transform*. Prinsip dasar *hough transform* adalah menggunakan titik dan garis untuk mengubah kurva yang ada pada gambar asli menjadi parameter titik dalam ekspresi kurva. Artinya melalui transformasi koordinat gambar, koordinat pesawat diubah menjadi parameter koordinat.

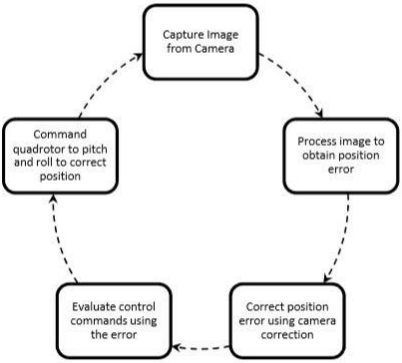
Untuk mempertahankan jarak kamera yang konstan, pada *drone* digunakan sensor ketinggian. Beberapa peneliti menggunakan sensor ultrasonik seperti pada [2] dan [12], ada juga yang menggunakan barometer [12] , dan lidar [15]. Dengan adanya sensor ketinggian, perkiraan koordinat z *drone* dapat ditentukan. Sebelumnya menggunakan visual untuk mengestimasi koordinat x, y dari *drone*.

2.1.2. Pendaraatan dan Kontrol

Secara garis besar alur proses pendaratan *drone* dapat dilihat pada diagram gambar 2.3 [3]. Dimulai dari kamera menangkap gambar video, diproses untuk menentukan posisi *drone* dan mengontrol *drone* menuju titik pendaratan yang ditetapkan. Ada dua bagian dalam proses pendaratan yaitu *search routine* dan *landing routine*[3]. *Search routine* adalah proses *drone* mencari letak landasan pendaratan. Pertama *drone* terbang

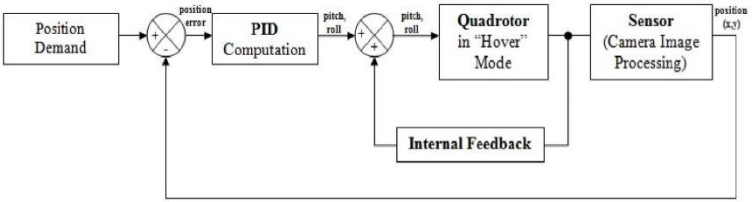
pada ketinggian tertentu , jika tidak menemukan landasan pendaratan , maka ketinggian terbang ditambah untuk memperluas bidang tangkap oleh kamera [2]. Setelah menemukan lokasi landasan, *drone* terbang menuju landasan dengan kontrol *pitch* dan *roll*. Setelah itu memasuki proses *landing routine*. Pada bagian ini kontrol PID untuk *pitch* dan *roll* digunakan untuk menyesuaikan posisi *drone* pada titik tengah landasan.

Hal yang sering terjadi adalah landasan hilang dari pandangan kamera. Hal ini terjadi disebabkan faktor seperti *drifting.* Mengatasi masalah tersebut, kontroler menggunakan data sebelumnya untuk menuntun *drone* kembali ke posisi terakhir saat mendeteksi landasan. Selain itu, penurunan ketinggian juga akan mengurangi bidang pandang saat proses pendaratan. penurunan yang terlalu cepat akan menyebabkan landasan tidak terdeteksi kamera. Oleh karena itu selama proses turun, jumlah kontrol ketinggian (*throttle)* dikaitkan dengan jarak penyimpangan terhadap target[2]. Ketika jarak terhadap target jauh, maka proses turun *drone* lambat, dan sebaliknya saat jarak *drone* tepat ditengah titik koordinat landasan, penurunan *drone* lebih cepat.



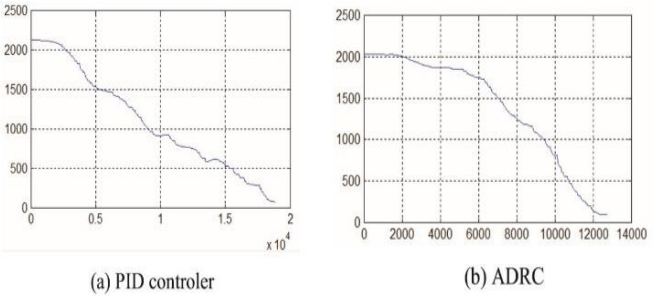
Gambar 2.3 Contoh Alur proses pendaratan [3]

Desain kontrol pendaratan *drone* banyak menggunakan kontrol tradisional namun efektif, yaitu PID. Kontrol *input* ada dua yaitu *pitch* dan *roll* untuk mengontorol posisi horizontal (x, y) *drone*. Diagaram pada gambar 2.4 adalah kontol *loop* untuk pendaratan *drone* oleh Venugopalan dan Taher [3]. Metode tuning PID yang digunakan adalah menggunakan Ziegler-Nichols dan kemudian dilanjutkan dengan tuning *trial and error.* Kontrol ini akan secara berkala mengontrol *drone* untuk tetap pada titik tengah landasan pendaratan.



Gambar 2.4 diagram blok Loop Kontrol PID *drone* [3]

Desain kontrol ADRC (*auto disturbance rejection controller)* juga telah dikembangkan oleh Bai, dkk pada [2]. Sistem kontrol ini menunjukkan kemampuan lebih baik dibandingkan kontrol tradisional PID. Kontrol ADRC terdiri dari tiga bagian: *tracking differentiator* berfungsi untuk mengurangi kesalahan awal sistem, mengatur transisi dari *input* yang diberikan sesuai kebutuhan sehingga respon *step* dicapai dengan cepat tampa *overshoot, extended state observer* untuk mengamati status lanjutan dan *non-linear error feedback* yaitu umpan balik eror non-linear. Kelebihan dari desain kontrol ini adalah *overshoot* rendah, dan anti interferensi yang kuat. Gambar 2.5[2] menununjukkan perbandingan performa antara kontrol PID tradisional dengan ADRC[2].



Gambar 2.5 Perbandingan Pendaratan dengan PID dan ADRC [2]

2.1.3. Pendaratan presisi drone

Sistem pendaratan otomatis menjadi hal penting dalam pengembangan drone. Salah satunya adalah penerapan *quadcopter* yang mampu melakukan pendaratan otomatis dengan akurasi yang tinggi[2], [3]. Pendaratan presisi pada drone dikembangkan dengan alasan kebutuhan pendaratan yang memiliki akurasi yang baik. Pendaratan

presisi banyak dikembangkan menggunakan komputer visi untuk mendarat pada suatu platform dengan ukuran tertentu. Salah satu penelitian tentang pendaratan presisi dilakukan menggunakan sensor dengan biaya rendah [5]. Dalam penelitianya sukses melakukan pendaratan presisi pada suatu platform dengan rata-rata error 0.3705 m pada ruangan tertutup. Selain itu berhasil juga diuji pada ruangan terbuka dengan rata-rata error 0.3888 m.

Pendaratan presisi juga dilakukan menggunakan optical flow dan GPS/INS yang terintegrasi[16]. Pada penelitian tersebut melakukan pendaratan presisi dengan maksimal error 0.27 m. Pengujian pendaratan dilakukan dengan variasi ketinggian 60cm hingga 150cm.

Sistem pendaratan otomatis juga dikembangkan menggunakan komputer visi untuk deteksi dan pengenalan terget yang tepat[17]. Target *helipad* yang menjadi target memiliki ukuran 120x120 cm. Sistem yang dikembangkan dengan mengkombinasikan visual dengan GPS untuk navigasi. Studi ini mencapai pendaratan yang tepat dengan rata-rata error posisi 42 cm.

Optical flow juga dikembangkan untuk navigasi UAV secara otomatis[18]. Pada penelitian ini penentuan posisi 2D menggunakan prinsip optical flow. Dengan menambahkan sensor ultrasonic, inframerah dan sensor tekanan dapat mengestimasi ketinggia sehingga posisi 3D dapat dihitung. hasil evaluasi pada studi tersebut berhasil melakukan pendaratan presisi dengan error 30 cm

Pengembangan sistem pendaratan presisi juga dikembangkan dengan menggunakan komputer visi[19]. Penulis menjelaskan berdasarkan hasil simulasi, sistem yang dirancang mendarat presisi dengan rata-rata error 5 cm. Pada studi ini UAV dapat mendarat hingga error 30 cm.

Pendaratan presisi dengan algoritma adaptif fuzzy data fusi dikembangkan dengan mengambil nilai tranlasi x dan y dari sensor optical flow[20]. Algoritma ini dikembangkan untuk mendapatkan perkiraan keadaan akurat saat melakukan pendaratan. Metode yang digunakan dilakukan pada ruangan tertutup dan mencapai pendaratan presisi dengan error posisi maksimum 10 cm.

Pendaratan presisi dikembangakan dengan metode pertukaran data telemtri dan fusi sensor GNSS (*Global Navigational Satellite System*), deteksi penanda inframerah dan beberapa sensor lainnya[21]. Dalam studi ini penulis menggunakan sistem visi yang terdiri dari penanda pelacakan dan kamera inframerah. Sistem visi pada studi ini lebih stabil karena

menggunakan penanda inframerah. Dengan metode tersebut penulis berhasil melakukan pendaratan dengan error posisi 8 cm.

Pendaratan presisi yaitu pendaratan dengan penggabungan beberapa sensor tambahan untuk memperbaiki akurasi pendaratan dibandingkan dengan menggunakan GPS/INS. Penelitain yang paling popular adalah pendaratan dengan bantuan visual. Pendaratan presisi dikembangkan dengan tujuan drone mampu mendarat pada *platform* tertentu. Tabel 2.1 adalah perbandingan pendaratan presisi pada drone yang telah dikembangkan sebelumnya.

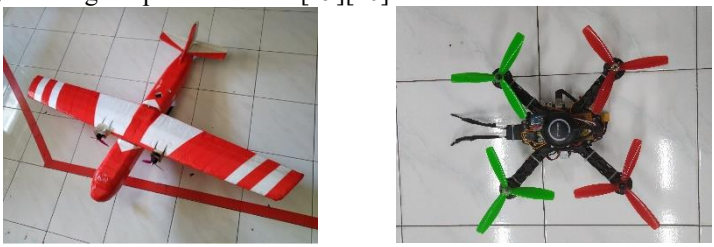
Tabel 2.1 Rangkuman pendaratan presisi penelitian sebelumnya

No	Makalah	Error posisi pendaratan presisi
1	Auto Takeoff and Precision Landing Using Integrated GPS/Optical Flow Solution[16]	Maksimal 0.27 m
2	Precision landing using an adaptive fuzzy multi-sensor data fusionarchitecture[20]	Kurang dari 0.1 m
3	An Autonomous UAV with an Optical Flow Sensor for Positioning and Navigation[18]	0.30 m
4	Drone Precision Landing using Computer Vision[19]	Maksimal 0.05 m
5	Precision Landing of a Quadrotor UAV on a Moving Target Using Low-cost Sensors[5]	Rata-rata 0.38 m
6	Multirotor UAV sensor fusion for precision landing[21]	0.08 m
7	Visually-Guided Landing of an Unmanned Aerial Vehicle[17]	Rata-rata 0.42 m

2.2. Drone

Unmanned Aerial Vehicle (UAV) atau *drone* adalah wahana yang tidak membutuhkan manusia untuk mengendarainya[19]. Karena tidak memiliki awak, UAV harus dikendalikan dari jarak jauh menggunakan remote control (RC) dari luar wahana. Selain itu UAV juga dapat bekerja secara otomatis berdasarkan program yang sudah ditanamkan pada sistemnya. Secara umum drone didesain dengan ukuran yang kecil.

Berdasarkan konfigurasi *airframe,* UAV dibedakan menjadi dua yaitu *fixed-wing* dan *rotary wing.* Gambar 2.6 adalah contoh penampakan fisik pesawat *fixed-wing* dan *rotary wing*. *Fixed-wing* adalah UAV yang memiliki bentuk sayap yang sudah tetap sedangkan *rotary wing* merupakan UAV yang komponen geraknya berupa baling-baling yang berputar(rotor). Salah satu contohnya adalah *quadcopter*, dimana dua motor bergerak searah jarum jam (*clockwise*) dan dua yang lain bergerak berlawanan arah jarum jam (*counter clockwise*) untuk menyeimbangkan posisi wahana [19][10].



Gambar 2.6 Pesawat *fixed-wing* (kiri) dan *quadcopter* (kanan)

Kontrol gerak dihasilkan dengan mengatur kecepatan rotor untuk mengubah torsi dan gaya dorong dari masing-masing rotor.Percepatan vertikal dari *quadcopter* dikendalikan dari kecepatan keempat motor. Torsi yang dihasilakan oleh rotor dapat dihitung dengan persamaan[22]

$$Q = K_q \cdot I \qquad (2.1)$$

$$V = R_a.I + K_e.\omega \qquad (2.2)$$

Dimana Q adalah torsi yang dihasilkan oleh rotor , V merupakan tegangan rotor , I adalah arus rotor, dan ω adalah sudut yang dihasilakan ketika berputar. Sedangkan untuk K_q adalah konstasnta torsui motor, K_e adalah konstanta kecepatan motor, dan R_a adalah konstanta resistansi rotor yang digunakan. Dengan mengetahui arus dan tegangan motor kita dapat menghitung daya yang dikonsumsi oleh motor. Dimana rumus daya adalah:

$$P = IV = \frac{Q}{Kq}V \qquad (2.3)$$

Daya yang dihasilkan oleh motor dalam kondisi ideal sama dengan torsi motor dikalikan dengan kecepatan motor. Ketika *hover* daya didapat dengan:

$$P = T_{vh} \qquad (2.4)$$

Dimana $_{vh}$ adalah perubahan kecepatan udara yang disebabkan oleh baling-baling. Dimana T pada persamaan 2.5 adalah gaya dorong yang dihasilkan untuk tetap *hover.* A adalah luas permukaan lingkaran yaitu

daerah turbulensi oleh rotor. Dan p adalah massa jenis udara.

$$vh = \sqrt{\frac{T}{2pA}} \qquad (2.5)$$

Daya angkat adalah salah satu parameter yang harus diperhitungkan dalam merancang UAV. *Lift force* atau daya angkat berhubungan dengan model airfoil pada pesawat *fixed-wing*. Namun untuk jenis *rotary wing* daya angkat juga dihasilkan oleh *propeller*. Pada dasarnya *lift* adalah usaha angkat untuk melawan gravitasi. Prinsip bernoulli menjelaskan bahwa aliran udara merupakan energi yang konstan. Ketika udara mengalir pada bagian dengan tekanan udara rendah maka aliran udara akan semakin cepat. Dari prinsip tersebut perbedaan tekanan udara menghasilkan gaya aerodinamik. Daya angkat dapat dicari dengan peresamaan

$$L = \frac{1}{2} p\, v^2\, A\, C_L \qquad (2.6)$$

Dimana p adalah massa jenis udara (1.225Kg/m^3) , v adalah kecepatan (m/s) , A adalah luas sayap (m^2) , dan C_L adalah koefisien *lift* didapat dari *ngel of attack mach number* dan *Reynolds*. Pada *rotary wing* daya angkat tida dihasilkan oleh sayap, namun dari *propeller* yang berputar. Sehingga quadcopter mampu *hover* dengan luas permukaan lingkaran hasil turbulensi dari *propeller*[22].

