TUGAS AKHIR – EE 184801

RANCANG BANGUN SISTEM PENJEJAKAN GARIS BERBASIS VISI KOMPUTER PADA *INDOOR PATROLLING DRONE*

Muhammad Abizhar Multazam
NRP 07111540000059

Dosen Pembimbing
Ronny Mardiyanto, ST., MT., Ph.D.
Astria Nur Irfansyah, ST., M.Eng., Ph.D.

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

TUGAS AKHIR – EE 184801

RANCANG BANGUN SISTEM PENJEJAKAN GARIS BERBASIS VISI KOMPUTER PADA *INDOOR PATROLLING DRONE*

Muhammad Abizhar Multazam
NRP 07111540000059

Dosen Pembimbing
Ronny Mardiyanto, ST., MT., Ph.D.
Astria Nur Irfansyah, ST., M.Eng., Ph.D.

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

FINAL PROJECT – EE 184801

DESIGN AND REALIZATION OF COMPUTER VISION BASED LINE FOLLOWER SYSTEM IN INDOOR PATROLLING DRONE

Muhammad Abizhar Multazam
NRP 07111540000059

Supervisor
Ronny Mardiyanto, ST., MT., Ph.D.
Astria Nur Irfansyah, ST., M.Eng., Ph.D.

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

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Rancang Bangun Sistem Penjejakan Garis Berbasis Visi Komputer pada *Indoor Patrolling Drone***" 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, Juli 2019

Muhammad Abizhar Multazam
NRP. 0711 15 4000 0059

RANCANG BANGUN SISTEM PENJEJAKAN GARIS BERBASIS VISI KOMPUTER PADA *INDOOR PATROLLING DRONE*

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
Ronny Mardiyanto, ST., MT., Ph.D.	Astria Nur Irfansyah, ST., M.Eng., Ph.D.
NIP. 198101182003121003	NIP. 198103252010121002

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

SURABAYA
JULI, 2019

RANCANG BANGUN SISTEM PENJEJAKAN GARIS BERBASIS VISI KOMPUTER PADA *INDOOR PATROLLING DRONE*

Nama : Muhammad Abizhar Multazam
Pembimbing : Ronny Mardiyanto, ST., MT., Ph.D.
Astria Nur Irfansyah, ST., M.Eng., Ph.D.

ABSTRAK

Indoor Patrolling drone adalah *drone* yang beroprasi di dalam ruangan dan dapat berpatroli pada area yang ditentukan. *Patrolling drone* mampu melakukan navigasi di dalam ruangan tanpa bergantung pada GPS. Kemampuan *patrolling drone* untuk tidak bergantung pada GPS sangat diperlukan karena ketika *drone* berada di dalam ruangan, sistem navigasi menggunakan GPS tidak dapat dilakukan secara optimal. *Indoor patrolling drone* memadukan *quadcopter drone* dengan sistem navigasi penjejakan garis. Penjejakan garis dilakukan untuk menentukan rute pengawasan yang akan dilalui oleh *drone*. *Patrolling drone* diperlukan karena hingga saat ini sebagian besar metode pengawasan dalam ruangan masih dilakukan oleh agen manusia yang melakukan patroli pada waktu dan area yang ditentukan. Metode tersebut masih rentan akan adanya kesalahan yang ditimbulakan oleh manusia atau sering disebut dengan *human error*. *Patrolling drone* diharapkan dapat menjadi alternatif dan meningkatkan sistem kemanan tersebut. *Patrolling drone* yang dimaksud pada tugas akhir ini telah berhasil dibuat dan telah dilakukan berbagai pengujian. Berdasarkan hasil pengujian tersebut, *patrolling drone* memiliki rata – rata durasi terbang selama 6 menit 56 detik. Sistem penjejakan garis yang digunakan pada drone ini memiliki akurasi estimasi jarak sebesar 0.51cm dan akurasi kecepatan sebesar 15.66cm/s. Dengan membandingkan dengan performa sistem navigasi dalam ruangan menggunakan GPS yang memiliki akurasi sebesar 4.5 hingga 8 meter, sistem navigasi penjejakan garis pada drone ini menunjukkan peningkatan tingkat akurasi yang cukup signifikan. Selain itu, sistem navigasi penjejakan garis pada *patrolling drone* ini dapat dilakukan dengan optimal hingga kecepatan 48cm/s.

Kata kunci : *Indoor patrolling drone,* visi komputer, Sistem penjejakan garis.

Halaman ini sengaja dikosongkan

DESIGN AND REALIZATION OF COMPUTER VISION BASED LINE FOLLOWER SYSTEM IN INDOOR PATROLLING DRONE

Name : Muhammad Abizhar Multazam
Supervisor : Ronny Mardiyanto, ST., MT., Ph.D.
Astria Nur Irfansyah, ST., M.Eng., Ph.D.

ABSTRACT

Indoor patrolling drone is a drone which operate indoor and can automatically patrol in the designated area. Patrolling drone is capable of indoor navigation without relying on GPS system. The ability of indoor patrolling drone to navigate without relying on GPS system is very important, because any form of GPS basednavigation system can not be used effectively in indoor applications. Indoor patrolling drone combines quadcopter drone and line follower navigation system. Line follower system is used to determine drone's surveillance route. Self-patrolling drone is needed because untill now, most of the surveillance method are still done by human. Human surveillance method are prone to error caused by human or frequently labeled as human error. Self-patrolling drone is expected to be an alternative and to improve conventional surveillance method that involve human in their system. Self-patrolling drone described in this final project have been successfully made and has undergone several test and observation. Based on the test and observation result, self-patrolling drone has 6 minutes and 56 seconds flight time. Line follower system used in this project has distance estimation with 0.51cm accuracy and cruise control with 15.66cm/s accuracy, in contrast with GPS based indoor navigation system that has 4.5-8m accuracy, line follower navigation system shows quite significant improvement in terms of accuracy. Moreover, line follower navigation in this project can be implemented effectively up to 48cm/s cruising speed.

Keywords: Drone, self-patrolling drone, image processing, *line follower system*.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Dengan menyebut rasa syukur kepada Allah SWT dengan segala rahmat dan karunia-Nya, penulis dapat menyelesaikan tugas akhir ini yang berjudul "Rancang Bangun Sistem Penjejakan Garis Berbasis Visi Komputer pada *Indoor Patrolling Drone*" yang dikerjakan untuk memenuhi persyaratan kelulusan Strata 1 di Departemen Teknik Elektro, Fakultas Teknik Elektro, Institut Teknologi Sepuluh Nopember.

Dalam pengerjaan Tugas Akhir ini, penulis akan mengucapkan terima kasih kepada beberapa pihak karena sebagai halnya manusia tidak dapat bekerja dengan sendiri. Maka dari itu, penulis akan mengucapkan banyak terima kasih kepada :

1. Orang tua, yang telah memberikan banyak dukungan baik berupa dukungan moral maupun materi.
2. Bapak Dr. Eng Ardyono Priyadi, ST, M.Eng. selaku kepala departemen teknik elektro ITS.
3. Bapak Ronny Mardiyanto, ST., MT., Ph.D dan Bapak Astria Nur Irfansyah, ST., M.Eng., Ph.D. selaku dosen pembimbing satu dan dosen pembimbing dua.

Dalam pengerjaan tugas akhir ini, penulis banyak menghadapi halangan serta tantangan. Namun, penulis tetap berusaha untuk dapat menyelesaikannya dalam kurun waktu yang ditetapkan. Penulis menyadari bahwa tugas akhir yang telah dibuat masih memiliki banyak kekurangan dan penulis berharap untuk pembaca dapat memberikan masukan serta saran. Masukan dan saran yang diberikan akan memberikan kesempatan untuk tugas akhir ini berkembang lebih baik lagi. Penulis juga berharap tugas akhir ini dapat memberikan manfaat untuk banyak orang kedepannya. Sekian dari penulis dan mohon maaf apabila memiliki kesalahan dalam penulisan maupun bentuk lainnya serta terima kasih untuk pembaca atas waktunya membaca tugas akhir ini.

Surabaya, Juli 2019

Muhammad Abizhar multazam

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Sampai saat ini sebagian besar sistem keamanan dalam ruangan masih banyak dilakukan dengan metode manual yaitu dengan adanya petugas keamanan yang berpatroli pada area pengawasan. Sistem keamanan tersebut masih banyak ditemui pada bangunan - bangunan berskala besar seperti gudang penyimpanan, gedung olah raga dan lain – lain. Sistem tersebut dirasa kurang efektif karena keterbatasan manusia sebagai pengawas keamanan[1]. Manusia rentan dengan kelalaian yang dapat berdampak pada keamanan area pengawasan. Oleh karena itu, pemanfaatan drone untuk mendukung atau menggantikan peran manusia pada sistem keamanan dalam ruangan dianggap penting untuk diimplementasikan.

Pemilihan *drone* untuk mendukung atau menggantikan peran manusia pada sistem keamanan dalam ruangan memiliki beberapa alasan. Dalam beberapa tahun terakhir, perkembangan *Unmanned Air Vehicle* (UAV) memiliki banyak potensi yang dapat dikembangkan. *Drone* atau UAV dapat dimanfaatkan untuk dapat menggantikan peran manusia dalam melakukan berbagai macam pekerjaan. Dalam beberapa penelitian, drone sudah dapat diaplikasikan untuk melakukan inspeksi jalur transmisi tegangan tinggi[2], pencitraan radar[3], penyemprot pestisida[4] dan banyak lainya[5]–[11].