2.3. Komputer Visi

Komputer visi (*computer vision*) adalah transformasi data dari kamera diam atau video menjadi suatu representasi baru. Contoh represantasi baru yang dimaksud bisa mengubah gambar warna menjadi gambar skala abu-abu[23]. Pengolahan citra merupakan proses awal (*preprocessing*) komputer visi. Pengolahan citra bertujuan untuk memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau komputer. Contohnya adalah pemampatan citra (*image compression*), perbaikan kualitas citra (*image enhancement*), pemugaran citra (*image restoration*), segmentasi citra (*image segmentation*), dan rekontruksi citra (*image recontruction*).

Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Artinya masukanya adalah citra dan keluaranya juga citra, namun keluarannya lebih baik dibandingkan citra masukan.

Citra dapat diartikan suatu keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, analog berupa sinyal-sinyal video atau berupa digital yang langsung disimpan dalam pita magnetik. Kata citra

atau biasa dikenal dengan gambar diartikan sebagai fungsi intesitas cahaya dua dimensi ,yang dinyatakan oleh fungsi *f(x,y)* ,dimana nilai *f* menyatakan instensitas atau kecerahan citra pada satu titik [24]. Misalkan suatu pixel dengan fungsi *f* (20,10) = 6, hal tersebut berarti pixel pada koordinat x = 20 dan y= 10 dari pojok kiri atas gambar memiliki tingkat kecerahan 6.

Kedalaman warna atau biasa dikenal dengan kedalaman pixel. Berdasarkan kedalaman citra dibagi menjadi citra warna , citra grayscale, dan citra biner[25].

2.3.1. *Color Spaces*

Dalam visi komputer dan pemrosesan gambar, ruang warna mengacu pada cara khusus mengatur warna. Ruang warna sebenarnya merupakan kombinasi dari dua hal: model warna dan fungsi pemetaan. Model warna membantu dalam merepresentasikan nilai piksel[26].

Ada banyak jenis ruang warna. Beberapa yang sering digunakan adalah RGB, HSV dan CMYK, dan sebagainya. Setiap ruang warna memiliki kelebihan dan kelemahan masing-masing dalam proses citra.

- RGB: merupakan ruang warna yang paling sering ditemui dan digunakan. RGB adalah singkatan dari *Red* (merah)*, Green* (hijau)*, Blue* (biru). Dalam ruang warna ini setiap warna direpresentasikan sebagai kombinasi merah, hijau dan biru. Setiap piksel direpresentasikan sebagai kanal dari tiga warna tersebut dalam rentang nilai 0 sampai dengan 255.
- HSV: model warna HSV adalah representasi silindris dari model RGB standar. HSV singkatan dari Hue, Saturation, dan Value. Hue diukur dalam derajat dan bervariasi 0-360 yang merupakan penentu warna dasar. Hue akan mempengaruhi nilai warna. Saturation dan Value menentukan kedekatan dengan putih dan hitam. Saturation mewakili tingkat intensitas warna. Pada tingkat kecerahan (*value)* yang konstan, nilai saturasi akan menggambarkan kedekatan suatu warna pada warna abu-abu. Pada library Opencv nilainya bervariasi 0-255 , namun model dasarnya variasi nilainya adalah 0-100[27]. *Value* pada HSV merepresentasikan tingkat kecerahan warna. Rentang nilainya adalah antara 0 (minimum) sampai dengan 1(maksimum). Transformasi konversi RGB ke HSV setiap pikselnya pada adalah sebagai berikut:

Nilai Hue dapat dihitung dengan persamaan:

$$H = \begin{cases} 0, & C_{max} = 0, \\ 60 \times \frac{G-B}{M}, & C_{max} = R, \\ 60 \times \frac{B-R}{M} + 120, & C_{max} = G, \\ 60 \times \frac{R-G}{M} + 240, & C_{max} = B. \end{cases} \quad (2.7)$$

persamanaa (1) adalah rumus untuk mengkonversi nilai RGB menjadi nilai Hue. C_{min} adalah nilai manimal dari R, G, B dan C_{max} adalah nilai maksimalnya dan M adalah selsih antara C_{min} dan $C_{max.}$

Nilai Saturation dihitung dengan persamaan (2.8):

$$S = \begin{cases} \frac{M}{C_{max}}, & C_{max} \neq 0, \\ 0, & C_{max} = 0. \end{cases} \quad (2.8)$$

Dan untuk Value dihitung dengan persamaan (2.9):

$$V = C_{min} \quad (2.9)$$

- CMYK: model warna ini lebih sering dipakai dalam bidang percetakan. Model warna ini tidak terlalu popular dalam proses pengolahan citra di komputer. CMYK sendiri terdiri dari empat warna yaitu: *Cyan, Magenta, Yellow,* dan *Key.*

2.3.2. Segmentasi citra

Segmentasi citra adalah proses membagi atau memecah citra digital menjadi beberapa bagian/segmen yang saling berhubungan. Dalam proses membagi citra, dilakukan dengan berdasarkan pendekatan-pendekatan tertentu. Tujuan segmentasi adalah mendapatkan representasi sederhana yang berguna dari citra digital. Ada tiga jenis pendekatan yang sering digunakan diantaranya:

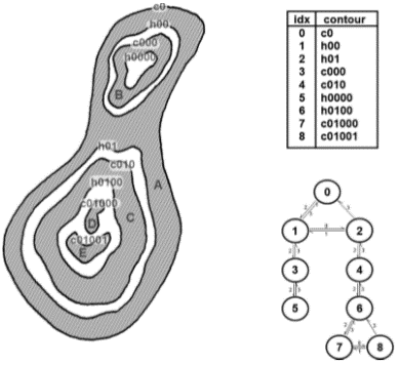
- Pendekatan batas (*boundary approach*) yaitu pendekatan untuk mendapatkan batas yang ada antar daerah.
- Pendekatan tepi (*edge approach*) pendekatan untuk mendeteksi piksel tepi dan menghubungkan masing-masing piksel mejadi suatu batas tertentu
- Pedekatan daerah (*region approach*) yaitu pendekatan yang bertujuan untuk membagi citra menjadi daerah-daerah sehingga didapatkan daerah sesuai kriteria yang diinginkan [28].

Pendekatan daerah merupakan salah satu yang sering digunakan.

Segmentasi warna merupakan proses segmentasi dengan pendekatan daerah dengan menganalisis nilai tiap warna pada citra digital. Citra digital secara umum menggunakan model warna RGB, namun pada pengolahan citra model warna HSV lebih sering digunakan. proses segmentasi warna dengan menggunakan deteksi warna HSV menghasilkan segmen warna yang akurat. Hasil segmentasi warna adalah bagian citra yang disebut dengan blob, yaitu sekumpulan piksel bertetangga yang memiliki nilai tertentu[28].

2.3.3. *Kontur*

Kontur adalah daftar titik yang memiliki kesamaan dalam suatu area pada gambar. Analisis kontur adalah alat yang sangat berguna dibidang komputer visi. Banyak bentuk di dunia nyata dapat dianalisis bentuknya dengan melihat konturnya. Ketika kita mengkonversi gambar menjadi skala abu-abu dan memberi ambang batas (*threshold*) maka akan banyak garis dan kontur[26]. Opencv memiliki fungsi yang disebut dengan cv:: FindContours (). Fungsi ini menggunakan konsep *contour tree*. Pada gambar 2.7 [23] bagian kanan merupakan hasil findContours opencv. Ada lima daerah berwarna (A, B, C, D dan E) kontur yang terbentuk dari tepi eksterior dan tepi interior tiap daerah yang berwarna. Jadi terdapat Sembilan kontur yang terdaftar pada tabel pada bagian kanan. Grafik bagian kanan bawah adalah contour tree yang terbentuk, dimana setiap node adalah kontur.



Gambar 2.7 Konsep *contour tree* [15]

Dengan dapat dianalisa bentuk-bentuk dan karakteristik dari suatu gambar. Contoh paling sederhana adalah menghitung luas kontur, atau menentukan titik tengah bentuk geometri dari gambar. Persamaan (2.10) adalah rumus untuk menghitung momen pada suatu kontur citra digital. Nilai p mengacu pada piksel kontur, w mengacu pada *weight*, N adalah jumlah titik dalam kontur, k mengacu pada *power* dan I mengacu pada momen. Dengan menentukan nilai w dan k dapat mengekstraksi karakteristik kontur. Contoh untuk mencari luas area kontur hanya perlu

$$I = \sum_{i=0}^{N} w_i p_i^k \qquad (2.10)$$

mengatur w = 1 dan k = 0, ini akan memberi area kontur.

2.3.4. *Opencv*

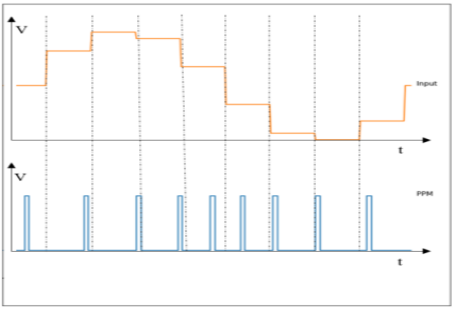
Gambar 2.8 Logo OpenCV

Opencv merupakan *library* dari komputer visi yang dapat dipakai secara *open source*. Artinya dapat digunakan secara gratis baik untuk keperluan komersil ataupun untuk akademik. Library ini dapat digunakan dalam banyak Bahasa pemograman seperti C, C++, Python, Java, dan lainya. *Library* opencv sangat membantu dalam pengolahan citra. Salah satu tujuan opencv adalah untuk menyediakan infrastruktur komputer visi yang mudah digunakan. Opencv memiliki lebih dari 500 fungsi yang mencakup banyak bidang dalam penglihatan seperti inspeksi produk pabrik, *medical imaging,* antarmuka pengguna (*user interface*) , kalibrasi kamera, dan robotika[23]. Gambar 2.8 adalah logo opencv resmi yang ada di situs *http://opencv.org.*

2.4. Pulse Position Modulation (PPM)

Pulse position modulation (PPM) adalah teknik modulasi pulsa dengan memvariasikan posisi pulsa sinyal pembawa secara proporsional dengan sinyal pesan. Dimana posisi pulsa dalam interval yang telah ditentukan, disesuaikan tergantung nilai sinyal *input* pada waktu tertentu.

Lebar dan amplitudo *output* dari sinyal ppm akan selalu konstan. Teknik modulasi ini kebal terhadap *noise* karena amplitude tidak dipertimbangakan selama pengambilan nilai[29].



Gambar 2.9 *Input* dan *output* sampel PPM [20]

PPM juga merupakan sinyal analog tetapi berbeda dengan PWM (*pulse width modulation*) yang menggunakan kabel terpisah tiap salurannya, PPM menumput masing-masing sinyal menjadi satu dan mengirimkannya menggunakan satu kabel. Hal ini membuat perkabelan pada drone lebih mudah. Variasi lain dari PPM yang diperkenalkan oleh produsen yaitu CPPM (*Combined Pulse Position Modulation*) dan PPMsum. Gambar 2.9 merupakan contoh sinya input berwarna orange dan siyal output ppm berwarna biru.

2.5. Komunikasi Serial

Ketika menggunakan dua processor dalam satu sistem akan membutuhkan komunikasi antar processor. Komunikasi serial memungkinkan proses pertukaran data antar dua processor. Komunikasi serial akan mengirim data secara bergantian dan berurutan. Komunikasi serial mengurangi distorsi sinyal , oleh karena itu memungkinkan untuk transfer data yang terpisah jarak sangat jauh[30].

2.6. Flight Controller OMNIBUS F4V3

Gambar 2.10 Perangkat OMNIBUS F4V3[16]

Flight controller (FC) adalah kumpulan komponen dan sensor yang berfungsi untuk menjaga *drone* tetap seimbang dan dapat dikendalikan. Dalam tugas akhir ini *Flight Controller* (FC) yang digunakan adalah OMNIBUS F4V3 yang memiliki pengontrol mikro STM32F405 LQFP64 (168Mhz, 1M flash, 192kB SRAM)[31]. Dilengkapi dengan regulator tengangan MP2359 sehingga dapat menerima input tegangan dari batrai secara langsung. Selain itu juga memilki USB CDC yaitu USB yang memiliki lebih dari satu antarmuka (*interface*). Beberapa firmware yang dapat digunakan pada perangakat ini adalah Betaflight, Cleanflight dan INav. Untuk menyimpan data log, omnibus memiliki fitur balckbox dengan menyimpan data sensor dan beberapa data lainya dari flicht controller. Gambar 2.10 [31] merujuk pada penampakan fisik Omnibus F4V3.

Berikut adalah fitur Omnibus F4V3:

- STM32F405 LQFP64 (168Mhz, 1M flash, 192kB SRAM)
- MPU6000 6DOF IMU
- BMP280 barometer
- 5V Switching regulator (MP2359)
- USB CDC serial, DFU firmware update
- micro SD card slot for Blackbox logging

2.7. Raspberry Pi 3 B+

Raspberry pi adalah SBC (*single board computer*) yang berukuran kecil yang dikembangkan di Wales oleh yayasan Raspberry Pi. Raspberry pi 3 B+ merupakan keluaran terbaru yang sudah dilengkapi dengan fitur wifi dan bluetooth. Sebelumnya perangkat komputer mini ini digunakan untuk keperluan pendidikan, namun saat ini banyak digunakan peneliti

dalam pengembangan keilmuan.

Gambar 2.11 Perangkat Raspberry Pi 3 B+ [23]

Raspberry pi 3 B+ ini memiliki prosesor Broadcom BCM28370B0 yang dengan kecepatan clock sampai dengan 1.4GHz. Memiliki memory 1GB LPDDR2 SDRAM. LPDDR adalah jenis RAM DDR (*Double Data Rate)* yang dikembangkan untuk komputer mobile. LPDDR adalah singkatan dari *Low Power Double Data Rate.* Artinya memoy ini mengonsumsi daya yang kecil yaitu 1.2V. Raspberry pi 3 B+ ini dilengkapai juga dengan port USB untuk menghubungkan dengan perangkat tambahan seperti kamera, mouse, keyboard, dll. Port CSi juga tersedia pada papan sirkuit yang berfungsi untuk menghubungkan dengan kamera raspberry pi. Gambar 2.11[32] adalah perangkat keras dari raspberry pi .

Berikut adalah spesifikasi dari raspberry pi 3 B+:

- Processor : Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
- Memory : 1GB LPDDR2 SDRAM
- *Input* power : 5V/2.5A DC via micro USB connector, 5V DC via GPIO header, Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
- Access : Extended 40-pin GPIO header

2.8. Arduino Nano

Arduino nano dengan pengontrol mikro ATmega328 yang dioperasikan pada tegangan 0-5V. Perangkat ini memiliki kecepatan clock 16MHz dan dilengkapi dengan 14 port digital I/O (yang menyediakan 6 pin PWM) dan 8 port analog I/O[33].

Gambar 2.12 Perangkat Arduino Nano

Berikut adalah spesifikasi arduino nano:

- *Microcontroller* : Atmel ATmega328
- Tegangan operasi (level logika) : 5V
- Tegangan *input* (rekomendasi) : 7-12V
- Tegangan *input* (limit) :6-20V
- Pin digital I/O :14 (dengan 6 pin ouput PWM)
- Pin analog input : 8
- Arus DC untuk pin :40mA
- Flash Memory :32 KB
- SRAM :2 KB
- EEPROM :1 KB
- Kecepatan clock :16MHz
- Dimensi : 0.73” x 1.70”

Halaman ini sengaja dikosongkan

BAB 3
PERANCANGAN SISTEM

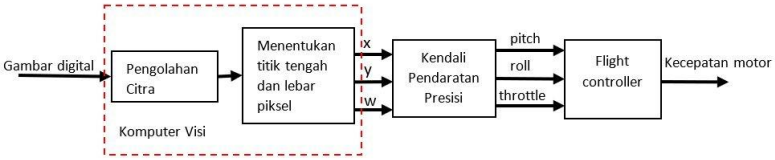
Pada bab ini akan dibahas perancangan sistem pendaratan otomatis pada *drone* yang dapat mendarat presisi pada landasan yang telah ditentukan. Dari beberapa tinjauan pustaka yang telah dibahas pada bab sebelumnya, pendaratan presisi yaitu ketika drone dapat mendarat pada suatu landasan, *marker* atau *paltform* tertentu. Pada tugas akhir ini landasan pendaratan yang digunakan memiliki ukuran 48x40cm. Pendaratan berhasil melakukan pendaratan presisi jika drone berhasil mendarat pada pada landasan tersebut.

Dalam perancangan tugas akhir ini, beberapa pertimbangan dalam pemilihan drone yang akan digunakan adalah sebagai berikut:

1. Drone dapat terbang dengan *flight mode position hold*, baik secara manual maupun secara otomatis.
2. Memungkinkan untuk mengakses lokasi melalui informasi yang didapat dari GPS pada drone.
3. Daya angkat (*lifting capacity*) *drone* yang cukup untuk membawa kamera dan unit pemrosesan visual yaitu mini komputer.
4. Akses dan mampu memodifikasi variabel penerbangan otomatis seperti *pitch, roll, yaw,* dan *throttle.*

Setelah melakukan studi tentang drone, pemilihan drone berdasarkan keemapat poin diatas, perancangan akan dibagi menjadi dua yaitu perancangan perangkat keras dan rancangan perangkat lunak. Perancangan perangkat keras meliputi mekanik dari *drone* dan pemasangan tiap komponen elektronik *drone*. Tujuan utama dalam perancangan *hardware* ini adalah meminimalisir ketidakseimbangan *drone*. Artinya penempatan setiap komponen akan fokus agar titik berat *drone* berada ditengah sehingga drone mampu terbang dengan struktur yang memiliki stabilitas yang baik. Sedangkan untuk perangkat lunak (*software*) meliputi perancangan sistem deteksi objek landasan dengan kamera dan kontrol pendaratan otomatis pada *drone*. Pendeteksian landasan ini bertujuan untuk mengetahui posisi landasan yang akan dituju. Dengan mengetahui posisi relatif landasan terhadap drone dapat dijadikan referensi yang akan dituju untuk pendaratan. Dengan begitu drone mampu mendarat dengan presisi pada posisi landasan yang ditentukan. Landasan didesain memiliki warna cerah dan memberi bentuk lingkaran pada bagian tengah. Dengan adanya lingkaran tersebut akan mempermudah proses deteksi objek melalui kamera. Tujuan lingkaran ditempatkan pada

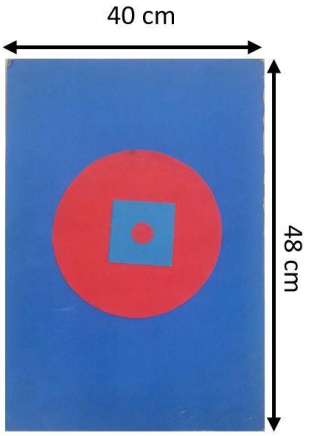
bagian tengah landasan adalah untuk mencapai tujuan pendaratan presisi pada drone. Gambar 3.1 adalah diagram blok keseluruhan sistem.



Gambar 3.1 Diagram blok keseluruhan sistem

3.1. Komputer Visi

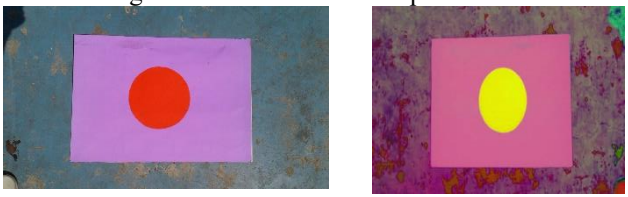
Komputer visi adalah transformasi data dari gambar atau video menjadi representasi baru. Pada perancangan ini komputer visi digunakan untuk mendapatkan nilai koordinat posisi relative landasan terhadap drone. Pada gambar 3.1 bagian komputer visi dapat dilihat terbagi menjadi dua. Pertama adalah bagian pengolahan citra dan penentuan titik tengah koordinat landasan. Pada bagian komputer visi ini bertujuan untuk mendeteksi landasan pendaratan seperti pada gamabr 3.2. Ukuran landasan pendaratan yang akan deteksi adalah 48x40 cm.



Gambar 3.2 Landasan Pendaratan

3.1.1. Pengolahan Citra

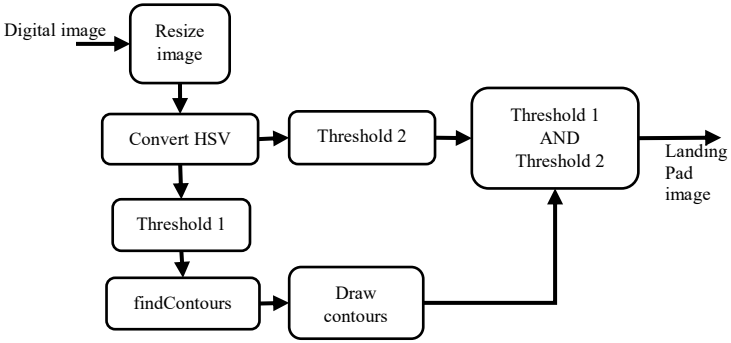
Pengolahan citra bertujuan untuk mendeteksi landasan pendaratan. sebelum merancang algoritma pengolahan citra, landasan dibuat dengan bentuk dan warna tertentu agar mudah dideteksi oleh kamera. Pertama adalah dengan memilih bentuk yang unik yaitu lingkaran. Kemudian untuk memperkuat dan mempermudah deteksi, perpaduan dua warna menjadi salah satu solusinya. Gambar 3.3 adalah contoh gambar dari landasan pendaratan yang di deteksi oleh kamera. Perpaduan dua warna yaitu merah dan ungu akan membantu dalam proses deteksi.



Gambar 3.3 Landasan pendaratan RGB (kiri) dan HSV (kanan)

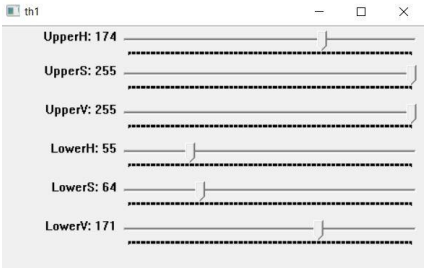
Secara garis besar pengolahan citra yang dirancang adalah untuk mencari titik tengah dari landasan pendaratan. Dimulai dengan memuat citra digital dari kamera. Gambar yang dimuat oleh kamera masih dalam ruang warna RGB. Ruang warna rgb adalah jenis yang paling popular di dunia nyata, namun untuk diolah dalam komputer visi, rgb sangat sensitif dengan perubahan intesitas cahaya sehingga harus dicari ruang warna lain. HSV adalah salah satu yang paling sering digunakan dalam preprocessing komputer visi. Gambar 3.3 (kanan) adalah hasil konversi gambar dari rgb ke hsv. Opencv sudah menyediakan fungsi untuk konversi gambar dalam ruang rgb menjadi hsv dengan syntax:*cv2.cvtColor(img, cv2.COLOR_BGR2HSV)*

Setelah citra digital dimuat dan dikonversi kedalam ruang warna hsv, selanjutnya adalah mencari titik tengah. Untuk bisa mencari titik tengah terlebih dahulu dilakukan *preprocessing* gambar.



Gambar 3.4 Diagram blok pengolahan citra

Dapat dilihat gambar 3.4 adalah diagaram blok pengolahan citra untuk mendeteksi landasan. Metode yang digunakan adalah dengan menggunakan ambang atas (*threshold)* warna. Prosesnya dilakukan dengan melakukan *threshold* terhadap dua warna. Cara untuk mendapatkan *threshold* dilakukan secara manual dengan *trackbar threshold*. Gambar 3.5 adalah cara untuk melakukan threshold terhadap citra digital. Hasilnya adalah gambar hitam putih. Warna putih adalah hasil dari *threshold*

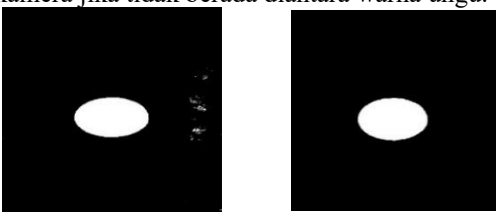


Gambar 3.5 Trackbar threshold HSV

Setelah dilakukan *threshold* terhadap dua warna, dengan memanfaatkan fungsi *findContour* dan *drawContour* pada opencv yang di aplikasikan pada hasil *threshold* 1 atau bagian luar dengan mengatur parameter ketebalan (*thickness*) sama dengan -1, pada syntax *cv2.drawContours*, maka hasilnya akan utuh seperti pada gambar 3.6 bagian kanan.

Gambar 3.6 Hasil *Threshold*(kiri) dan Hasil *Drawcontour*(kanan)

Jika hanya menggunakan satu kali *threshold* maka objek yang yang di deteksi sangat rentan terhadap *noise* dan banyak warna-warna mirip yang tidak diinginkan masih ikut terdeteksi. Maka dengan menggunakan logika AND antara nilai piksel pada hasil penggambaran kontur dari *threshold* pertama dengan hasil *threshold* kedua maka dihasilkan gambar 3.7 bagian kanan. Gambar yang dihasilkan akan lebih selektif. Dengan metode ini objek yang terlihat adalah warna merah yang berada diantara warna ungu. Sehingga meskipun ada warna merah lainya tidak akan terdeteksi oleh kamera jika tidak berada diantara warna ungu.



Gambar 3.7 Hasil Threshold kedua(kiri), Hasil akhir (kanan)

Dengan menggunakan metode threshold sederhana sudah dapat mendeteksi landasan dengan baik dan tahan terhadap noise, hal ini akan membuat drone mengenali landasan saja sehingga pendartan pada drone lebih presisi.

3.1.2. Mencari Titik Tengah Landasan

Hasil pengolahan citra adalah deteksi landasan pendaratan. Setelah mendapatkan gambar tersebut dapat diolah untuk mengetahui posisi pada gambar yang ditangkap kamera. Dengan menggunakan fungsi *findcontour* dapat dengan mudah menentukan titik tengah dari kontur yang terdeteksi. Kontur yang terdeteksi akan Digambar menajdi blob atau gumpalan berwarna putih. Kemudian dengan menggunakan *cv2.boundrect()* untuk mendapatkan posisi koordinat blob yang terdeteksi beserta panjang dan lebarnya. program untuk menghitung titik tengah

konturnya adalah sebagai berikut:

```
import cv2
import numpy as np
_,contours,hirearchy=cv2.findContours(threshold,cv2.RETR_LIST,
                    cv2.CHAIN_APPROX_SIMPLE,offset=(0, 0))
for cnt in contours:
          cv2.drawContours(threshold, [cnt], -1, (255, 255, 255), -1)
          areaK = cv2.contourArea(cnt)
          if (areaK > 100):
                    x,y,w,h = cv2.boundingRect(cnt)
                    cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
                    cent_rectx = (x+(w/2))
                    cent_recty = (y+(h/2))
```

Dengan menggunakan *syntax cv2.boundRect*() didapatkan empat parameter koordinat titik sudut persegi yang menglilingi blob dan panjang dan lebar blob. Nilai (x, y) yaitu koordinat piksel titik sudut kiri atas dari persegi yang mengelilingi blob atau kontur yang terdeteksi. Lalu w dan h adalah panjang dan lebar persegi. Untuk menyeleksi *noise* yaitu blob - blob kecil maka di filter dengan area piksel minimal 100 piksel. Fungsi *cv2.contourArea* bisa mendapatkan luasan setiap kontur yang terdeteksi.

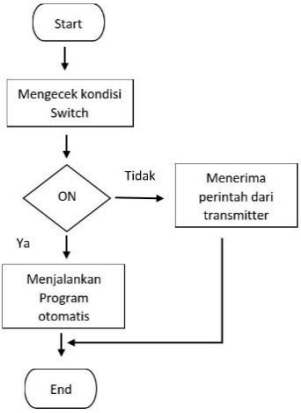
Untuk mengetahui jarak *drone* dengan titik tengah landasan adalah dengan memberikan tanda pada titik tengah gambar yang ditangkap. Karena ukuran gambar yang ditangkap kamera di ubah ukurannya menjadi 160x160 piksel, maka kita dapat menghitung selisih posisi *drone* terhadap titik tengah landasan. Selisih x adalah (80 - cent_rectx) dan selilih y = (80 - cent_recty).

3.2. Kendali Pendaratan Presisi

Sistem kendali yang digunakan adalah sistem kendali sederhana. Sistem ini baik digunakan dengan parameter drone terbang dengan *flight mode position hold.* Sehingga dengan kontrol yang sederhana dapat digunakan pada drone untuk pendaratan presisi. Proses pendaratan otomatis yang dirancang akan mengontrol nilai *pitch, roll* dan *throttle*. Sedangkan untuk *yaw, channel 5* dan *channel 6* akan tetap independen.