Pada tugas akhir ini, penulis berusaha rancang bagun sistem penjejakan garis berbasis visi komputer pada *indoor patrolling drone*. Metode penjejakan garis digunakan untuk menentukan rute pengawasan yang akan dilalui oleh drone. Metode penjejakan garis sudah banyak diimplementasikan pada berbagai macam robot, salah satunya terdapat pada sistem penyimpanan objek[12]. Dalam berbagai aplikasi, sistem *line follower* digunakan karena memiliki kompleksitas yang rendah dan keandalan yang tinggi. Pada robot – robot yang menggunakan sistem navigasi *line follower*, robot dapat melakukan navigasi secara otomatis dengan rute yang ditentukan berdasarkan garis yang telah dipersiapkan.

Pada tugas akhir ini, sistem penjejakan garis dilakukan dengan menggunakan metode pengindraan visual komputer. Pada umumnya metode deteksi garis dilakukan dengan menggunakan dua macam metode,

yaitu dengan menggunakan sensor yang berupa *photodiode* dan metode deteksi garis berbasis visual. Sistem penjejakan garis berbasis visual memanfaatkan kamera untuk mendapatkan gambar garis. Gambar yang telah didapatkan kemudian diproses lebih lanjut untuk mendapatkan parameter – parameter garis yang berisi informasi mengenai posisi dan kemiringan garis. Pada aplikasi ini metode deteksi garis menggunakan sensor berupa *photodiode* tidak dapat digunakan karena metode ini mengharuskan jarak antara garis dan sensor untuk berdekatan.

Sistem *positioning* dalam ruangan telah menjadi tantangan yang sulit pada bidang navigasi[13]. Sistem navigasi dalam ruangan menggunakan GPS memeiliki performa yang buruk karena tingginya tingkat atenuasi dan *multipath interferences* pada penerimaan sinyal GPS. Hingga saat ini, banyak GPS *receiver* dapat memberikan estimasi *position, velocity* dan *timing* (PVT) yang baik pada aplikasi luar ruangan namun tidak dapat memberikan performa yang sama ketika digunakan di dalam ruangan. Oleh karena itu, sistem penjejakan garis perlu dilakukan karena metode navigasi konvensional menggunakan jaringan GPS tidak dapat dilakukan secara optimal di dalam ruangan.

1.2. Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah :

1. Desain sistem navigasi *drone* dalam ruangan, dimana sistem navigasi konvensional menggunakan GPS tidak dapat digunakan di dalam ruangan.
2. Bagaimana desain *indoor patrolling drone* yang mampu mengakomodasi sistem penjejakan garis berbasis visual?

1.3. Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut :

1. Memberikan rancangan sistem penjejakan garis berbasis visi komputer pada *indoor patrolling drone*.
2. Mendesain *indoor patrolling drone* yang mampu mengakomodasi sistem penjejakan garis berbasis visi komputer untuk melakukan patroli di dalam ruangan.

1.4. Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut :

1. Pada tugas akhir ini, penelitian difokuskan pada sistem

penjejakan garis dalam ruangan yang diimplementasikan pada *patrolling drone*, oleh karena itu proses *take-off dan landing* masih dilakukan secara manual oleh pengguna.

2. *Patrolling drone* yang dibuat diperuntukkan pada aplikasi dalam ruangan dikarenakan pencahayaan di luar ruangan tidak stabil sehingga mempengaruhi kinerja sistem deteksi garis.
3. Dikarenakan keterbatasan kapasitas dan berat drone, *patrolling drone* pada riset ini hanya memiliki waktu terbang selama kurang lebih 7 menit.

1.5. Metodologi Penelitian

Metodolgi dalam suatu penelitian dapat diartikan sebagaia langkah kerja atau strategi ilmiah yang memiliki tujuan untuk mendapatkan data yang akan digunakan untuk keperluan penelitian. Pada subbab ini akan dijelaskan langkah – langkah yang dilakukan untuk menyelesaikan tugas akhir ini.

1. Studi Literatur

   Tahap studi literatur berisi pengumpulan sumber informasi, dasar teori dan data – data yang dapat mendukung keabsahan tugas akhir ini. Studi literatur dilakukan dengan mengkaji sumber – sumber terpercaya seperti jurnal ilmiah, buku, makalah dan paper yang relevan. Adapun studi literatur tersebut mencakup dua topik utama yaitu mengenai *drone* dan sistem penjejakan garis.
2. Perakitan *drone quadcopter*

   Perakitan *drone quadcopter* dimulai dengan memilih komponen dengan spesifikasi yang sesuai dengan kebutuhan. Komponen yang digunakan merupakan komponen yang sudah populer digunakan. Setelah semua komponen didapatkan, maka dilanjutkan dengan merakit *drone quadcopter*. Proses perakitan *drone quadcopter* kemudian dilanjutkan dengan pengaturan *firmware* pada *flight controller* agar dapat terintegrasi dengan komponen - komponen yang lain.
3. Mengintegrasikan *hardware* raspberry pi dan kamera pada *drone quadcopter*

   Dalam tugas akhir ini raspberry pi merupakan komponen utama yang melakukan proses pengolahan input untuk menghasilkan sinyal output. Input sistem berupa gambar yang diperoleh dari kamera yang kemudian dilakukan proses pengolahan data pada raspberry pi untuk menghasilkan output berupa nilai *pitch, roll* dan *yaw* yang dikirimkan

pada *flight controller.*

Flight controller bertugas untuk mengirimkan sinyal kontrol pada keempat motor pada *drone*. Sinyal kontrol yang dihasilkan oleh *flight controller* menentukan sudut *pitch, roll* dan *yaw* pada *drone*.

4. Pengujian dan pengambilan data.

   Pengujian dilakukan untuk mengetahui kinerja sistem. Pengujian dilakukan dengan menerbangkan *drone* pada posisi tertentu kemudian memindahkan control pada mode otomatis. Dengan adanya pengujian tersebut bisa didapatkan beberapa parameter penting diantaranya untuk mendapatkan karakteristik kontroler yang digunakan, pengujian sensor, evaluasi pada proses interpretasi garis, dan lain lain.

5. Penulisan laporan

   Penulisan laporan dilakukan beriringan dengan tahap – tahap pengerjaan tugas akhir lainnya. Penulisan laporan bertujuan untuk melaporkan hasil tugas akhir dalam format yang sudah ditentukan. Laporan tugas akhir meliputi diantaranya pendahuluan, pengujian, analisa dan penutup berupa kesimpulan yang didapatkan.

1.6. Sistematika Penulisan

Untuk mempermudah dalam menyelesaikan penulisan buku ini, penulis membagi pembahasan menjadi lima bagian. Kelima bagian tersebut ditulis dengan menggunakan sistematika penulisan sebagai berikut :

- **BAB I : Pendahuluan**
  Pada bagian yang pertama ini akan dimuat mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, sistematika penulisan, dan relevansi.
- **BAB II : Teori Penunjang dan Tinjauan Pustaka**
  Pada bagian yang kedua, akan dibahas mengenai teori – teori pendukung penelitian ini. Teori tersebut harus berasal dari sumber – sumber terpercaya dan dapat dipertanggungjawabkan kebenarannya. Dalam tugas akhir ini penulis menggunakan teori pendukung yang berasal dari, riset paper, halaman web terpercaya, buku dan lain – lain.
- **BAB III : Perancangan Sistem**
  Pada bab ini akan dijelaskan mengenai rancangan dari sistem penjejakan garis pada *indoor patrolling drone.* Perancangan sistem penjejakan garis pada *indoor patrolling drone* meliputi hardware, *software*, sistem secara menyeluruh dan subsistem

pendukungnya.

- **BAB IV : Pengujian dan Analisis**
  Bagian keempat dalam penyusunan buku ini adalah pengujian dan analisis. Pada bagian ini akan dijabarkan data – data yang didapatkan dari pengujian yang telah dilakukan. Pada bagian ini juga akan dibahas mengenai metode – metode pengujian yang digunakan.

- **BAB V : Penutup**
  Bab lima merupakan bab terakhir pada penulisan buku ini. Pada bab yang terakhir ini penulis akan menyampaikan kesimpulan yang didapatkan setelah melalui proses perancangan dan pengujian pada sistem penjejakan garis *indoor patrolling drone* yang telah dibuat.

1.7. Relevansi

Sistem penjejakan garis pada indoor patrolling drone menggunakan penjejakan garis berbasis visual adalah metode baru yang dapat digunakan untuk mengawasi suatu area tertentu di dalam ruangan secara otomatis. Sistem penjejakan garis sudah banyak digunakan pada robot *line follower* dimana sebagian besar robot tersebut menggunakan sensor inframerah untuk mendeteksi garis. Dalam tugas akhir ini pengindraan visual komputer digunakan untuk menggantikan sensor inframerah tersebut.

Halaman ini sengaja dikosongkan

BAB 2
TEORI PENUNJANG DAN TINJAUAN PUSTAKA

2.1. Teori Penunjang

Dalam suatu penulisan ilmiah yang baik, tentunya diperlukan teori – teori pendukung. Teori pendukung tersebut akan menjadi dasar dari berbagai argumen yang terdapat didalamnya. Pada subbab ini akan dipaparkan mengenai landasan tori yang digunakan dalam merumuskan dan menyediakan solusi terhadap permasalahan yang telah disampaikan sebelumnya. Tinjauan pustaka akan memuat berbagai teori dan informasi mengenai komponen, teori dan sistem yang digunakan dalam penelitian ini.