Untul alasan keamanan saat drone terbang otomatis, dirancang satu switch untuk mengaktifkan dan menonaktifkan mode otomatis. Arduino akan menunggu switch *channel* 5 di aktifkan untuk mode pendaratan otomatis. Jika switch dalam kondisi tidak aktif maka sinyal dari receiver akan langsung di teruskan ke *flight controller* tampa ada modifikasi sinyal ppm. Sebaliknya saat *switch* diaktifkan makan Arduino akan menjalankan

program pendaratan otomatis. Gambar 3.8 adalah alur kerja arduino.



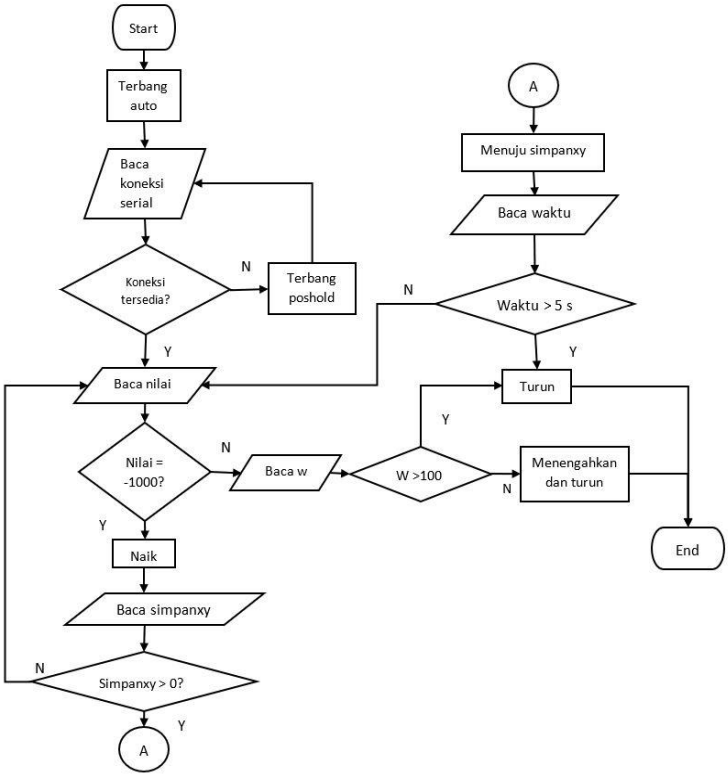
Gambar 3.8 Diagram alur kendali

Ketika Arduino menjalankan mode otomatis data yang dikirim oleh raspberry pi melalui komunikasi serial selanjutnya dibaca daca oleh Arduino. Ada tiga nilai yang diterima yaitu koordinat posisi relatif drone terhadap landasan (x, y) dan lebar blob hasil *threshold* (w). Kondisi saat landasan tidak terekam oleh kamera sehingga tidak ada blob yang terdeteksi, maka data serial akan tetap berjalan dengan hasil pembacaan -1000. Hal ini bertujuan agar Arduino nano mengetahui program pengolahan citra masi berjalan namun dalam proses pencarian landasan. Berbeda saat kondisi tidak ada data serial yang di terima oleh Arduino. Kondisi ini terjadi ketika program pengolahan citra tidak dijalankan atau kamera gagal mengakses gambar. Selama koneksi serial tidak ada maka drone akan terbang dalam mode *poshold* dan menunggu pembacaan data serial.

Ketika serial sudah tersedia Arduino mulai membaca nilai-nilai yang diterima. Hal pertama yang dilakukan adalah mengecek 3 nilai yang terbaca yaitu x, y (koordinat titik landasan) dan w (lebar piksel). Apakah nilainya lebih besar dari atau sama dengan 0, ketika nilai tidak lebih besar dari 0, yaitu nilai yang terbaca adalah -1000. Hal ini berarti drone tidak mengenali objek pendaratan atau landasan tidak dalam cakupan kamera. Algoritma ketika tidak ada landasan terdeteksi yaitu drone diperintahkan untuk menambah ketinggian terbang. Tujuannya adalah memperluas pandangan kamera. Dengan begitu drone mampu mencari posisi landasan

berada. Setiap penambahan ketinggian Arduino akan membaca nilai yang x, y yang tersimpan saat pembacaan nilai di awal pembacaan data serial. Gambar 3.9 adalah diagram alur kendali otomatis pada Arduino.

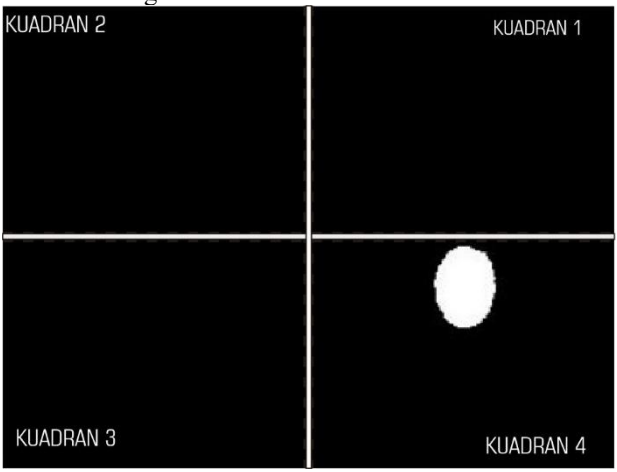
Pembacaan nilai yang tersimpan ini akan dicek lagi. Apakah nilai pembacaan sebelumnya masih -1000 (tidak ada landasan terdeteksi) atau tidak. Jika masih -1000 maka drone akan menambah ketinggian lagi. Pada saat nilai x, y dan w yang terbaca lebih besar dari nol, Arduino akan mengecek lebar blob (w) terlebih dahulu. Ukurannya lebih besar dari 100 artinya drone dalam kondisi terbang rendah diatas landasan. Lebar blob 100 piksel *drone* terbang pada ketinggian kurang dari 30cm. Saat kondisi seperti ini, ketinggian drone akan dikurangi sampai mendarat tampa proses menengahkan posisi.



Gambar 3.9 Diagram alur pendaratan otomatis pada arduino

Ketika data nilai posisi w lebih kecil dari 100 yang berarti ketinggian drone lebih dari 30 cm, selanjutnya adalah menjalankan perintah untuk menengahkan posisi ke landasan pendaratan dan turun dengan perlahan. Selama proses menengahkan posisi, nilai x, y disimpan tiap melakukan perintah. Sehingga ketika drone bergerak melewati landasan hingga tidak terdeteksi maka nilai yang tersimpan sebelumnya adalah titik acun drone bergerak. Drone akan bergerak mencari posisi terakhir selama 5 detik. Jika dalam 5 detik drone tidak menemukan landasan maka drone diperintahkan turun secara perlahan.

Sistem kendali menengahkan posisi drone menggunakan algoritma perkiraan posisi drone berdasakan kuadran kartesian. Gambar dari kamera akan dibagi menjadi 4 bagian. Kuadran 1 adalah bagian kanan atas, kuadran 2 adalah bagian kiri atas, kuadran 3 berada pada kiri bawah dan kuadran 4 adalah bagian kanan bawah.



Gambar 3.10 Pembagian kuadran gambar kamera

Ukuran gambar hasil pengolahan citra adalah 160x160 piksel. Titik tengahnya berada pada titik (80,80). Berdasarkan pemasagan kamera pada drone, titik (0,0) piksel berada pada pojok kiri bagian atas. Dengan asumsi posisi drone berada pada titik tengah gambar yaitu (80,80). Gambar 3.10 adalah pembagian kuadran gambar dari kamera. Lingkaran berwarna putih adalah landasan yang terdeteksi oleh kamera. Landasan berada pada kuadran 4.

Perintah keluaran dari Arduino yaitu: maju (sudut inklinasi *pitch* = +2.1°), mundur (sudut inklinasi *pitch* = -2.1°), kiri (sudut inklinasi *roll* =

-2.1°) dan kanan (sudut inklinasi *roll* = +2.1°). Penambahan dan pengurangan nilai tersebut didapatkan dengan *trial and error* hingga pergerakan drone cukup untuk melakukan pendaratan. Sumbu y adalah pergerakan maju atau mundur drone, sumbu x adalah kiri dan kanan drone. Nilai keluaran untuk perintah kendalinya adalah pulsa tiap kanal dengan *range* nilai 1000-2000. Drone dalam keadaan diam (*poshold*) ketika nilai pulsa keluarannya adalah nilai tengahnya yaitu 1500. *Range* sudut inklinasi *pitch* dan *roll* ditentukan pada 0°- 30°.

Dengan memanfaatkan lebar piksel, kita dapat memperkirakan ketinggian *drone*. Semakin dekat panjang atau lebar blob yang terdeteksi dapat diasumsikan *drone* semakin dekat dengan landasan. Dan semakin tinggi *drone* terbang blob yang terdeteksi akan semakin kecil. Namun ketika nilai w lebih kecil dari nol, artinya landasan tidak terdeteksi. Maka *drone* diperintahkan untuk menambah ketinggian lagi untuk memperluas pandangan kamera sampai landasan kembali terdeteksi oleh kamera.

Berikut adalah perintah yang akan dilakukan pada masing-masing kondisi:

1. Landasan berada pada kuadran 1

Gambar 3.11 Kendali drone saat landasan pada kuadran 1

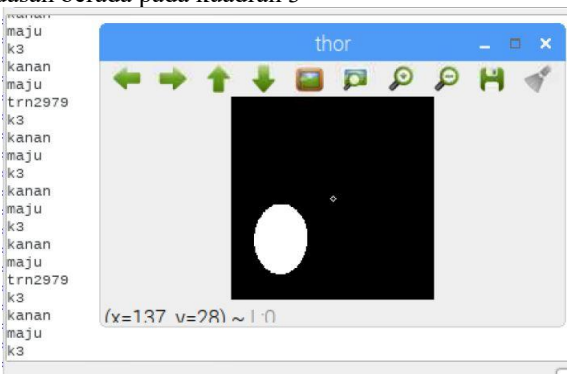
Ketika landasan pendaratan berada di kuadaran 1 seperti gambar 3.11, maka kendali yang dibutuhkan adalah mundur dan kiri. Titik putih pada ditengah adalah asumsi posisi *drone*. Saat drone bergerak mundur maka posisi blob akan bergeser ke bawah, dan ketika drone bergerak ke kiri maka blob akan berpindah ke arah kiri gambar. Setelah bergerak posisi koordinat x, y akan disimpan sebagai referensi titik terakhir. Perintah turun dikeluarkan karena sudah masuk dalam ambang batas, dimana syarat untuk turun adalah jika lebar blob piksel yang terdeteksi lebih besar dari 5 satuan piksel.

2. Landasan berada pada kuadran 2

Gambar 3.12 Kendali drone saat landasan berada pada kuadran 2

Ketika landasan berada pada kuadran 2 seperti gambar 3.12 drone akan mendapatkan perintah dari Arduino untuk mundur dan mengarah ke kanan sampai landasan yang terdeteksi berada di titik tengah. Perintah turun juga dikeluarkan karena blob yang terdekteksi cukup besar dan dalam ambang batas. Pulsa sinyal *pitch* untuk perintah ke mundur adalah 1465 dan perintah mundur nilai pulsa sinyal *roll* yang di keluarkan adalah 1535. Pada kondisi ini drone masi cukup tinggi karena lebar blob masih kurang dari 100.

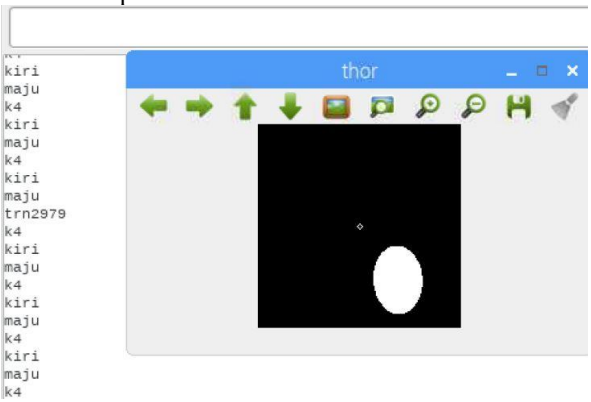
3. Landasan berada pada kuadran 3



Gambar 3.13 Kendali drone saat landasan berapa pada kuadran 3

Ketika landasan yang terdeteksi berada pada kuadran 3 pada gambar 3.13 maka perintah yang akan dilakukan adalah maju dan ke kanan. Kemudian posisi terakhir x, y disimpan kembali. Blob yang terdeteksi masi kurang dari 100 maka drone diperintahkan menengahkan dan turun.

4. Landasan berada pada kuadran 4



Gambar 3.14 Kendali drone saat lintasan berada pada kuadran 4

Ketika landasan terdeteksi berada pada kuadran 4 , perintah yang akan dilakukan adalah drone harus maju dengan mengeluarkan output pusa sinyal *pitch* = 1535 dan bergerak kearah kiri dengan pulsa sinyal *roll* = 1465. Gambar 3.14 adalah pengujian perintah keluaran saat drone berada pada kuadran 4.

5. Landasan tidak terdeteksi

Gambar 3.15 Landasan tidak terdeteksi

Saat drone terbang dan kondisi bergerak namun landasan yang terlewat, maka nilai pembacaan data serial x dan y adalah -1000. Namun data data simpan terakhir berada pada kuadran tertentu. Pada gambar 3.15, kondisi terakhir landasan terdeteksi adalah pada kuadran ke-3, maka drone akan diperintahkan untuk menuju ke kuadran ke-3. Dalam perintah ini Arduino akan menghitung selama 5 detik melakukan perintah kearah

posisi kuadran terdeteksi. Jika nilai xdan y tidak sama dengan -1000 maka nilai penyimpanan koordinat akan diubah dengan nilai yang baru. Berlaku sama dengan kondisi pada keempat kuadran.

6. Landasan terdeteksi dengan lebar lebih dari 100

Karena kamera mengalami perubahan intensitas cahaya saat drone terbang dekat dengan landasan maka, algoritma saat proses drone turun dimanipulasi dengan mengecek ukuran blob landasan. Ketika ukuran lebar dari blob lebih besar atau sama dengan 100, maka drone di perintakan turun tampa menengahkan lagi.

7. Landasan tidak terdeteksi selama 5 detik

Kondisi ketika drone terbang otomatis maka raspberry pi akan mengolah citra digital untuk mendeteksi landasan. Namun ketika tidak ada landasan yang terdeteksi, maka raspberry pi akan tetap mengirimkan nilai -1000. Pada rancangan mikrokontroller Arduino saat drone menerima nilai -1000 dari data serial, maka diasumsikan drone tidak mendeteksi landasan, dan akan diperitnahkan untuk menambah ketinggian untuk memperluas pandangan. Saat drone terbang dan menambah ketinggian namun tetap tidak mendeteksi landasan, untuk keamanan drone di perintahkan untuk mendarat dengan mengurangi *throttole*. hal seperti ini terjadi ketika nilai *threshold* warna yang kurang tepat atau drone terbang terlalu tinggi sehingga landasan sudah tidak terlihat dengan jelas. Maka tidak dapat dideteksi oleh kamera raspberry pi lagi.

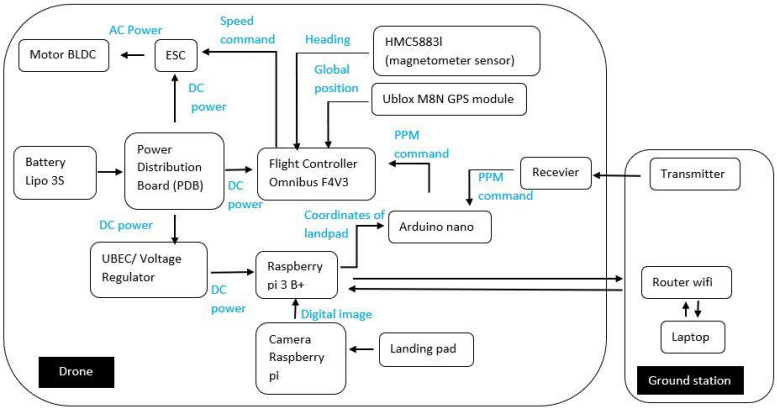
3.3. Flight Controller

Flight Controller (FC) adalah modul yang berfungsi untuk mengedalikan kestabilan drone. FC mendapatkan input dari beberapa sensor yang ada pada drone. Sensor internal yang sudah ada pada perangkat adalah sensor accelerometer, gyroscope, dan magnetometer. Sensor ekseternal untuk mendukung kestabilan drone yaitu magnetometer dan GPS. Dengan bantuan sensor GPS dan magenetometer drone dapat tebang dalam mode *position hold.*

Pada mode pendaratan otomatis FC menerima sinyal ppm berupa perintah *pitch, roll,* dan *throttle*. Sedangkan untuk *yaw* tetap independen tidak dikontrol oleh mikrokontroler. Keluaran dari FC adalah perintah kecepatan untuk ESC pada drone. Dengan perintah tersebut drone akan bergerak menuju titik landasan pendaratan dan mendarat sampai titik yang ditentukan.

3.4. *Drone*

Perancangan *drone* meliputi rancangan mekanik dan rancangan elektronik. Gambar 3.16 adalah rancangan sistem keseluruhan perangkat pada *drone*.



Gambar 3.16 Digram blok perangkat keras drone

Berikut penjelasan masing-masing bagian:

1. Kamera raspberry pi

   Sistem pendaratan yang dirancang adalah berbasis pengolahan citra, maka dibutuhkan sensor kamera. Tugas akhir ini menggunakan modul kamera raspberry pi v1 dengan pertimbangan dimensi *drone* yang kecil dan harga yang murah. Kamera raspberry pi dengan mudah diintegrasikan dengan Raspberry pi melalui konektor CSI yang sudah tersedia. Sensor kamera ini dapat digunakan untuk mengambil gambar dan merekam video. Pada rancangan drone ini Raspberry pi akan menerima *input* gambar landasan pendaratan dari kamera.

2. Raspberry pi 3 B+

   Raspberry pi yang terkoneksi dengan kamera akan mengolah citra digital untuk melakukan *tracking* terhadap landasan pendaratan. Dari pegolahan citra digital tersebut akan didapatkan koordinat relatif landasan yang kemudian dikirim dengan komunikasi serial ke Arduino nano melalui serial port pada raspberry pi.

3. Arduino nano

   Dalam sistem ini, Arduino akan menerima data posisi landasan dari raspberry pi dan sinyal ppm dari *receiver*. Koordinat landasan

pendaratan dalam satuan piksel yang telah diolah oleh raspberry pi akan diterima olah Arduino melalui komunikasi serial. Koordinat tersebut menjadi titik acuan untuk kendali pendaratan pada drone. Selain itu Arduino juga menerima sinyal *pulse position modulation (PPM)* dari *receiver*. Sinyal tersebut digunakan sebagai perintah untuk mengendalikan gerak drone. Untuk pendaratan otomatisnya, sinyal PPM dari receiver akan direkayasa untuk kendali drone menuju koordinat landasan pendaratan yang telah diterima dari raspberry pi. Drone dapat mendarat dengan presisi pada landasan dengan bantuan kendali otomatis dari Arduino.

4. Receiver dan Transmitter

   Pengendalian drone manual adalah menggunakan *radio controller*. Komponen *receiver* berfungsi untuk menerima sinyal dan transmitter sebagai pemancar sinyal. Dalam tugas akhir ini receiver yang digunakan adalah receiver FS-IA6. Receiver ini kompatibel untuk menerima tiga jenis sinyal, yaitu parallel PWM, PPM dan i-Bus.

5. Flight controller Omnibus F4V3

   Flight controller berfungsi untuk mengontrol kestabilan terbang *drone* dengan mengakses sensor-sensor yang sudah terntegrasi pada papan sirkuit FC. Setelah menerima perintah dari Arduino, *flight controller* akan mengolah perintah tersebut dan meneruskannya ke aktuator yaitu motor. FC Omnibus F4V3 memiliki beberapa sensor internal yaitu sensor gyro, accelerometer ditambah satu barometer. Untuk suplai daya FC omnibus F4V3 sudah memiliki regulator internal dengan range *input* daya adalah maksimal 24V sehingga dapat dihubungkan langsung dengan *power distribution board* (PDB)

6. Sensor HMC5883l dan modul GPS

   Untuk memudahkan menjalankan misi pendaratan, *drone* harus terbang dengan mode *position hold* guna menghindari *drifting* yang berlebihan. *Drifting* yang terjadi adalah drone bergeser terus menerus dari posisi awalnya tampa ada perintah masukan. Hal tersebut terjadi karena drone berusaha mempertahankan kestabilan sudutnya. Untuk mengatasi hal tersebut drone ditambahkan sensor kompas dan GPS pada sistem. Sensor tersebut adalah sebagai bantuan untuk navigasi drone sehingga dapat mempertahankan posisi ketika terjadi *drifting*. Sensor eksternal adalah sensor tambahan yang tidak ada pada papan sirkuit *Flight controller*. Dua sensor tambahan pada rancangan sistem adalah sensor ublox m8n modul GPS dan sensor magnetometer HMC5883l. Dua sensor ini akan membantu untuk *drone* bisa

terbang dengan mode *position hold.* Mode *position hold* adalah salah satu mode terbang drone dengan mempertahankan posisi dengan mengandalkan sensor GPS. Mode ini juga sering disebut dengan loiter. FC Omnibus F4V3 juga bisa ditambahkan sensor eksternal lainnya seperti rangefinder atau sensor ketinggian, dan optical flow. Namun untuk firmware yang tersedia belum mendukung untuk menggunakan fitur tersebut.

7. Regulator tegangan / UBEC

   Sumber tengangan yang digunakan adalah baterai lipo 3S. Tiap selnya memiliki tengangan 3.7-4.2 Volt. Sehingga untuk 3 sel memiliki tegangan 11.1 V. Secara umum komponen elektronik beroperasi pada tengan 5V. UBEC (*ultimate battery eliminator circuit)* adalah rangkaian yang befungsi menurunkan nilai tegangan layanyaknya regulator teganagan. Pada rancangan sistem ini UBEC digunakan untuk menyuplai raspberry pi.

8. Laptop

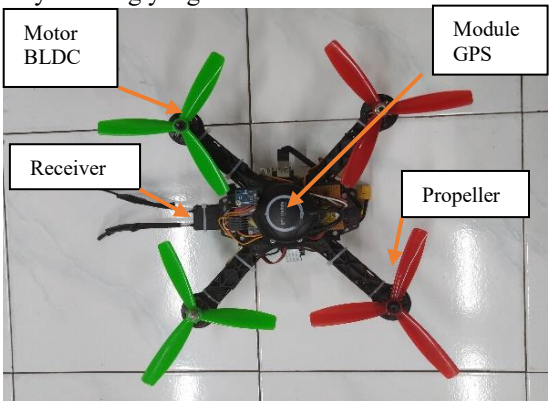
   Untuk memantau proses tracking landasan, laptop atau *personal computer* yang terhubung dengan jaringan wifi yang sama dengan raspberry pi digunakan untuk memantau proses pendaratan. Selain itu pengaturan parameter dan kalibrasi sensor pada *flight controller* menggunakan konfigurator iNav dilakukan sebelum drone uji untuk terbang. iNav merupakan perangkat lunak *open source* yang dikembangkan untuk pengguna beberapa brand *flight controller*, salah satunya adalah Omnibus F4V3. Hal yang ingin dicapai dalam tahap ini adalah kestabilan dari *drone*, dengan mengubah parameter yang sesuai konfigurator iNav.

3.5.1. Rancangan Mekanik

Gambar 3.17 *Frame* FPV 250

Perancangan mekanik dimulai dari penentuan bentuk frame *drone* yang akan digunakan. Pada tugas akhir ini ukuran *frame drone* yang digunakan memilki diameter 25 cm dengan tipe FPV 250 *quadcopter mini small*.

Gambar 3.17 adalah penampakan *frame* yang digunakan pada tugas akhir ini. Selain itu *drone* menggunakan propeller dengan spesifikasi 6045, 3 blades. Alasan menggunakan propeller 3 *blades* adalah untuk mendapatkan daya dorong yang lebih besar.



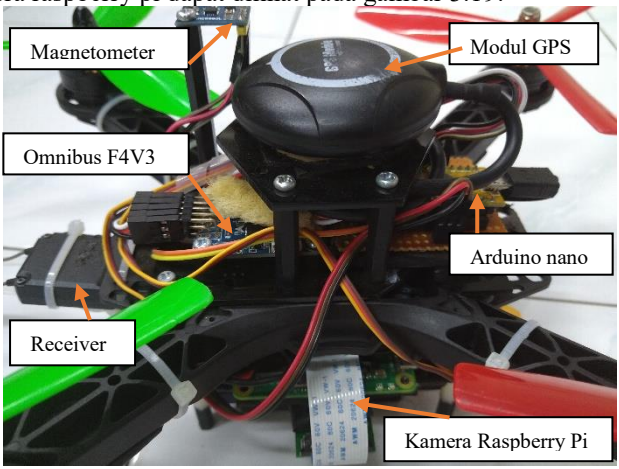
Gambar 3.18 *Drone* tampak atas

Pada bagian mekanik titik berat drone harus bisa sedekat mungkin dengan titik tengah drone. Distribusi berat dari setiap komponen harus sama. Kesalahan penempatan komponen yang menimbulkan berat tidak seimbang akan membuat drone terbang tidak stabil. Masalah yang dihadapi adalah ukuran frame yang kecil sehingga penempatan komponen elektronik tidak memadai, sehingga pada mekanik diberikan tambahan untuk ruang penempatan komponen elektronik. Gambar 3.18 adalah tampilan *drone* tampak atas. Penempatan posisi setiap komponen elektronik dapat dilihat pada gambar. Pada bagian atas dan bawah *drone* diberikan tambahan ruang menggunakan akrilik dengan ketebalan 2mm.

Desain pada gambar tersebut adalah desain final setelah sebelumnya mengalami perubahan. Rancangan awal penempatan komponen berada pada bagian atas frame. Namun rancangan tersebut mengurangi kestabilan karena titik beratnya terlalu jauh dari titik tengah gravitasi *drone*. Sehingga dilakukan perubahan desain dengan memindahkan posisi raspberry dibagian bawah frame. Dengan begitu titik beratnya akan bergeser kebawah yang mendekati titik berat *drone*.

Kamera raspberry pi berada pada kanan *drone*. Kamera dipasang menghadap ke bawah dan tidak terlalu jauh dari bagian tengah *drone*. posisi penempatan kamera akan mempengaruhi perhitungan nilai titik

tengah citra digital yang ditangkap oleh kamera. Posisi penempatan kamera raspberry pi dapat dilihat pada gambar 3.19.

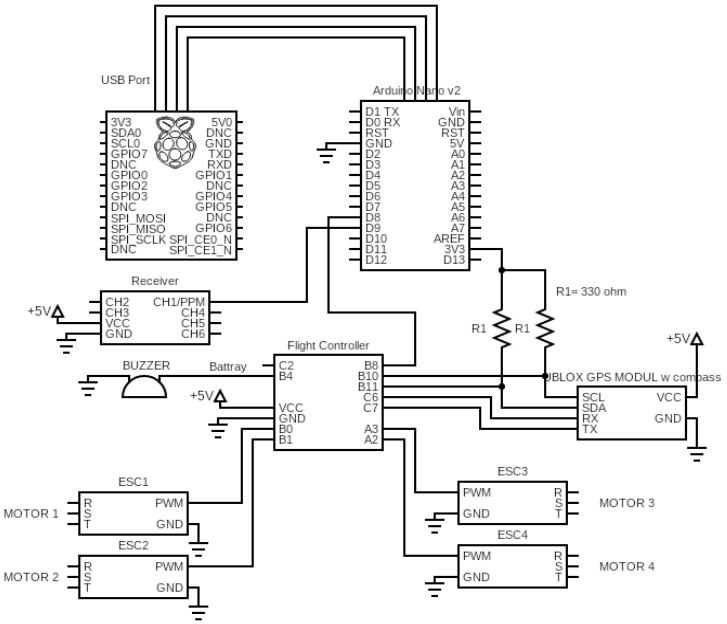


Gambar 3.19 *Drone* tampak samping

3.5.2. Rancangan Elektronik

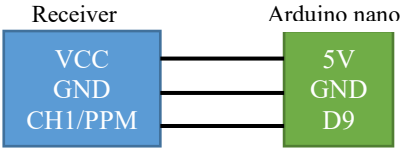
Pada bagian perancangan elektronik *drone,* terdiri dari penentuan suplai tegangan dan distribusinya ke tiap-tiap komponen, serta *wiring* untuk koneksi tiap komponen yang digunakan (lihat gambar 3.20). Baterai yang digunakan untuk menyuplai drone adalah baterain *litium polymer* (LiPo) 3sel dengan kapasistas 1500mAh. Tegangan didistribusikan melalui PDB (*power distribution board).* ESC yang digunakan adalah jenis brushless motor DC BLHeli 12A. Tegangan operasinya adalah 2-4S LiPo sehingga dihubungkan langsung dengan PDB. Motor yang digunakan adalah jenis motor Brushless DC 2204 dengan rating 2300KV. Motor akan mandapat suplai daya AC dari ESC sesuai perintah kecepatan dari *flight controller.*

Suplai yang dibutuhkan raspberry pi sesuai spesifikasi adalah 5V 2.5A. Maka dipilih regulator tegangan/UBEC dengan nilai tegangan keluaran 5V 3A, untuk menyuplai raspberry pi. *Flight controller* Omnibus F4V3 sudah memiliki regulator tegangan sendiri pada papan sirkuitnya, sehingga dapat disuplai langsung melalui baterai. Arduino nano disuplai melalui *port usb* dari raspberry pi.