2.1.1. Drone

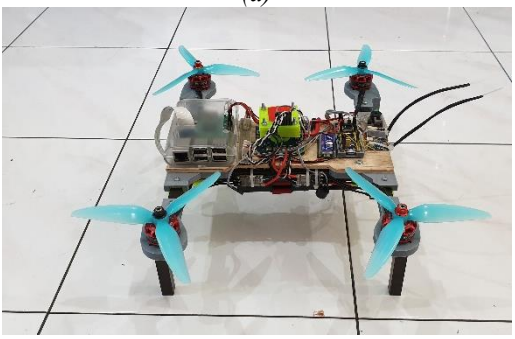
Drone atau *unmanned air vehicle* (UAV) adalah pesawat udara dimana tidak memiliki pilot yang berada di dalam pesawat tersebut. Peran pilot digantikan dengan suatu sistem komputer dan radio jarak jauh [14]. Pada sistem yang lebih kompleks drone juga dapat dioprasikan secara otomatis ataupun semi otomatis.

Secara garis besar terdapat dua kelompok UAV yaitu kelompok *fixed wing* dan *multirotor*. Ciri utama pada *multirotor* UAV adalah penggunaan lebih dari satu motor penggerak. Pada multirotor UAV, daya angkat didapatkan secara langsung dari motor – motor yang berada pada UAV tersebut. *Multirotor* UAV memiliki kelebihan yaitu dapat melakukan *take-off* dan *landing* secara vertikal, sehingga tidak diperlukan area yang terlalu besar. UAV berjenis *fixed wing* memiliki sayap untuk menghasilkan daya angkat. Berbeda dari *multirotor* UAV, pada *fixed wing* daya angkat tidak dihasilkan secara langsung oleh motor. Motor pada *fixed wing* UAV menghasilkan daya dorong untuk meningkatkan kecepatan terbang. Kecepatan terbang tersebut menghasilkan aliran udara pada sayap yang pada akhirnya memberikan daya angkat pada UAV. Pada gambar 2.1 berikut disajikan contoh UAV berjenis *fixed wing* (a) dan *multirotor* (b).

Seiring dengan perkembangan zaman, dalam beberapa tahun terakhir popularitas drone meningkat dengan pesat, baik pada kalangan masyarakat umum maupun dalam lingkup industri. Dalam kalangan masyarakat drone banyak digunakan pada *aerial photography*, akrobat dan *racing*. Dalam kalangan industri drone juga mulai banyak digunakan

diantaranya sebagai alat bantu survey kawasan[3], penebar pestisida[4], keperluan medis [5], monitoring lingkungan hidup[6], inspeksi infrastruktur jalur transmisi[2] dan lain – lain.

(a)

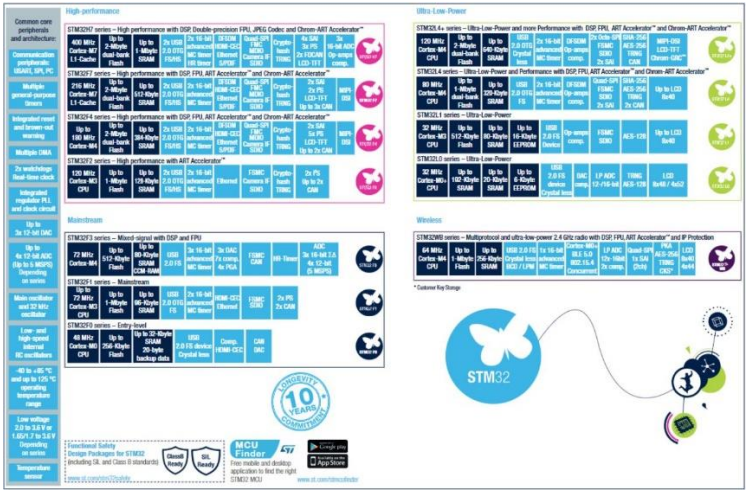


(b)

Gambar 2.1 Contoh *fixed wing* UAV*(a)* dan *multirotor* UAV*(b)*

2.1.2. STM32F103C8T6

STM32 adalah jajaran mikrokontroler yang diproduksi oleh ST Microelectronic. STM32 dapat dibagi menjadi Sembilan *sub family* yang dimana masing – masing *sub family* memiliki fitur yang berbeda - beda. Setiap mikrokontroler STM32 terdiri dari prosesor inti, RAM, memori *flash*, antarmuka *debugging* dan beberapa *peripheral* seperti timer, hardware USART, I2C, CAN bus, Watchdog timer dan lain – lain[15]. Beberapa fitur dan *peripheral* pada masing – masing *sub family* disajikan lebih ddetail pada gambar 2.2 berikut:



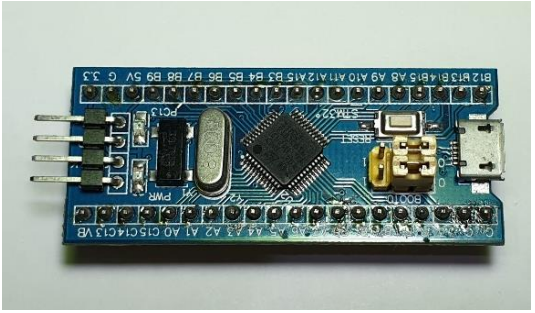
Gambar 2.2 Fitur dan *peripheral* STM32[16]

Diantara jajaran seri mikrokontroler lainnya, STM32 memiliki beberapa keunggulan sebagai berikut :

- **Mudah untuk dikembangkan**
  Jajaran mikrokontroler STM32 mudah untuk dilakukan pengembangan karena sudah banyak dijual baik dalam bentuk IC mikrokontroler maupun dalam bentuk *development board*. Beberapa *development board* yang banyak ditemui antara lain; blue pills, STM32 discovery board dan STM32 nucleo board.
- **Berbasis arsitektur ARM 32-bit**
  ARM 32-bit memiliki performa yang lebih baik jika dibandingkan dengan mikrokontroler lain yang berbasis 8/16-bit. Arsitektur ARM juga sudah menjadi standard industri dalam berbagai macam aplikasi, khususnya pada aplikasi *embedded system*, sehingga sudah banyak pengembangan dan *software* pendukung yang akan membantu pengembangan perangkat STM32.
- **Dapat menerima input 5v**
  Meskipun mikrokontroler STM32 beroprasi pada tegangan 3.3v,

sebagian besar pin GPIO dapat menerima input hingga 5v. Dengan demikian mikrokontroler STM32 memiliki kompatibilas dengan perangkat – perangkat lain yang bekerja pada level tegangan 5v.

- **Ketersediaan *peripheral***
  Salah satu alasan utama digunakannya mikrokontroler STM32 adalah ketersediaan *peripheral*. Meskipun STM32 dilengkapi dengan banyak *peripheral*, STM32 memiliki bentuk fisik yang relatif kecil.



Gambar 2.3 STM32F103 Blue Pill *development board*

STM32F103C8T6 adalah salah satu mikrokontroler STM32 yang termasuk dalam *sub family* STM32F1. STM32F103C8T6 memiliki inti ARM Cortex-M3 yang memiliki keepatan maksimum hingga 72MHz. STM32F103C8T6 juga dilengkapi dengan memori RAM sebesar 20kB dan memori flash sebesar 64kB[17].

2.1.3. Raspberry Pi

Raspberry pi merupakan *single board computer* yang dikembangkan oleh Raspberry Pi foundation. Tujuan utama dikembangkannya Raspberry Pi adalah sebagai media pengajaran mengenai sistem komputer, jaringan dan pemrograman. Raspberry Pi juga banyak digunakan dalam keperluan lain seperti *prototyping*, pengembangan robot dan lain – lain. Berbeda dari mikrokontroler, Raspberry Pi merupakan miniatur komputer yang dikemas dalam bentuk satu papan terintegrasi.

Gambar 2.4 Raspberry Pi versi 3B

Dalam sejarahnya Raspberry Pi memiliki beberapa versi. Setiap versi Raspberry Pi berbasis pada *system on chip* dari Broadcom yang terdiri dari RAM, CPU dan GPU. Pada beberapa versi Raspberry Pi juga dilengkapi dengan perangkat WiFi yang terintegrasi sehingga memudahkan pengguna untuk terhubung dengan internet.

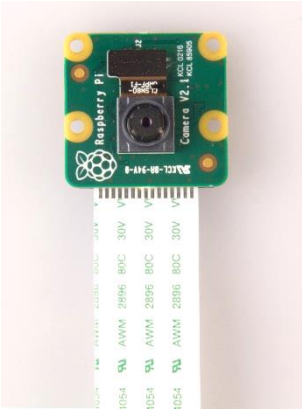
Raspberry Pi 3 model B adalah salah satu dari jajaran versi Raspberry Pi yang berbasis pada prosesor Broadcom BCM2837. Prosesor Broadcom BCM2837 dilengkapi dengan empat inti yang bekerja pada kecepatan clock sebesar 1.2GHz. Raspberry Pi 3 model B juga dilengkapi memori RAM dengan kapasitas sebesar 1GB[18]. Pada papan Rapberry Pi 3 model B terdapat CSI port yang digunakan untuk menghubungkan raspberry Pi camera.

2.1.4. Raspberry Pi Camera V2

Raspberry Pi camera V2 adalah kamera dengan konektor CSI yang kompatibel dengan konektor CSI pada Raspberry Pi. Raspberry Pi camera dirancang khusus untuk dapat diintegrasikan dengan mudah pada Raspberry Pi. Raspberry Pi Camera V2 dapat digunakan untuk merekam video dan untuk menggambil gambar.

Raspberry Pi camera V2 menggantikan versi sebelumnya Raspberry Pi camera V1. Raspberry Pi camera V2 menggunakan sensor Sony IMX219. Sony IMX219 adalah sensor kamera berbasis *back-illumintated* CMOS *image sensor* yang memiliki resolusi hingga 8.08M pixels[19]. Raspberry Pi camera V2 dapat mengambil gambar dengan resolusi 3280 x 2464 pixel dan video dengan resolusi 1080p pada 30fps, 720p pada 60fps dan 640x480p pada 90fps[20]. Dalam penggunaannya Raspberry Pi

camera V2 dapat diakses menggunakan API MMAL dan V4L, sehingga mudah untuk diintegrasikan dengan *library* dan bahasa pemrograman yang digunakan.



Gambar 2.5 Raspberry Pi camera V2

2.1.5. Flight Controller

Flight controller adalah perangkat yang berfungsi untuk mengendalikan drone/UAV. *Flight controller* mendapatkan input dari berbagai sensor yang terdapat pada drone diantaranya; gyroscope, accelerometer, magnetometer dan barometer. *Flight controller* akan melakukan penyesuaian pada masing – masing motor sehingga menghasikan daya dorong yang sesuai agar drone dapat mempertahankan penerbangan yang stabil. Pada beberapa sistem yang lebih kompleks, *flight controller* dapat beroprasi secara otomatis dan semi otomatis.

Terdapat berbagai macam dan versi *flight controller*, dari berbagai macam tersebut dapat digolongkan menjadi dua yaitu *opensource* dan *closed source*. Pada *open source flight controller*, pengguna dapat mekakukan modifikasi terhadap *software* maupun hardware pada *flight controller* tersebut. *Open source flight controller* pada umumnya memiliki pilihan perangkat pendukung yang lebih luas. Beberapa contoh *opensource flight controller* diantaranya CC3D, Sparky, MultiWii APM dan lain – lain. Pada *closed source flight controller* pengguna tidak dapat

melakukan modifikasi pada *software* maupun hardware *flight controller*. Beberapa closed source *flight controller* diantaranya Super-X, Mini-X, DJI NAZA dan lain – lain.

2.1.6. Omnibus F4 V6

Omnibus F4 V6 merupakan salah satu *open source flight controller* yang dikembangkan oleh Airbot. Omnibus F4 V6 memiliki prosesor utama berupa mikrokontroler STM32F405RGT6. Mikrokontroler STM32F4 mampu memberikan kemampuan perhitungan yang memadai, selain itu mikrokontroler terebut juga memiliki banyak *peripheral* yang dibutuhkan pada aplikasi tertentu. *Flight controller* ini juga dilengkapi dengan IMU berupa MPU6000/ICM-20608 yang dapat beroprasi dengan kecepatan sampling hingga 32kHz[21]. Omnibus F4 V6 dilengkapi dengan regulator tegangan STM L78 sehingga dapat secara langsung menerima input dari batrai.

Omnibus F4 V6 memiliki beberapa kelebihan jika dibandingkan dengan *flight controller* yang lain. Omnibus F4V6 memiliki bentuk fisik yang kecil sehingga dapat digunakan pada drone dengan dimensi frame yang kecil. Mikrokontroler yang digunakan pada omnibus F4 merupakan mikrokontroler dengan arsitektur 32-bit, hal tersebut membuat omnibus F4 lebih superior jika dibandingkan dengan *flight kontroler* lain yang berbasis mikrokontroler 8-bit. Salah satu kelebihan utama omnibus F4 adalah *open source*. Karena sifatnya yang *open source*, pengguna memiliki berbagai macam *firmware* yang dapat digunakan, diantaranya INAV, betaflight, cleanflight dan lain – lain.

2.1.7. Radio Controlled System

Radio controlled system (R/C) *system* sudah banyak digunakan dalam berbagai macam keperluan, diantaranya mengoprasikan wahana darat, air maupun udara dari jarak jauh. R/C *system* umumnya terdiri dari dua bagian utama yaitu Perangkat pengirim atau *transmitter* dan perangkat penerima atau *receiver. Transmitter* adalah perangkat yang menterjemahkan input dari pengguna menjadi sinyal radio dan selanjutnya akan diterima oleh *receiver*. *Receiver* adalah perangkat yang berfungsi untuk menterjemahkan sinyal radio menjadi sinyal yang dapat dimengerti oleh *flight controller*. Terdapat tiga jenis R/C *receiver* yaitu Parallel PWM, PPM dan serial receiver.

(a)

(b)

Gambar 2.6 *(a)* R/C *transmitter (b) receiver*

Pulse Width Modulation (PWM) *receiver* adalah *receiver* yang memberikan output berupa sinyal PWM. *Receiver* PWM memerlukan satu pin untuk setiap *channel* output. Pada *receiver* PWM, output akan dimodulasikan pada gelombang kotak dengan periode 20mS dan *on time* sebesar 1000uS hingga 2000uS. Waktu *on time* sebesar 2000uS menggambarkan nilai maksimum pada suatu *channel*, dan 1000uS menggambarkan nilai minimum.

Pulse Position Modulation (PPM) *receiver* adalah R/C *receiver* yang memberikan output berupa sinyal PPM. Pada *receiver* jenis ini, dalam satu pin output dapat mengirimkan informasi hingga 12 *channel*.

Serial receiver adalah R/C *receiver* yang menggunakan komunikasi UART untuk mengirimkan informasi pada setiap *channel*. Receiver jenis

ini dapat menerima hingga 16 *channel*. *Serial receiver* memiliki berbagai jenis protokol komunikasi, diantaranya; S.BUS, IBUS, XBUS, SUMH dan lain – lain[22]. Pada umumnya setiap produsen pembuat R/C *system* memiliki protokol komunikasi tersendiri. Berbeda dengan jenis *receiver* lain, *serial receiver* menggunakan sistem komunikasi digital.

Dalam melakukan komunikasi antara *transmitter* dan *receiver*, *radio controlled system* beroperasi pada spektrum gelombang radio[23]. Terdapat beberapa standar frekuensi gelombang *carrier* pada *Radio controlled system,* diantaranya yang paling banyak digunakan adalah 2.4GHz dan 900MHz. *Radio controlled system* yang beroprasi pada frekuensi 900MHz umumnya digunakan pada aplikasi jarak jauh dan 2.4GHz digunakan dalam aplikasi jarak dekat hingga jarak menengah.

2.1.8. Flysky FS-X6B

Flysky FS-X6B adalah R/C receiver yang diproduksi oleh salah satu produsen *radio controlled system* yang terkenal yaitu flysky. FS-X6B dapat memberikan output dalam bentuk sinyal PWM, PPW, S.BUS dan IBUS. Receiver ini didesain secara khusus untuk digunakan dalam *multirotor* UAV. Flysky FS-X6B memiliki dua buah antena *omnidirectional* yang digunakan untuk meningkatkan reliabilitas penerimaan sinyal dari *transmitter*[24]. Flysky FS-X6B dapat digunakn dengan *transmitter* yang menggunakan protokol AFHDS2A seperti FS-i10 FS-i6,FS-i6S, FS-i6X, FS-i4, FS-i4X dan lain – lain.

Receiver ini memiliki keunggulan dimensi yang kecil dan jarak penerimaan yang jauh. Flysky FS-X6B juga memiliki berat yang ringan yaitu 4.5gr. Oleh karena keunggulan – keunggulan tersebut, *receiver* ini cocok digunakan pada *multirotor* UAV.

2.1.9. Protokol IBUS

Protokol IBUS dalah protokol komunikasi yang digunakan pada *serial receiver* yang diproduksi oleh flysky. Protokol IBUS menggunakan komunikasi UART dengan *baudrate* sebesar 115200bd[25]. IBUS protocol mampu mengirimkan hingga 14 *channel* informasi dalam satu *packet data*. Dalam IBUS protocol *packet data* dikirimkan setiap 7ms. Maing - masing *packet data* terdiri dari 32byte data.

Setiap *packet data* diawali dengan dua byte *header*. *Header* berfungsi untuk menandakan awal dari setiap transmisi *packet data*. Dalam sistem R/C setiap *channel* memiliki range nilai 1000 hingga 2000. Nilai dalam setiap *channel* tersebut direpresentasikan dalam bilangan 16bit integer,

namun dalam komunikasi serial, pengiriman data dilakukan dalam format 8bit, maka masing – masing *channel* dikirimkan dalam bentuk dua byte data. *Packet data* dalam protokol IBUS diakhiri dengan dua byte *checksum*. Nilai *checksum* didapatkan dengan menjumlahkan setiap byte data dari indeks ke-0 hinggga ke-29. *Checksum* digunakan untuk mendeteksi kesalahan pada *packet data* yang diterima.