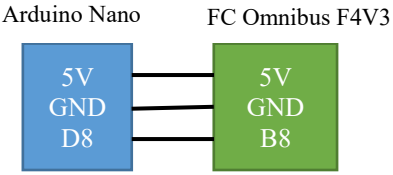


Gambar 3.20 Wiring Diagram drone

Raspberry dihubungakan dengan kamera melalui konektor CSI. Arduino nano terkoneksi melalui port usb, selain sebagai penyuplai daya dari raspberry pi, juga sekaligus untuk komunikasi serial. Pada Arduino nano akan membaca sinyal ppm dari receiver. Sinyal PPM dikirim dengan *single wire*, jadi receiver terhubung dengan satu pin Arduino yaitu pin D9. Gambar 3.21 adalah antarmuka antara *receiver* dengan Arduino nano. Sinyal keluaran PPM hasil diteruskan ke *flight controller* yang terhubung pada pin D8. Gambar 3.21 adalah antarmuka antara Arduino nano dengan *Flight controller* Omnibus F4V3.

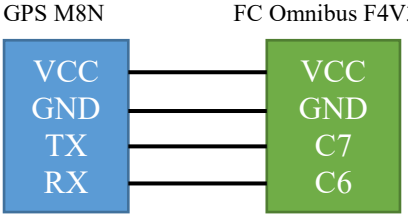


Gambar 3.21 Antarmuka receiver dengan Arduino nano



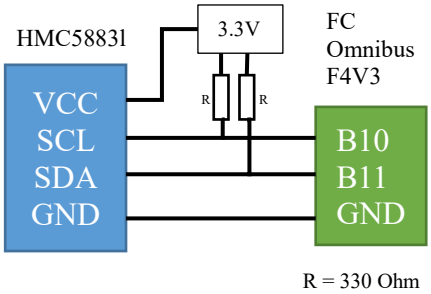
Gambar 3.22 Antarmuka Arduino nano dengan *Flight controller*

Pada bagian *Flight controller,* data sensor GPS akan dibaca melalui UART6 dan untuk magnetometer melalui pin UART3 yang berfungsi juga sebagai pin i2c.



Gambar 3.23 Antarmuka GPS dengan *Flight Controller*

Sensor HMC5883l disuplai dengan tegangan 3.3V. Tegangan suplai dan rangkaian *pullup* untuk sensor magnetometer didapat dari pin 3.3V Arduino nano. Antartarmuka sensor magnetometer dapat dilihat pada gambar 3.24.



Gambar 3.24 Antarmuka HMC5883l dengan *Flight Controller*

Pin PPM pada FC Omnibus F4V3 dihubungakan dengan pin D8 arduino nano yang merupakan output ppm yang telah dimodifikasi untuk kendali otomatis. Supali daya dari PDB terhubunga pada pin C2 atau Vbat. Drone yang digunakan adalah jenis *quadcopter* maka ada empat motor yang menjadi actuatornya. Setiap motor dikontrol menggunakan ESC yang

terhubung pada pin PWM pada FC Omnibus F4V3. Selain itu ditambahkan buzzer sebagai indikator perubahan pada *flight controller.* Buzzer dihubungkan pada pin B4 (buzzer). Tabel 3.1 adalah konfigurasi pin FC Omnibus F4V3 dengan beberapa komponen pada drone.

Secara umum sistemnya dimulai dari mini komputer Raspberry pi akan mengolah citra digital dari kamera untuk mendapatkan posisi realtif drone terhadap ladansan pendaratan. Receiver yang dipasang pada drone berfungsi menerima sinyal dari R/C transmitter. Kemudian sinyal yang diterima oleh receiver akan dibaca oleh arduino nano. Sinyal yang dibaca adalah sinyal analog dengan modulasi PPM. Arduino sudah memiliki library Combined Pulse Position Modulation (CPPM) sehingga mempermudah dalam merancang algoritma pendaratan presisi *drone*. Selain itu arduino juga menerima data dari raspberry pi melalui komunikasi serial. Transfer data antara raspberry dan arduino menggunakan usb serial port pada raspberry. Flight controller akan menerima sinyal ppm dari arduino. Hasil akhirnya adalah sinyal perintah ke esc untuk mengatur kecepatan motor secara auto untuk melakukan pedaratan.

Tabel 3.1 Kofigurasi pin FC Omnibus F4V3

Komponen	Bagian	*Pin Flight Controller Omnibus F4V3*
PDB (*power distribution board)*	Vbat +	C2 (VBat)
	GND	GND
ESC	Motor 1	B0 (PWM1)
	Motor 2	B1 (PWM2)
	Motor 3	A3 (PWM3)
	Motor 4	A2 (PWM4)
Buzzer	Buz+	B4 (Buzzer)
	GND	GND
UBLOX M8N GPS module with compass	VCC	5V
	GND	GND
	TX	C7 (UART RX6)
	RX	C6 (UART TX6)
	SCL	B10 (UART3 TX)
	SDA	B11 (UART3 RX)
Arduino nano	VCC	5V
	GND	GND
	D8	B8 (PPM)

3.5. Fitur Pendukung

Fitur pendukung ini betujuan agar sistem dapat berjalan dengan baik. Beberapa fitur pendukung adalah komunikasi serial antara mini komputer dengan mikrokontroler Arduino nano, dan pembacaan sinyal ppm dari receiver.

3.4.1. Mengirim Data Serial

Setelah mendapatkan nilai x, y, w, h tahap selanjutnya adalah mengirimkan nilai-nilai tersebut ke arduino. Python memiliki library serial yang dipakai pada tugas akhir ini. Pertama nilai-nilai yang akan dikirim di konversikan menjadi data *string.* Sebelum mengirim data terlebih dahulu ditentukan *serial port* penerima dan *baudrate.* Kemudian dengan perintah *serial.write*, nilai-nilai yang dikirim dipisahkan dengan tanda koma sehingga tiap nilai yang akan diterima oleh Arduino dapat dibedakan. Pengiriman diawali dan diakhiri dengan tanda pemisah yaitu koma (‘,’) dengan tujuan data dapat dibedakan saat pembacaan serial oleh arduino. Berikut adalah syntax pengiriman data serial dari raspberry pi:

```
import serial
ser = serial.Serial ('/dev/ttyUSB0',115200)
cent_x = str(cent_x)
cent_y = str(cent_y)
w1 = str (w)
ser.write(',')
ser.write(cent_x)
ser.write(',')
ser.write(cent_y)
ser.write(',')
ser.write(w1)
ser.write(',')
```

3.4.2. Membaca Sinyal Receiver

Sistem pendaratan otomatis drone ini diancang menggunakan Arduino nano sebagai pengontrol kendalinya. Mikrokontroler Arduino nano akan mengontrol perintah arah pergerakan drone. Artinya ketika drone terbang dalam mode otomatis Arduino akan merekayasa sinyal yang dibaca dari receiver. Pembacaan sinyal ppm oleh arduino dengan menggunakan library CPPM dari arduino. Dengan menggunakan library tersebut sinyal ppm dapat dibaca dengan mudah. Untuk mengeluarkan sinyal ppm dari arduino, dapat menggunakan perintah *CPPM.oservos []*.

Syntax pemabacaan sinyal ppm dengan Arduino adalah sebagai berikut:

```
#include CPPM.h
void baca_ppm(){
    if (CPPM.synchronized()){
        roll = CPPM.read(CPPM_AILE) ; // aile
        pitch = CPPM.read(CPPM_ELEV) ; // elevator
        thro = CPPM.read(CPPM_THRO) ; // throttle
        yaw = CPPM.read(CPPM_RUDD) ; // rudder
        ch5 = CPPM.read(CPPM_GEAR) ; // gear
        ch6 = CPPM.read(CPPM_AUX1) ; // flap
        }
      else {
        //if not synch do sth
      }
}
```

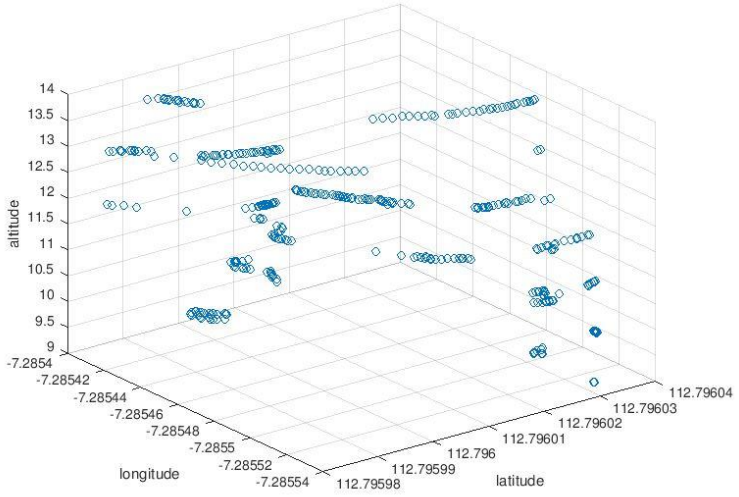
Halaman ini sengaja dikosongkan

BAB 4
PENGUJIAN DAN ANALISIS

Pada bab ini akan dibahas mengenai pengujian dan analisa sistem. Pengujian dilakukan pada setiap bagian rancangan. Pengujian perangkat keras yaitu diamati drone dapat terbang stabil sehingga algoritma dapat diaplikasikan dengan baik. Kemudian pengujian perangkat lunak, dimulai dengan pembacaan sinyal receiver yang akan direkayasa berdasarkan input posisi relatif drone terhadap landasan, kemudian menguji kontrol yang tepat hingga mendarat.

4.1 Pengujian *Flight Mode Drone*

Pada subbab ini dilakukan pengujian terbang drone dalam mode *angle* dan mode *nav poshold*. Mode yang lebih stabil akan sangat membantu dalam proses pendaratan. Mode *angle* adalah salah satu contoh mode *non-navigation*. *Flight mode angle* akan membatasi sudut inklinasi *pitch* dan *roll*. Sedangkan nav poshold adalah salah satu navigation mode yang mempertahankan posisi 3 dimensi. Titik acuan untuk mempertahankan posisinya berdasarkan pembacaan sensor gps.

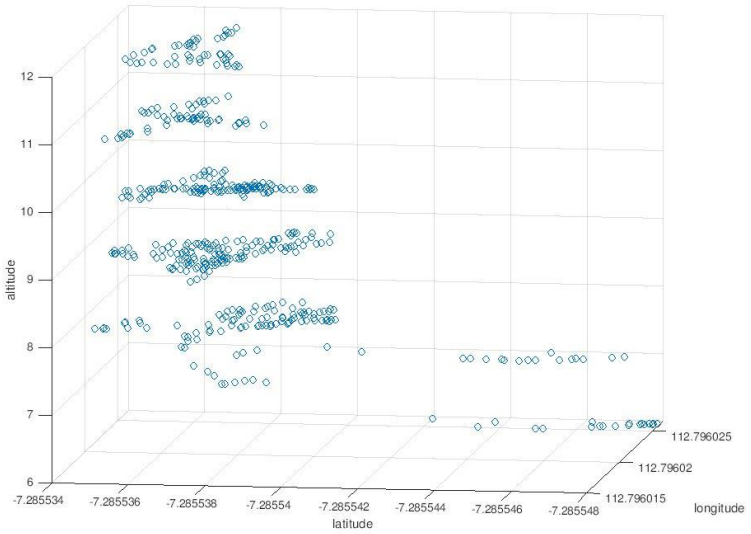


Gambar 4.1 Lintasan drone dalam *flight mode angle*

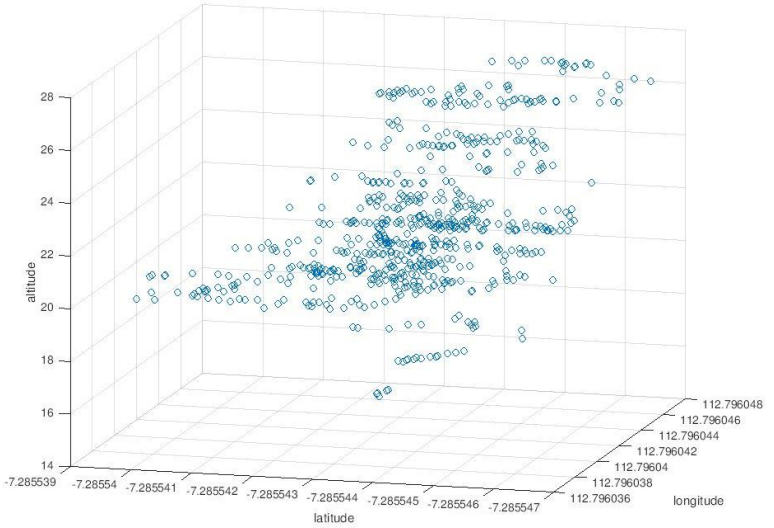
Mengacu gambar 4.1 adalah gambaran lintasan drone dalam saat

flight mode angle. Sumbu x adalah pembacaan nilai longitude atau garis bujur, sumbu y adalah nilai latitude atau garis lintang dan sumbu z adalah nilai altitude atau ketinggian. Nilai-nilai ini didapatkan dari *datalog* dari *blackbox* yang tersedia pada fitur konfigurator pada inav.

Massa drone mempengaruhi kestabilan terbang drone. Setelah dialakukan pengujian, drone lebih stabil saat terbang tampa membawa beban. Pada gambar 4.2 adalah gambaran lintasan drone saat terbang dalam mode *position hold* dengan terbang tampa dipasang raspberry pi. Gambar 4.3 adalah gambar lintasan drone terbang dengan diapasang raspberry pi pada drone. Dari kedua gambar tersebut dapat dilihat perbandingan kestabilan posisi drone saat terbang. Titik-titik plot posisi drone lebih stabil dan mengumpul lebih sempit.



Gambar 4.2 Lintasan drone poshold tampa raspberry pi



Gambar 4.3 Lintasan drone poshold dengan raspberry pi

4.2 Pengujian Pendaratan Menggunakan GPS

Pada subbab ini dilakukan pengujian pendaratan pada drone dengan mengandalkan sensor GPS. Tujuan pengujian ini adalah untuk mengetahui pedandaratan drone tampa menggunakan computer visi sebagai perbandigan dengan rancangan sistem.