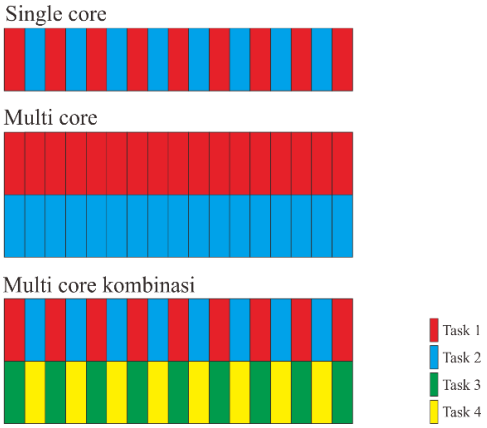
Tabel 2.1 Protokol IBUS

Indeks	Nilai	Keterangan
0	0x20	Header 1
1	0x40	Header 2
2	Channel1 | 0xFF	Channel 1 lower byte
3	Channel1 >> 8	Channel 1 upper byte
4	Channel2 | 0xFF	Channel 2 lower byte
5	Channel2 >> 8	Channel 2 upper byte
6	Channel3 | 0xFF	Channel 3 lower byte
7	Channel3 >> 8	Channel 3 upper byte
8	Channel4 | 0xFF	Channel 4 lower byte
9	Channel4 >> 8	Channel 4 upper byte
10	Channel5 | 0xFF	Channel 5 lower byte
11	Channel5 >> 8	Channel 5 upper byte
12	Channel6 | 0xFF	Channel 6 lower byte
13	Channel6 >> 8	Channel 6 upper byte
14	Channel7 | 0xFF	Channel 7 lower byte
15	Channel7 >> 8	Channel 7 upper byte
16	Channel8 | 0xFF	Channel 8 lower byte
17	Channel8 >> 8	Channel 8 upper byte
18	Channel9 | 0xFF	Channel 9 lower byte
19	Channel9 >> 8	Channel 9 upper byte
20	Channel10 | 0xFF	Channel 10 lower byte
21	Channel10 >> 8	Channel 10 upper byte
22	Channel11 | 0xFF	Channel 11 lower byte
23	Channel11 >> 8	Channel 11 upper byte
24	Channel12 | 0xFF	Channel 12 lower byte
25	Channel12 >> 8	Channel 12 upper byte
26	Channel13 | 0xFF	Channel 13 lower byte

27	Channel13 >> 8	Channel 13 upper byte
28	Channel14 | 0xFF	Channel 14 lower byte
29	Channel14 >> 8	Channel 14 upper byte
30	Checksum | 0xFF	Checksum lower byte
31	Checksum >> 8	Checksum upper byte

2.1.10. Multi*thread*ing

Multithreading adalah salah satu bentuk dari *multitasking*. *Multitasking* sendiri adalah suatu teknik dalam pemrograman komputer yang digunakan agar komputer dapat melakukan dua atau lebih *task*/aktifitas secara bersamaan[26]. Terdapat dua bentuk *multitasking* yaitu *multitasking* berbasis proses dan *multitasking* berbasis *thread*. *Multitasking* berbasis *thread* disebut juga dengan *multithreading*. *Multitasking* berbasis *thread* bertujuan untuk menjalankan dua aktifitas atau lebih pada satu program yang sama, sedangkan *multitasking* berbasis proses menjalankan dua aktifitas atau lebih pada waktu yang sama dalam dua atau lebih program yang berbeda.



Gambar 2.7 Eksekusi *multitasking* pada *singlecore* dan *multicore computer*

Dalam aplikasinya *Multitasking* dieksekusi dengan dua metode, bergantung pada arsitektur komputer. Seperti yang disakjikan pada gambar 2.7, *Multitasking* pada komputer dengan arsitektur *multicore* berbeda dengan komputer dengan arsitektur *single core*. Pada komputer

dengan arsitektur *multi core*, masing - masing *core* dapat menjalankan *task* yang berbeda. Pada komputer dengan arsitektur *single core*, semua task dijalankan pada satu *core* yang sama secara bergantian. Pada keadaan tertentu komputer juga dapat menjalankan kombinasi dari keduanya, diamana pada *multicore computer* masing - masing *core* dapat menjalankan lebih dari satu task.

2.1.11. Lucas-Kanade Optical Flow

Optical flow merupakan salah satu teknik dalam pengolahan citra untuk dapat mengetahui perpindahan titik – titik dalam suatu urutan gambar. *Optical flow* umumnya dilakukan dalam pengolahan citra yang berbentuk video, dimana dapat diasumsikan bahwa sebagian besar titik pada frame pertama terdapat pada frame selanjutnya. Pada aplikasinya, *optical flow* dapat digunakan untuk mengestimasi pergerakan dari suatu objek, *optical flow* juga dapat digunakan untuk mendeteksi pergerakan kamera.

Dalam keadaan ideal, algoritma *optical flow* dapat memberikan output berupa perkiraan kecepatan atau vektor perpindahan dari masing masing pixel yang menyusun suatu gambar. *Optical flow* yang diaplikasikan pada setiap pixel dalam suatu frame gambar disebut dengan *dense optical flow.* Sebaliknya, algoritma *optical flow* yang hanya diaplikasikan pada sebagian poin pada gambar disebut dengan *sparse optical flow.* Pada umumnya algoritma *sparse optical flow* lebih cepat dan handal karena hanya dibatasi pada titik – titik yang mudah untuk dilakukan *tracking*[27].

Lucas-Kanade *Optical-flow* pertama kali diusulkan pada 1981, sebagai salah satu metode *dense optical flow.* Namun karena algoritma Lukas-Kanade mudah untuk diaplikasikan pada sebagian titik saja, Lukas-kanade juga dapat digunakan pada *sparse optical flow*. Algoritma lucas-kanade bekerja berdasarkan tiga asumsi.

- *Brightness Constancy*
  Dalam algoritma Lucas-Kanade suatu titik diasumsikan tidak mengalami perubahan warna.
- *Temporal Persistence*
  Asumsi yang kedua yaitu titik – titik pada serangkaian gambar dianggap bergerak dengan pelan. Asumsi ini juga dapat diartikan bahwa selisih koordinat suatu titik antar frame gambar bernilai kecil.

- *Spatial Coherence*
  Titik – titik yang berdekatan dianggap berada pada permukaan yang sama, memiliki pergerakan yang sama dan diproyeksikan pada titik yang berdekatan pada bidang gambar.

2.1.12. Momen Pixel

Dalam ilmu fisika, momen adalah suatu persamaan yang melibatkan jarak dengan suatu besaran fisik lainnya. Momen pada umumnya memiliki satu titik referensi, dimana jarak adalah selisih koordinat antara titik referensi dan objek yang diamati. Dalam *image processing* momen pixel adalah hasil perkalian dari nilai pixel dengan jarak pixel tersebut dari titik referensi. Momen pixel dapat ditinjau dari dua sumbu, yaitu momen pada sumbu vertical (M_y) dan horizontal (M_x). Momen pixel pada sumbu vertikal dan horizontal dapat dijelaskan dengan persamaan berikut.

$$M_x = \sum_{i=1}^{w} \sum_{j=1}^{h} i \, . \, p_{(i,j)} \tag{2.1}$$

$$M_y = \sum_{j=1}^{h} \sum_{i=1}^{w} j \, . \, p_{(i,j)} \tag{2.2}$$

Perhitungan momen pixel dilakukan pada gambar binary dimana $p_{(i,j)}$ adalah nilai pixel pada koordinat (i, j). Nilai pixel dapat bernilai 1 atau 0. Penjumlahan momen pixel dilakukan pada seluruh frame gambar dengan lebar w dan tinggi h. Untuk mengetahui koordinat pusat persebaran pixel (x, y) dari suatu gambar binary, dapat digunaka persamaan berikut :

$$x = \frac{M_x}{\sum_{i=1}^{w} \sum_{j=1}^{h} p_{(i,j)}} \tag{2.3}$$

$$y = \frac{M_y}{\sum_{i=1}^{w} \sum_{j=1}^{h} p_{(i,j)}} \tag{2.4}$$

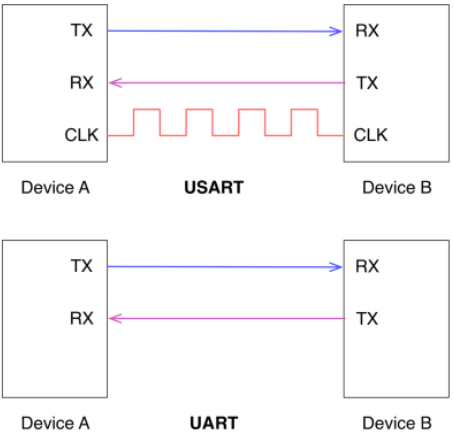
2.1.13. Komunikasi UART

Dalam melakukan komunikasi data antara dua atau lebih perangkat, dapat dilakukan dengan menggunakan komunikasi parallel ataupun serial[15]. Dalam komunikasi parallel komunikasi dilakukan dengan menggunakan sejumlah jalur komunikasi. Jumlah jalur komunikasi umumnya memiliki jumlah yang sama denga ukuran data yang dikirimkan. Jika data yang dikirimkan memiliki ukuran 8-bit maka jalur komunikasi yang digunakan sejumlah 8-bit. Kelebihan dari komunikasi data parallel adalah kecepatan pengiriman dan/atau penerimaan yang tinggi. Komunikasi parallel menjadi kurang efisien jika digunakan dalam komunikasi jarak jauh karena membutuhkan sambungan fisik yang cukup banyak.