Pengujian dilakukan dilapangan futsal Teknik Elektro Institut Teknologi Sepuluh Nopember. Pengujian pendaratan ini dilakukan dengan menambahkan 1 *waypoint* yang akan dituju oleh drone. Setelah mencapai *waypoint* tersebut, *drone* akan melakukan perintah RTH (*return to home*) dan melakukan pendaratan. Misi ini dilakukan berulang-ulang sebanyak 10 kali dengan posisi *home* dan *waypoint* yang sama. Dari tabel 4.1 dapat dilihat berhasil melakukan pendaratan dengan rata-rata mendarat pada jarak 285.4 cm dari titik *home.* Titik *home* dalam pengujian adalah menjadi titik landasan pendaratan. Dengan error minimum mendarat pada jarak titik 26 cm, error maksimum mendarat pada radius lebih dari 5 meter.

Tabel 4.1 Tabel pendaratan dengan GPS

Pengujian	Jarak titik mendarat dari *home* (*error*)
1	305 cm
2	100 cm
3	473 cm
4	422 cm
5	528 cm
6	130 cm
7	368 cm
8	26 cm
9	338 cm
10	164 cm

4.3 Pengujian Keseluruhan Sistem

Pengujian sistem dilakukan dilapangan futsal Teknik Elektro Institut Teknologi Sepuluh Nopember dengan kondisi cahaya yang cukup. Pengujian dilakukan dengan dua tahap. Tahap pertama adalah pengujian dengan mengrahkan drone secara manual mendekati landasan, kemudian *switch* pendaratan diaktifkan, dan pengujian kedua dilakukan dengan memadukan sistem pendaratan berbasis computer visi dengan *waypoint.* Pada pengujian pertama dilakukan seanyak 10 kali percobaan. Rata-rata selisih error yang berhasil adalah 46,4 cm.

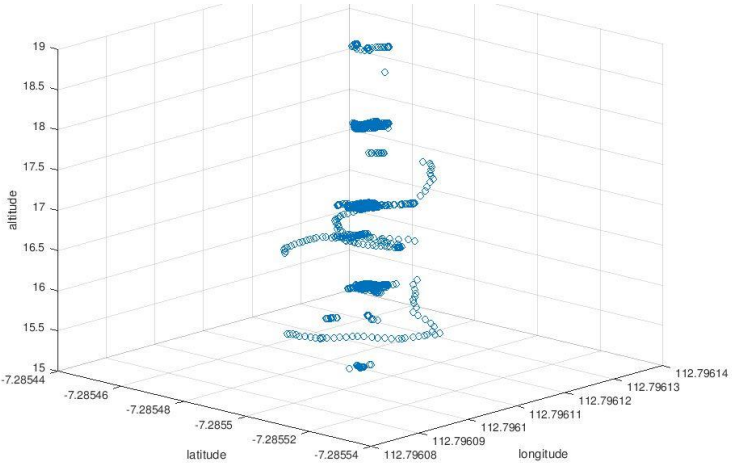
Tabel 4.2 Pengujian sistem landing

Pengujian	Berhasil	Error
1	Ya	120 cm
2	Ya	96 cm
3	Ya	37 cm
4	Tidak	-
5	Ya	20 cm
6	Ya	30 cm
7	Ya	21 cm
8	Ya	25 cm
9	Ya	43 cm
10	Ya	26 cm

Pada percobaan 1 drone terbang otomatis dan mendeteksi landasan. Drone menajalankan perintah menengahkan pendaratan. Namun pada akibat perubahan ketinggian yang terlalu cepat mengakibatkan perubahan

piksel yang terlalu signifikan yang mengakibatakan drone gagal mendeteksi landasan saat lebar blob lebih besar dari 100. Dimana ketika blob lebih besar dari 100, maka maka drone menjalankan perintah turun tampa menengahkan. Sehingga mengalami error hingga 120 cm. Percobaan kedua drone juga berhasil mendeteksi dan mendarat dengan error yang lebih kecil. Pendaratan dan pendeteksian landasan berhasil namun karena terbang ruangan terbuka, adanya gangguan dari angin yang membuat error posisi pendaratan hingga 96 cm. Pada pengujian ketiga berhasil mendarat dengan error yang lebih kecil.

Percobaan keempat gagal mengalami deteksi karena drone tidak dapat terbang dengan stabil. Hal yang mengakibatkan hal tersebut adalah turbulensi dari angina pada ruangan terbuka. Pergeseran posisi drone yang terlalu jauh mengakibatkan drone tidak bisa mendeteksi landasan. Drone terbang selama 5 detik hingga ketinggian lebih dari 6 meter. Sedangkan dari hasil pengujian kamera raspberry pi hanya dapat mengenali landasan pada jarak 5,8 meter.



Gambar 4.4 Lintasan proses pendaratan drone otomatis.

Kestabilan terbang *poshold* menjadi parameter yang sangat penting dalam proses pendartan. Pada pengujian yang ketujuh drone berhasil mendarat dengan error 21cm dari titik pusat. Gambar 4.4 adalah lintasan drone saat melakukan pendaratan ototmatis. Pada bagian titik dengan jumlah yang banyak adalah proses drone menengahkan posisi, kemudian

mengurangi ketinggian secara bertahap. Hingga akhirnya mencapai landasan.

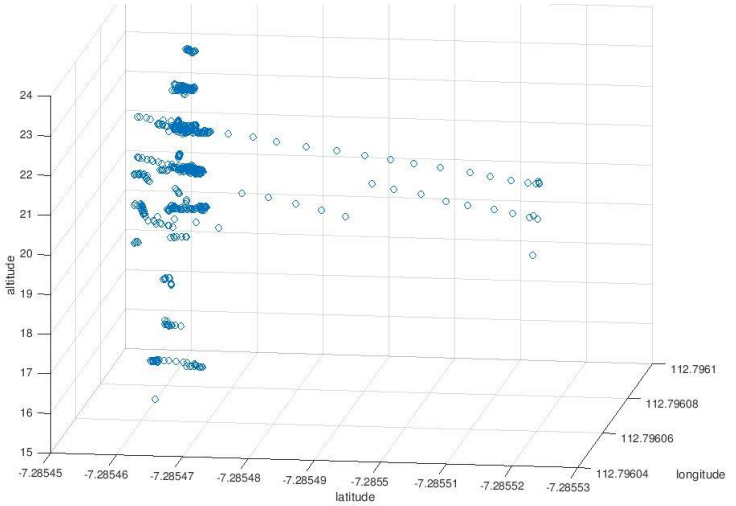
Pada pengujian tahap kedua adalah menggunakan *waypoint* pada drone. Tahapannya adalah drone tebang menuju *waypoint* kemudian melakukan terbang *hover* tampa landing. Setelah itu drone melakukan pendaratan otomatis menggunakan sensor kamera secara otomatis. Dari pengujian yang telah dilakukan drone berhasil mendarat dengan erorr yang lebih kecil dibadingkan dengan pendaratan menggunakan landing dengan rth. Drone berhasil mendarat ketika *hover* drone tidak terlalu jauh dari landasan.

Tabel 4.3 Pengujian pendaratan integrasi *waypoint/*computer visi

Pengujian	Berhasil	Error
1	Ya	137 cm
2	Ya	24 cm
3	Tidak	-
4	Ya	21 cm
5	Tidak	-
6	Ya	53 cm
7	Ya	31 cm
8	Ya	48 cm
9	Ya	21 cm
10	Ya	25 cm

Drone berhasil mendeteksi dan melakukan pendaratan dengan error terkecil 21 cm. Persentasi kesuksesan dari 10 kali percobaan adalah Dengan error rata-rata pendaratan yang berhasil adalah 45 cm. Pada percobaan yang gagal drone tetap mendarat namun tidak mengandalkan pengolahan citra. Hal ini diakibatkan RTH dari drone terlalu jauh dari landasan pendaratan, walaupun drone terbang semikin tinggi landasan tetap tidak terdeteksi. Sehingga drone mendarat otomatis setelah tidak mendeteksi selama waktu yang ditentukan

yaitu 5 detik. Gambar 4.5 dalah lintasan proses pendaratan drone saat melakukan pendaratan otomatis. Dari gambar dapat dilihat drone pertama menuju *waypoint* kemudian terbang *position hold* dan melakukan pendaratan otomatis.



Gambar 4.5 Lintasan drone saat melakukan pendaratan otomatis dengan *waypoint*

4.4 Pendaratan Presisi

Pendaratan presisi tidak didefenisikan secara baku dengan angka tertentu. Setelah melakukan studi literatur dapat disimpulkan bahwa, pendaratan presisi merupakan pendaratan dengan mengandalkan visual untuk memperbaiki akurasi pendaratan. Dengan mengandalkan visual, drone dapat mengetahui posisi landasan pendaratan yang ditentukan. Semakin kecil landasan, semakin baik tingkat presisi pendaratan. Dari beberapa literatur yang telah melakukan penelitan pendaratan presisi, dapat disimpulkan hasil pendaratan memiliki error yang jauh lebih baik dari pendaratan dengan mengandalkan GPS. Pada tugas akhir ini dilakukan pengujian pendaratan mengandalkan GPS, drone mendarat dengan rata-rata error posisi adalah 2.854 m. Dari hasil pengujian pada tugas akhir ini, drone dapat melakukan pendaratan presisi dengan rata-rata error posisi adalah 45 cm dengan minimal posisi landasan pendaratan 21 cm.

Berikut adalah rangkuman penelitian tentang pendaratan presisi.

Tabel 4.4 Rangkuman pendaratan presisi

No	Makalah	Error posisi pendaratan presisi
1	Auto Takeoff and Precision Landing Using Integrated GPS/Optical Flow Solution[16]	Maksimal 0.27 m
2	Precision landing using an adaptive fuzzy multi-sensor data fusionarchitecture[20]	Kurang dari 0.1 m
3	An Autonomous UAV with an Optical Flow Sensor for Positioning and Navigation[18]	0.30 m
4	Drone Precision Landing using Computer Vision[19]	Maksimal 0.05 m
5	Precision Landing of a Quadrotor UAV on a Moving Target Using Low-cost Sensors[5]	Rata-rata 0.38 m
6	Multirotor UAV sensor fusion for precision landing[21]	0.08 m
7	Visually-Guided Landing of an Unmanned Aerial Vehicle[17]	Rata-rata 0.42 m
8	Pendaratan Presisi Pada Drone Berbasis Komputer Visi	Rata-rata 0.45 m

BAB 5
PENUTUP

5.1. Kesimpulan

Beradasarkan hasil pengujian yang telah dilakukan dapat disimpulkan pendaratan presisi pada drone dengan bantuan komputer visi akan memperbaiki akurasi pendaratan drone. Drone terbang menggunakan *flight mode position hold*. Berdasarkan hasil pengujian pada tugas akhir ini drone berhasil melakukan pendaratan berabasis komputer visi dengan rata-rata error posisi adalah 0.46 m dan error minimumnya adalah 0.21 m. Pengujian pendaratan juda dilakukan dengan menggunakan navigasi GPS. Dari hasil yang didapatkan drone mendarat dengan rata-rata error posisi 2.854 m dan minimum error posisinya adalah 0.26 m. Pengujian yang sama seperti sebelumnya tetapi dengan menambahkan bantuan visual. Dari hasil pengujian dapat disimpulkan drone dapat melakukan pendaratan presisi dengan error posisi rata-rata 0.45 m dan nilai minimumnya adalah 0.21 m. Kontrol untuk menegahkan posisi drone selama proses pendaratan, menggunakan kontrol dengan kecepatan tetap pada channel *pitch*, *roll* dan *throttle*. Sudut inklinasi *pitch* dan *roll* untuk pergerakan drone adalah -2.1° dan +2.1°, atau sama dengan menambah atau mengurangi 7% pulsa tiap channelnya. Sedangkan *thorottle* secara konstan akan berkurang 2.5% tiap 2 detik selama pendaratan.

5.2. Saran

Pada penelitian selanjutnya diharapkan pendaratan berbasis pengolahan citra menggunakan kamera yang tidak mudah terpengaruh oleh perubahan warna dalam merekam gambar sehingga proses pendeteksian landasan lebih stabil. Selain itu drone yang digunakan tidak melebihi daya angkat sesuai spesifikasi motor dan propeller, agar dapat terbang dengan stabil. Kelebihan beban akan membuat drone *drifting, sehingga* mempersulit kendali drone menuju landasan pendaratan.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] C. N. Hamdani, R. E. A. K, and E. Iskandar, "Perancangan Autonomous Landing pada Quadcopter Menggunakan Behavior - Based Intelligent Fuzzy Control," vol. 2, no. 2, 2013.

[2] W. Bai, F. Pan, B. Xing, C. Pan, and M. Pei, "Visual Landing System Of UAV Based On ADRC," pp. 7509–7514, 2017.

[3] T. K. Venugopalan, "Autonomous Landing of an Unmanned Aerial Vehicle on an Autonomous Marine Vehicle," 2012.

[4] R. Kharisman, K. Astrowulan, and E. Ak, "Perancangan Sistem Navigasi Menggunakan Kamera pada Quadcopter untuk Estimasi Posisi dengan Metode Neural Network," vol. 3, no. 1, pp. 1–6, 2014.

[5] K. Ling, "Precision Landing of a Quadrotor UAV on a Moving Target Using Low-cost Sensors," University of Waterloo, Ontario, Canada.

[6] A. Yu, "Drones for Deliveries."

[7] L. Jinwo and J. Hosang, "Drone Delivery Scheduling Simulations Focusing on Charging Speed, Weight And Battery Capacity :Case of Remote Islands in South Korea," in *Proceedings of the 2017 winter simulation conference*, 2017, pp. 4220–4227.

[8] A. W. Sudbury and E. B. Hutchinson, "A COST ANALYSIS OF AMAZON PRIME AIR (DRONE DELIVERY)," vol. 16, no. 1, pp. 1–12, 2016.

[9] D. Domingos, G. Camargo, and F. Gomide, "Autonomous Fuzzy Control and Navigation of Quadcopters," *IFAC-Pap.*, vol. 49, no. 5, pp. 73–78, 2016.

[10] R. Mahtani and A. Ollero, "Control and Stability Analysis of quadcopter," in *International Conference on Computing, Mathematics and Engineering Technologies – iCoMET*, 2018, pp. 2–7.

[11] S. Huh and D. Hyunchul, "A Vision-Based Automatic Landing Method for Fixed-Wing UAVs," pp. 217–231, 2010.

[12] M. F. Sani and G. Karimian, "Automatic Navigation and Landing of an Indoor AR . Drone Quadrotor Using ArUco Marker and Inertial Sensors," pp. 102–107, 2017.

[13] D. Honegger, L. Meier, P. Tanskanen, M. Pollefeys, and Z. Eth, "An Open Source and Open Hardware Embedded Metric Optical Flow CMOS Camera for Indoor and Outdoor Applications."

[14] S. K. Phang, J. J. Ong, R. T. C. Yeo, B. M. Chen, and T. H. Lee, "Autonomous Mini-UAV for Indoor Flight with Embedded On-board Vision Processing as Navigation System," pp. 722–727, 2010.