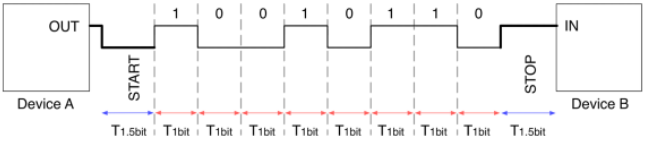
Komunikasi serial adalah suatu sistem komunikasi dimana pengiriman data dilakukan satu- persatu secara sekuensial. Komunikasi serial dilakukan melalui satu channel komunikasi. Jumlah jalur komunikasi pada komunikasi serial pada umumnya berjumlah dua untuk komunikasi *full duplex* dan satu pada komunikasi *half duplex*. Kelebihan utama dari komunikasi serial adalah hanya membutuhkan sedikit koneksi fisik. Dengan demikian komunikasi serial seringkali digunakan pada komunikasi jarak jauh.

Universal Synchronous and Asynchronous Receiver-Transmitter (USART) adalah salah satu komunikasi serial yang banyak digunakan. Pada umunya perangkat yang mendukung komunikasi USART dapat dioperasikan pada mode *Universal Asynchronous Receiver-Transmitter* (UART). Perbedaan utama dari USART dan UART adalah pada USART tersapat satu jalur *clock* yang digunakan untuk melakukan sinkronasi, sedangkan pada komunikasi UART tidak terdapat jalur *clock*. Sinkronasi pada USART memungkinkan mode komunikasi ini untuk memiliki kecepatan transaksi data yang lebih tinggi dari UART, namun komunikasi USART membutuhkan lebih banyak koneksi fisik.

Dikarenakan komunikasi UART tidak memiliki clock yang tersinkronasi, maka kedua perangkat yang sedang melakukan transaksi data harus menentukan kecepatan transfer data yang sama. Protokol komunikasi UART dapat memiliki delapan atau Sembilan bit dataframe. Ketika menggunakan sembilan bit *dataframe*, *dataframe* berisi delapan bit data dan satu bit *parity check*. *Parity check* digunakan untuk mendeteksi kesalahan pengiriman/penerimaan yang dapat diakibatkan oleh noise.



Gambar 2.8 Perbedaan USART dan UART



Gambar 2.9 Kommunikasi UART dengan 8bit *dataframe*

Komunikasi UART diawali dengan START bit. START bit diwakili dengan logika 0 selama 1.5 Tbit. Komunikasi UART kemudian dilanjutkan dengan delapan atau Sembilan bit *dataframe* dan diakhiri dengan STOP bit. STOP bit diwakili dengan logika 1 selama 1.5 Tbit. START dan STOP bit digunakan untuk melakukan sinkronasi *dataframe* antara perangkat pengirim dan penerima.

2.1.14. Regresi Linear

Regresi linear adalah suatu metode analisis yang digunakan untuk mengetahui hubungan linear antara dua variabel. Kedua variabel yang dimaksud adalah variabel dependent dan variabel independent. Dalam suatu percobaan, variable independent adalah variable yang dirubah secara terkontrol untuk mengetahui efeknya pada variabel dependen. Variable dependen adalah variable yang sedang diamati dalam suatu percobaan. Dengan metode regresi linear, kita dapat memperkirakan

hubungan linear antara variable dependent terhadap perubahan pada variable independent.

Regresi linear akan memperkirakan suatu fungsi linear yang mampu memetakan variabel dependen terhadap variable independen. Dengan asumsi fungsi linear yang dihasilkan adalah y = ax +b, maka nilai a dan b dipilih agar fungsi tersebut dapat memberikan estimasi yang optimal[28].

Gambar 2.10 (a) menunjukkan data point yang menyerupai suatu fungsi linear. Regresi linear dilakukan terhadap data point tersebut untuk menghasilkan estimasi fungsi linear yang dapat mendeskripsikan hubungan antara variabel dependen dan variable independen. Fungsi hasil estimasi regresi linear disajikan pada gambar 2.10 (b).

Untuk mengetahui seberapa baik fungsi linear yang dihasilkan dalam mengestimasi variabel dependen terhadap variabel independen, maka digunakan fungsi *sum of squared residual* (SSR). Dengan mengasusmsikan (x_i, y_i) adalah *data point* ke-i dan N adalah jumlah *data point*. Maka persamaan SSR dapat ditulis sebagai berikut.

$$SSR = \sum_{i}^{N} (y_i - a - bx_i)^2 \tag{2.5}$$

Fungsi linear dengan estimasi terbaik didapatkan dengan meminimalkan fungsi SSR tersebut. Fungsi SSR dapat diminimalkan dengan mendapatkan turunannya terhadap masing masing a dan b.

$$0 = \frac{\partial SSR}{\partial a} = -2 \sum_{i}^{N} y_i - a - bx_i \tag{2.6}$$

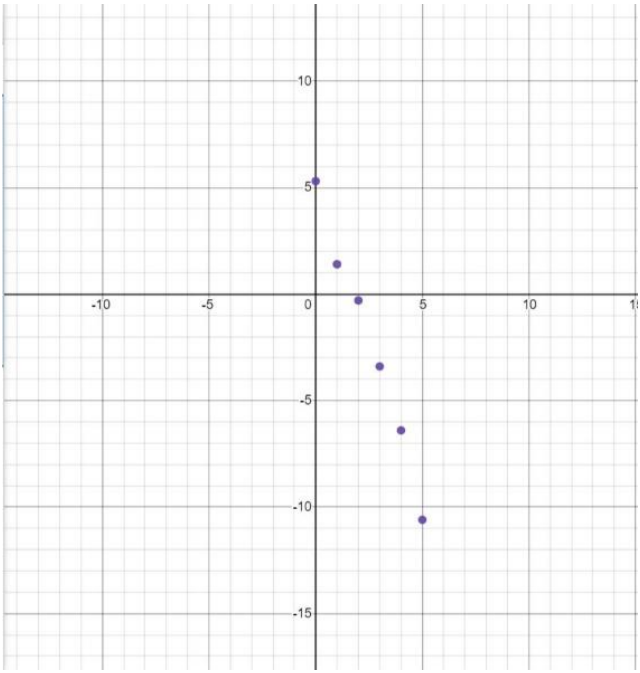
$$0 = \frac{\partial SSR}{\partial b} = -2 \sum_{i}^{N} x_i (y_i - a - bx_i) \tag{2.7}$$

Dengan melakukan eliminasi pada persamaan 2.6 dan 2.7, didapatkan :

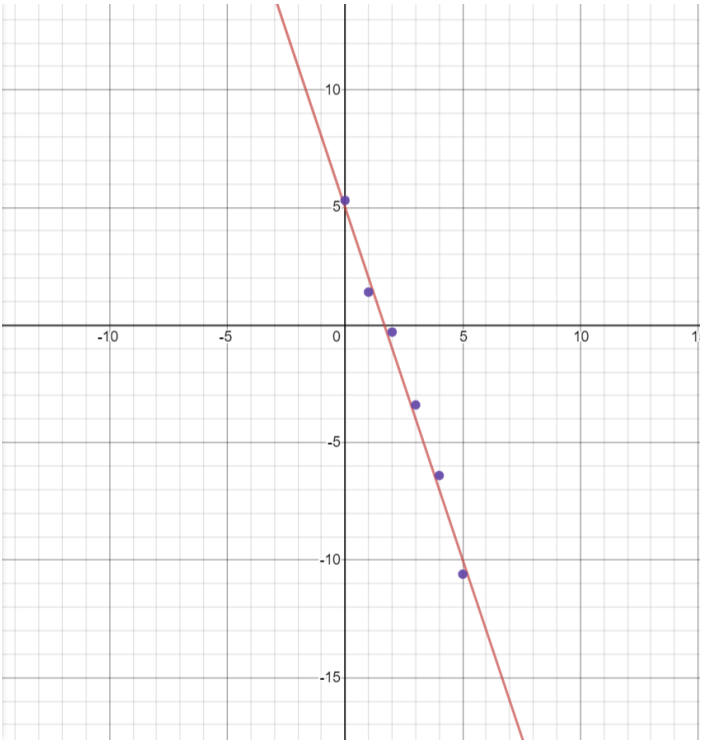
$$a = \sum_{i}^{N} \frac{N \sum_{i}^{N} x_i y_i - \sum_{i}^{N} x_i \sum_{i}^{N} y_i}{N \sum_{i}^{N} x_i^2 - (\sum_{i}^{N} x_i)^2} \quad (2.8)$$

$$b = \sum_{i}^{N} \frac{\sum_{i}^{N} y_i - a \sum_{i}^{N} x_i}{N} \quad (2.9)$$

Dengan menggunakan persamaan 2.8 dan 2.9 maka bisa didapatkan nilai a dan b agar y = ax+b optimal. Untuk melakukan evaluasi terhadap persamaan linear tersebut, dapat digunakan metode SSR seperti yang telah dijelaskan sebelumnya.



(a)

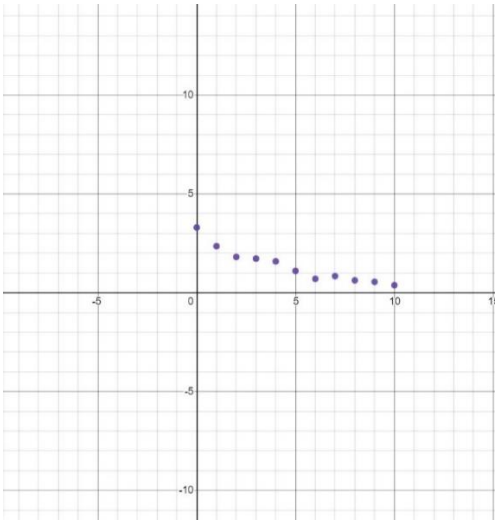


(b)

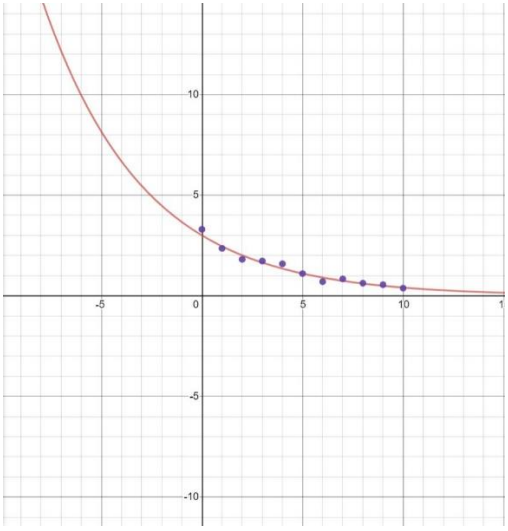
Gambar 2.10 Contoh regresi linear

2.1.15. Regresi Eksponensial

Sama seperti pada regresi linear yang telah dijelaskan sebelumnya, regresi eksponenisal adalah suatu metode analisis yang digunakan untuk mengetahui hubungan antara dua variable. Kedua variable yang dimaksud adalah variabel dependen dan variabel independen. Regresi eksponensial akan memperkirakan suatu fungsi eksponensial yang mampu memetakan variabel dependen terhadap variabel independen. Dengan asumsi fungsi eksponensial yang dihasilkan adalah $y = \beta e^{\alpha x}$, maka nilai α dan β dipilih agar fungsi tersebut dapat memberikan estimasi yang optimal.



(a)



(b)

Gambar 2.11 Contoh regresi eksponensial

Gambar 2.11 (a) menunjukkan *data point* yang menyerupai suatu fungsi eksponensial. Regresi eksponensial dilakukan terhadap data point tersebut untuk menghasilkan estimasi fungsi linear eksponensial. Fungsi hasil estimasi regresi eksponensial disajikan pada gambar 2.11 (b). Pada dasarnya regresi eksponensial akan menghasilkan persamaa dengan bentuk umum sebagai berikut:

$$y = \beta e^{\alpha x} \tag{2.10}$$

Dengan melakukan logaritma natural pada kedua sisi persamaan 2.10 maka didapatkan:

$$\ln y = \alpha x + \ln \beta \tag{2.11}$$

Dengan menggunakan asumsi Y = ln(y) dan B = ln(β) maka persamaan 2.11 dapat ditulis kembali sebagai berikut :

$$Y = \alpha x + B \tag{2.12}$$

Bentuk persamaan 2.12 menyerupai persamaan dasar regresi linear. Persamaan tersebut kemudian dapat dilakukan analisa regresi linear seperti yang telah dijelaskan sebelumnya. Pada langkah ini telah didapatkan konstanta α. Untuk mendapatkan nilai konstanta β dapat digunakan persamaan 2.14

$$\ln \beta = B \tag{2.13}$$

$$\beta = e^{B} \tag{2.14}$$

2.1.16. Kontrol PID

Kontroler PID adalah suatu kontroler yang terdiri dari tiga blok kontroler yaitu blok Proporsional, Integral dan Derivatif. Ketiga kontroler tersebut dapat digunakan secara bersamaan untuk membentuk sebuah kontroler PID ataupun sebagian untuk membentuk kontroler P, PI, PD atau kombinasi lainnya. Kontroler PID diawali degan perhitungan sinyal error. Sinyal error sendiri dapat didefinisikan sebagai selisih antara *setpoint* dan sinyal *feedback*, dimana *setpoint* adalah keadaan sistem yang diinginkan. Kontroler PID memiliki beberapa keuntungan, diantaranya stuktur sederhana, mudah diimplementasikan dan akurasi yang tinggi. Dengan mendefinisikan *u(t)* sebagai sinyal output maka kontroler PID

dapat ditulis dalam bentuk persamaan berikut :

$$u(t) = K_P e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \qquad (2.15)$$

Seperti yang telah dijelaskan sebelumnya, kontroler PID terditi dari tiga blok kontrol yaitu P, I dan D. Blok pertama dalam kontroler PID adalah blok P atau blok proporsional. Blok proporsonal memberikan respon yang proporsional terhadap sinyal error. Konstanta Kp atau konstanta proporsional adalah gain yang dimiliki oleh blok proporsional. Ketika melakukan tuning pada kontroler P maka pada dasarnya kita melakukan manipulasi pada konstanta Kp sehingga kontroler memberikan respon yang diiginkan. Secara praktis, ketika blok P memiliki gain yang terlalu besar maka sistem akan mengalami osilasi. Sebaliknya, ketika gain proporsional terlalu kecil, maka sistem menjadi kurang responsif.

Blok selanjutnya adalah blok I atau blok integral. Blok integral memberikan respon aditif atau integral dari sinyal error. Blok integral memiliki konstanta integral Ki yang berperan sebagai gain pada blok ini. Ketika kontroler tidak memiliki blok I maka kemungkinan besar kontroler tersebut memiliki *steady-state error*. Hal tersebut dikarenakan respon kontroler P akan semakin mengecil ketika error semakin mengecil pula. Ketika pada kasus seperti itu, kontroler I dapat berfungsi untuk meminimalisir atau bahkan menghilangkan *steady state error*. Kontroler I mampu mengurangi *steady state error* karena kontroler ini akan mengakumulasi nilai error, sehingga meskipun error bernilai kecil, seiring berjalannya waktu sinyal kontrol akan semakin membesar. Selain itu, blok I juga dapat mempercepat sistem untuk mencapai setpoint yang ditentukan. Dalam praktisnya, ketika blok I memiliki gain yang terlalu besar maka sistem akan mengalami overshoot, dan ketika gain terlalu kecil maka respon sistem terhadap gangguan akan lambat. Pengaturan gain yang baik pada blok I akan meberikan nilai oversoot yang kecil dan steady state error minimal.

Blok terakhir yang dimiliki kontroler PID adalah blok D atau derivative. Blok derivative memiliki gain yang disebut Kd atau konstanta derivative. Berbeda dengan blok – blok yang sebelumnya, blok D tidak merespon terhadap sinyal error secara langsung, melainkan merespon terhadap perubahan sinyal error. Blok D memberikan keluaran sinyal kontrol yang bergantung pada seberapa cepat perubahan sinyal error.

Keran blok ini hanya merespon terhadap perubahan sinyal error, blok D mampu mengurangi osilasi dan overshoot pada sistem. Dalam menetukan gain Kd, kita perlu berhati – hati karena blok D memiliki sifat yang ensitif terhadap noise. Ketika nilai Kd terlalu besar maka noise yang terdapat pada sinyal error juga akan dikuatkan.

Terdapat banyak metode yang dapat digunakan untuk melakukan tuning pada parameter PID. Dari bebrbagai metode tersebut, terdapat dua metode yang banyak digunakan. yaitu metode Ziegler-Nichols dan metode *trial and error.* Pada aplikasi PID yang berhubungan dengan UAV/drone, metode yang banyak digunakan adalah metode *trial and error*.

Metode *trial and error* diawali dengan mengatur Ki=0 dan Kd=0. Nilai Kp diatur pada nilai yang relatif kecil. Nilai Kp selanjutnya ditingkatkan hingga sistem mulai mengalami osilasi. Selajutnya tingkatkan nilai Kd hingga osilasi sistem menurun atau menghilang. Ketika melakukan tuning nilai Kd perlu juga diperhatikan respon sistem terhadap perubahan setpoint. Pada sistem dengan noise yang kecil maka nilai Kd dapat diatur lebih tinggi juka dibandingkan dengan sistem dengan noise yang lebih besar. Jika setelah pengaturan nilai Kd masih terdapat osilasi pada sistem, maka nilai Kp dapat dikurangi hingga osilasi menghilang. Nilai gain terakhir yang perlu diatur adalah Ki, pengaturan nilai Ki yang bagus akan menebabkan sistem lebih tahan terhadap gangguan. Pada aplikasi drone/UAV gangguan tersebut dapat berupa perubahan nilai throttle, angina dan lain – lain. Drone atau UAV dengan gain Ki yang baik dapat mempertahankan sudut pitch dan roll ketika nilai throttle berubah secara drastis.

Metode tuning parameter yang kedua adalah metode Ziegler-Nichols Metode Ziegler-Nichols pertama kali dikenalkan oleh John G. Ziegler dan Nathaniel B. Nichols pada tahung 1940. Seperti pada metode sebelumnya, metode Ziegler-Nichols diawali dengan mengatur Ki=0 dan Kd=0. Nilai Kp selanjutnya ditingkatkan hingga sistem mulai mengalami osilasi. Nilai Kp dimana sistem mulai mengalami osilasi disebut Ku. Setelah didapatkan nilai Ku tersebut, atur nilai Kp, Ki dan Kd sesuai dengan tabel berikut.

Tabel 2.2 Nilai gain Ziegler-Nichols

Jenis Kontroler	Gain	Gain	Gain
P	0.5	-	-
PI	0.45*Ku	Ku/1.2	-
PID	0.6*Ku	0.5*Ku	Ku/8

2.1.17. Threshold Warna

Threshold warna atau seleksi warna adalah suatu metode yang digunakan untuk melakukan ekstraksi gambar atau bagian dari gambar yang memiliki range warna tertentu[27]. Metode ini dapat digunakan untuk memisahkan informasi yang memiliki warna tertentu dari latarnya. Seleksi warna dapat dilakukan baik pada gambar dengan satu komponen warna ataupun tiga komponen warna. Dalam *image processing* intensitas warna pada masing – masing komponen diwakili oleh nilai 0-255.