[15] M. Engineering, “Lidar-guided Autonomous Landing of an Aerial Vehicle on a Ground Vehicle,” pp. 228–231, 2017.
[16] M. K. S. Al-Sharman, “AUTO TAKEOFF AND PRECISION LANDING USING INTEGRATED GPS/INS/OPTICAL FLOW SOLUTION.”
[17] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, “Visually guided landing of an unmanned aerial vehicle,” *IEEE Trans. Robot. Autom.*, vol. 19, no. 3, pp. 371–380, Jun. 2003.
[18] N. Gageik, M. Strohmeier, and S. Montenegro, “An Autonomous UAV with an Optical Flow Sensor for Positioning and Navigation,” *Int. J. Adv. Robot. Syst.*, vol. 10, no. 10, p. 341, Oct. 2013.
[19] L. Goeller, “Drone Precision Landing using Computer Vision.”
[20] M. K. Al-Sharman, B. J. Emran, M. A. Jaradat, H. Najjaran, R. Al-Husari, and Y. Zweiri, “Precision landing using an adaptive fuzzy multi-sensor data fusion architecture,” *Appl. Soft Comput.*, vol. 69, pp. 149–164, Aug. 2018.
[21] M. Grzes, M. Slowik, and Z. Gosiewski, “Multirotor UAV sensor fusion for precision landing,” *Aircr. Eng. Aerosp. Technol.*, vol. 91, no. 2, pp. 241–248, Feb. 2019.
[22] “BAB II Unmanned Aerial Vehicle (UAV).” [Online]. Available: http://digilib.unila.ac.id/5709/10/BAB II.pdf. [Accessed: 03-Jul-2019].
[23] A. Kaehler and G. Bradski, *Learning OpenCV 3 COMPUTER VISION IN C++ WITH THE OPENCV LIBRARY*, First Edit. United States of America: O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. O’Reilly, 2016.
[24] M. Syuhada, “Realisasi pengenalan plat nomor kendaraan dengan metode histogram citra dan jaringan syaraf tiruan backpropagation,” Lampung, 2015.
[25] A. M. Utama, D. Purwanto, and R. Mardiyanto, “Rancang Bangun Sistem Pengukuran Posisi Target dengan Kamera Stereo untuk Pengarah Senjata Otomatis,” *J. Tek. ITS*, vol. 5, no. 2, pp. 216–220, 2016.
[26] P. Joshi, *OpenCV with Python By Example*. Birmingham: Packt Publishing Ltd., 2015.
[27] K. Nikolskaia, N. Ezhova, and A. Sinkov, “Skin Detection Technique Based on HSV Color Model and SLIC Segmentation Method,” vol. 211, 2018.
[28] B. Y. Budi Putranto, W. Hapsari, and K. Wijana, “Segmentasi Warna

Citra Dengan Deteksi Warna Hsv Untuk Mendeteksi Objek," *J. Inform.*, vol. 6, no. 2, 2011.
[29] M. Computing and L. Alsadoon, "A Hybrid Communication System Using Pulse Position and Width Modulation," *Int. J. Comput. Sci. Mob. Comput.*, vol. 8, no. 3, pp. 1–13, 2019.
[30] C. Science and S. Engineering, "A Review on Serial Communication by UART," vol. 3, no. 1, pp. 366–369, 2013.
[31] V. Flori, "OMNIBUS F4 V3 – Documentation," vol. 64. 2017.
[32] R. P. M. B, "Raspberry Pi 3 Model B + datasheet.".
[33] BAUDAELETRONICA, "Arduino Nano," *Arduino Nano R3*. 2019.

Halaman ini sengaja dikosongkan

LAMPIRAN A

Program pengolahan citra untuk mendeteksi landasan

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2
import time
import serial
import numpy as np

ser =    serial.Serial('/dev/ttyUSB0',115200)
def    mencari_contour (frame):
# define range of    color in HSV
      upper_hsv1 = np.array([145,255,255])
      lower_hsv1 = np.array([61,90,80])

      upper_hsv2 = np.array([194,255,255])
      lower_hsv2 = np.array([121,63,82])

      mask1 = cv2.inRange(frame, lower_hsv1, upper_hsv1)
      mask2 = cv2.inRange(frame, lower_hsv2, upper_hsv2)
      _,contours, hirearchy = cv2.findContours(mask1,cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE,offset=(0, 0))
      for cnt in contours:
            cv2.drawContours(mask1,[cnt],-1,(255, 255, 255),-1)
      _,contours, hirearchy = cv2.findContours(mask2,cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE,offset=(0, 0))
      for cnt in contours:
            cv2.drawContours(mask2,[cnt],-1,(255, 255, 255),-1)
      mask2 = mask2&mask1
      cv2.erode(mask2,(3,3))
      cv2.dilate(mask2,(5,5))
      #cv2.imshow('mask1',mask1)
      #cv2.imshow('mask2',mask2)
      #mask2 = cv2.morphologyEx(mask2, cv2.MORPH_CLOSE, (3,3))
      return mask2

def titik_tengah (threshold):
      threshol= threshold
```

```
	_,contours, hirearchy =
cv2.findContours(threshol,cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE,offset=(0, 0))
	cent_rectx = -1000
	cent_recty = -1000
	w1 = -1000
	for cnt in contours:
		areaK = cv2.contourArea(cnt)

		(x,y,w,h) = cv2.boundingRect(cnt)
		if(areaK>100) :
			cv2.drawContours(threshol,[cnt],-1,(255, 255, 255),-
1)
			#cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
			cent_rectx = (x+(w/2))
			cent_recty = (y+(h/2))
			w1=w
			#cv2.imshow("thr", threshol)
	return cent_rectx,cent_recty,w1
def kirim (x,y,w):
	x = str(x)
	y = str(y)
	w1 = str (w)
	#print	("x = " +str(x) + " y=" + str(y)+ " w= " + str(w1))

	ser.write(',')
	ser.write(x)
	ser.write(',')
	ser.write(y)
	ser.write(',')
	ser.write(w1)
	ser.write(',')
centerx = 80
centery = 80

camera = PiCamera()
#camera.iso = 200
time.sleep(2)
#camera.shutter_speed= camera.exposure_speed
```

```
#camera.exposure_mode = 'off'
#g=camera.awb_gains
camera.awb_mode='fluorescent'
#camera.awb_gains=g
camera.resolution = (640, 480)
camera.framerate = 20
rawCapture = PiRGBArray(camera, size=(640,480))

fourcc = cv2.VideoWriter_fourcc (*'XVID')
out = cv2.VideoWriter ('23junsiang5.avi', fourcc,20.0, (160,160))

for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):

	frame = frame.array
	frame= cv2.resize(frame,(160,160))
	cv2.circle(frame,(centerx,centery),2,(255,255,255),1) # gambar
titik tengah
	hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
	threshold = mencari_contour(hsv)
	#threshold= cv2.dilate(threshold,(1,1))
	x,y,w=titik_tengah(threshold)
	kirim (x,y,w)
	out.write(frame)
	#cv2.imshow("frame", frame)
	cv2.imshow("thor", threshold)
	key = cv2.waitKey(1)

	rawCapture.truncate(0)
	if key == 27:
		camera.close()
		cv2.destroyAllWindows()
		break
```

Halaman ini sengaja dikosongkan

LAMPIRAN B

Program kendali pada Arduino nano

```
#include <CPPM.h>

//ppm input pin 8
//ppm ouput pin 9

#define nilai_tengah 3004
#define nilai_min 2000
#define nilai_max 4000

int nilai_atas =3070;
int nilai_bawah = 2930;

bool turun = false;
int x,w,y;
int simpan_x,simpan_y,simpan_w;
volatile int pitch,roll,yaw,thro,ch5,ch6,thro_last =
nilai_tengah,roll_simpan = nilai_tengah ,pitch_simpan = nilai_tengah;
int last_datax = 0;
int last_datay = 0;
double set_point = 80;
long int time_prev = 0, waktu_prev_roll = 0, waktu_prev_pitch =0
,waktu_prev_thro =0,waktu_prev_th=0 ,waktu_prev_thro_ut=0;
long int i;
long int a=0;

void setup (){
   Serial.begin(115200);
   CPPM.begin();
}

void baca_ppm(){
   if (CPPM.synchronized()){
         roll = CPPM.read(CPPM_AILE) ; // aile
         pitch = CPPM.read(CPPM_ELEV) ; // elevator
         thro = CPPM.read(CPPM_THRO) ; // throttle
         yaw = CPPM.read(CPPM_RUDD) ; // rudder
```

```
            ch5 = CPPM.read(CPPM_GEAR) ; // gear
            ch6 = CPPM.read(CPPM_AUX1) ; // flap
          }
      else {
             //if not synch do sth
      }
}

void cppm_cycle(void){

// cek mode auto
// mode landing mati maka sinyal diteruskan

if (ch5 <= nilai_tengah){
       CPPM.oservos[0]=roll;
       CPPM.oservos[1]=pitch;
       CPPM.oservos[2]=thro;
       CPPM.oservos[3]=yaw;
       CPPM.oservos[4]=ch5;
       CPPM.oservos[5]=ch6;

       thro_last=nilai_tengah;
       a= 1;
       //Serial.println ("m");

    }

else {
       CPPM.oservos[3]=yaw;
       CPPM.oservos[4]=ch5;
       CPPM.oservos[5]=ch6;

       if (Serial.available()){
             x = Serial.parseInt();
             y = Serial.parseInt();
             w = Serial.parseInt();

             long int time_now = millis ();
             if ((time_now - time_prev) >=50){
```

```
                time_prev = time_now;
// menengahkan drone
                if (x>=0 && y>=0) {
                    i = 0;
                    //kuadran 1    landasan di pojok kanan    atas
                    if (w>100){
                        while (a==0){
                            long int waktu_thro_ut = millis ();
                            if ((waktu_thro_ut -
waktu_prev_thro_ut) >=2000){

                                waktu_prev_thro_ut =
waktu_thro_ut;
                                if (w >= 1 ){
                                   thro_last -= 70;
                                   if (thro_last<=nilai_min){
                                        thro_last = nilai_min;
                                   }
                                   CPPM.oservos[0]= nilai_tengah; //
kanan
                                   CPPM.oservos[1]= nilai_tengah;
                                   CPPM.oservos[2]=thro_last;

                                   Serial.print ("TRN");
                                   Serial.println ( thro_last);
                                }
                            }
                        }
                    }else if (x>=80 && y <80) {
                        CPPM.oservos[0]= nilai_atas; // kanan
                        CPPM.oservos[1]= nilai_bawah; // mundur
                        Serial.println("k1");
                    }
                    //kuadran 2 landasan di pojok kiri atas
                    else if (x<80 && y<80) {
                        CPPM.oservos[0]= nilai_atas;//kanan
                        CPPM.oservos[1]= nilai_atas;//maju
                        Serial.println("k2");
                    }
```

```
        //kuadran 3 landasan di pojok kiri bawah
        else if (x<80 && y>=80) {
            CPPM.oservos[0]= nilai_bawah;//kiri
            CPPM.oservos[1]= nilai_atas;//maju
            Serial.println("k3");
        }
        // kuadran 4 landasan di pojok kanan bawah
        else if (x>=80 && y>=80) {
            CPPM.oservos[0]= nilai_bawah;//kiri
            CPPM.oservos[1]= nilai_bawah;//mundur
            Serial.println("k4");
        }
    simpan_x = x;
    simpan_y = y;
}

    else if ( y == -1000 && x ==-1000){
        i++;
        long int waktu_now_roll = millis ();
        if ((waktu_now_roll - waktu_prev_roll) >=500){
          waktu_prev_roll = waktu_now_roll;
          if (simpan_x>0 && simpan_y>0) {
            //kuadran 1
            if (simpan_x>=80 && simpan_y <80) {
                CPPM.oservos[0]= nilai_atas; // kanan
                CPPM.oservos[1]= nilai_bawah; // mundur
                Serial.println("k_1");
            }
            //kuadran 2
            else if (simpan_x<80 && simpan_y<80) {
                CPPM.oservos[0]= nilai_atas;//kanan
                CPPM.oservos[1]= nilai_atas;//maju
                Serial.println("k_2");
            }
            //kuadran 3
            else if (simpan_x<80 && simpan_y>=80) {
                CPPM.oservos[0]= nilai_bawah;//kiri
                CPPM.oservos[1]= nilai_atas;//maju
                Serial.println("k_3");
```

```
                    }
                    // kuadran 4
                    else if (simpan_x>=80 && simpan_y>=80) {
                        CPPM.oservos[0]= nilai_bawah;//kiri
                        CPPM.oservos[1]= nilai_bawah;//mundur
                        Serial.println("k_4");
                    }
                }

                else{
                    CPPM.oservos[0]= nilai_tengah;
                    CPPM.oservos[1]= nilai_tengah;
                    Serial.println("diam");
                }
            }
        }
    }

// proses turun
    long int waktu_now_thro = millis ();
    if ((waktu_now_thro - waktu_prev_thro) >=2000){

        waktu_prev_thro = waktu_now_thro;
        if (w >= 1 ){
            thro_last -= 40;
            if (thro_last<= nilai_min){
                thro_last = nilai_min;
            }
            CPPM.oservos[2]=thro_last;
            Serial.print ("trn");
            Serial.println ( thro_last);
            simpan_w = w;

        }
        else {
            if (i>160){
                long int waktu_now_th = millis ();
                if ((waktu_now_th - waktu_prev_th) >=500){
                    waktu_prev_th = waktu_now_th;
```

```
                    thro_last -=30;
                    if (thro_last<= nilai_min){
                        thro_last = nilai_min;
                    }
                    CPPM.oservos[2]=thro_last;
                    Serial.print ("b");
                    Serial.println (thro_last);
                }
            }
            else{
            thro_last += 50;
            if (thro_last>= 3300){
                thro_last = 3300;
            }
            CPPM.oservos[2]=thro_last;
            Serial.print ("nk");
            Serial.println (thro_last);
            Serial.print (i);
            }
           }
        }
    }

    else {
        CPPM.oservos[0]=roll;
        CPPM.oservos[1]=pitch;
        CPPM.oservos[2]=thro;
        //Serial.println("tdksrl");
    }
  }
}

void loop() {
  baca_ppm();
  cppm_cycle();

}
```

BIODATA PENULIS

Petra Matatias Situmorang, lahir di Belegen ,27 Maret 1997. Penulis menyelesaikan Pendidikan dasar di SD030289 Panji. Kemudian melanjutkan Pendidikan tingkat menegnah di SMPN3 Sidikalang dan SMAN2 Balige. Pada tahun 2015 melanjutkan kuliah di departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah penulis aktif sebagai anggota tim robot terbang Bayucaraka ITS dalam divisi VTOL (Soero Miber), asisten laboratorium, dan beberapa kepanitiaan di kampus.