(a) (b)

Gambar 2.12 Contoh *threshold* warna

Gambar 2.12 (a) menunjukkan metode seleksi warna yang menggunakan gambar denga tiga komponen warna. Metode seleksi warna akan menghasilkan gambar *binary*. Gambar *binary* adalah gambar yang hanya memiliki dua kemungkinan nilai intensitas yaitu 0 dan 1. Gambar 2.12(b) menunjukkan gambar *binary* yang didapatkan dari hasil seleksi warna. Metode seleksi warna pada gambar dengan tiga komponen warna dapat dijelaskan dengan persamaan sebagai berikut:

$$O_{(x,y)} = \begin{cases} 1 \; if \; l_r \leq I_{r(x,y)} \leq u_r \; \cap \; l_g \leq I_{g(x,y)} \leq u_g \; \cap \; l_b \leq I_{b(x,y)} \leq u_b \\ 0 \; otherwise \end{cases} \quad (2.16)$$

Persamaan 2.16 digunakan untuk melakuan seleksi warna pada gambar $I_{(x,y)}$. Gambar $I_{(x,y)}$ memiliki tiga komponen warna RGB yaitu $I_{r(x,y)}$, $I_{g(x,y)}$ dan $I_{b(x,y)}$. Seleksi warna akan menghasilkan gambar binary $O_{(x,y)}$. Nilai intensitas pada $O_{(x,y)}$ akan bernilai 1 jika masing – masing intensitas warna $I_{r(x,y)}$, $I_{g(x,y)}$ dan $I_{b(x,y)}$ berada diatara *threshold*(l_r, u_r), (l_g, u_g) dan (l_b, u_b).

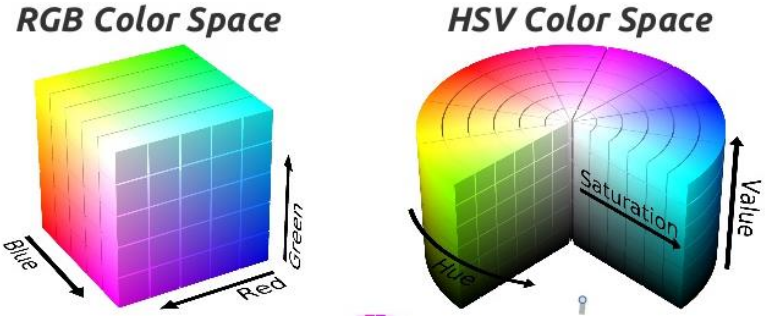
2.1.18. Color Space

Dalam melakukan identifikasi suatu objek pada gambar, warna dapat digunakan sebagai salah satu pengenal objek tersebut. Dalam ranah *image processing* terdapat beberapa *color space* yang banyak digunakan. Beberapa diantaranya adalah RGB, HSV, CMYK, CIELAB, YCrCb dan lain –lain. Dalam melakukan pemrosesan gambar secara digital, pemilihan *color space* sangatlah dibutuhkan untuk bisa mendapatkan hasil yang memuaskan. Dalam tugas akhir ini akan digunakan dua jenis *color space* yaitu RGB dan HSV.

RGB *color space* memiliki tiga elemen warna yaitu *red(R), green(G)* dan *blue(B).* Untuk keperluan pendeteksi objek seperti pendeteksi kulit manusia, RGB *color space* kurang dapat memberikan hasil yang baik karena adanya hubungan yang kuat antar elemen warna[30].

Sama seperti RGB *color space,* HSV *color space* memiliki tiga elemen warna yaitu *hue(H), saturation(S)* dan *value(V)*. Elemen *hue* mendeskripsikan warna pada suatu pixel. Nilai saturasi mendeskripsikan perubahan warna, dimana nilai saturasi maksimum adalah warna primer dan nilai saturasi minimum menghasilkan warna hitam-putih. *Value*

menggambarkan intensitas warna suatu pixel. Semakin tinggi nilai *value*, maka semakin terang pixel tersebut dan semakin kecil nilai *value* maka semakin gelap.



Gambar 2.13 RGB dan HSV *color space*[29]

Pada *image processing,* umumnya gambar yang ditangkap oleh kamera merupakan gambar RGB. Untuk melakukan pemrosesan gambar dalam HSV *color space* dapat dilakukan konversi RGB menjadi HSV. Untuk melakukan konversi tersebut, dapat digunakan persamaan sebagai berikut:

$$V \leftarrow \max(R, G, B) \tag{2.17}$$

$$S \leftarrow \begin{cases} \dfrac{V - \min(R, G, B)}{V} \; jika\, V \neq 0 \\ 0 \; lainnya \end{cases} \tag{2.18}$$

$$H \leftarrow \begin{cases} 60\dfrac{(G - B)}{V - \min(R, G, B)} \; jika\, V = R \\ 120 + 60\dfrac{(B - R)}{V - \min(R, G, B)} \; jika\, V = G \\ 240 + 60\dfrac{(R - G)}{V - \min(R, G, B)} \; jika\, V = B \end{cases} \tag{2.20}$$

2.1.19. BMP280

BMP280 adalah sensor tekanan brometrik digital. Sensor

BMP280 memiliki dimensi yang sangat kecil yaitu 2.0x2.5mm dan memiliki ketebalan kurang dari satu millimeter. Dimensinya yang kecil dan konsumsi daya yang kecil yaitu 2.7uA mendukung aplikasi sensor BMP280 pada perangkat – perangkat yang menggunakan sumber daya batrai seperti, telepon genggam, modul GPS, jam tangan dan lain - lain.

Gambar 2.14 Modul sensor tekanan barometrik BMP280

BMP280 merupakan penerus dari sensor barometer sebelumnya yairu BMP180. Dibandingkan dengan BMP180, BMP280 mampu memberikan performa yang lebih baik pada berbagai aplikasi yang membutuhkan pengukuran tekanan yang presisi. BMP280 juga mampu beroprasi pada tingkat noise yang lebih kecil dibandingkan pendahulunya, selain itu BMP280 memiliki dimensi 63% lebih kecil.

Range pengukuran pada BMP280 bernilai antara 300 hingga 1100 hPa. Akurasi pengukuran BMP280 dapat mencapai 0.12hPa atau setara dengan ketinggian 1 meter. BMP280 mampu bekerja pada suhu -40 hingga 85^0C. BMP280 umunya digunakan pada sistem seperti navigasi GPS, navigasi dalam ruangan, ramalan cuaca dan pendeteksi kecepatan vertikal[31].

2.2. Tinjauan Pustaka

Tinjauan pustaka dilakukan untuk melakukan perbandingan antar penelitian yang sedang dierjakan dengan teknologi tau penelitian yang sudah dilakukan sebelumnya. Dalam subbab ini akan dilakukan peninjauan terhadap beberapa hasil penelitian dengan topik yang tidak berbeda jauh dengan topik penelitian tugas akhir ini.

2.2.1. Medical Drones System for Amusement Parks[5]

Dengan semakin pesatnya perkembangan teknologi pada saat ini, banyak perusahaan yang berkerak dalam bidang tekonologi dan industry

yang bekerja keras untuk mengimbangi perkembangan teknologi tersebut. Drone adalah salah satu teknologi yang sedang berkembang pesat.

Drone dapat dikembangkan untuk berbagai kebutuhan dan dapat memberikan banyak keuntungan, salah satunya adalah *medical drone* yang dapat menyelamatkan nyawa manusia dengan memberikan bantuan medis. *Medical drone* dapat memberikan pertolongan tambahan pada tim medis dengan menyediakan bantuan berupa memberikan suplai medis pada daerah yang jauh dan sulit untuk dijangkau. Drone juga dapat digunakan untuk mempercepat pencarian korban dan memberikan pertolongan pertama hingga tim medis dapat menjangkau tempat korban.

Dalam paper ini penulis akan mengimplementasikan *Medical Drones System* (MDS). MDS adalah sistem yang dapat diimplementasikan pada tempat – tempat seperti *ski resort* dan taman hiburan. Tempat – tempat tersebut seringkali dipenuhi oleh turis yang tersebar pada daerah yang luas. Hal tersebut akan mempersulit tim medis untuk mencapai korban ketika terjadi keadaan darurat. Oleh karena itu MDS dirancang untuk dapat mencapi korban dengan cepat.

Dalam penelitian ini telah dibuat sistem yang terdiri dari beberapa drone. Kumpulan drone tersebut dapat dengan mudah menemukan korban dengan bantuan navigasi GPS. Drone kemudian dapat mengirimkan bantuan suplai medis pada lokasi terjadinya keadaan darurat. Sistem MDS juga memiliki aplikasi telepon genggam yang digunakan untuk melakukan komunikasi dengan stasiun pusat.

2.2.2. Adaptive Filter Design for UAV Navigation with GPS/INS/Optic Flow [32]

Sistem GPS dan *Inertial Navigation System* (INS) yang terintegrasi seringkali digunakan untuk menentukan informasi *Position, Velocity* dan *Altitude* (PVA) pada UAV. Kekurangan dari sistem tersebut adalah ketidakmampuannya dalam menentukan ketinggian permukaan tanah keti melakukan proses *landing.* Metode *optical flow* dapat digunakan untuk mendukun sistem INS. Namun pada paper ini ditemukan bahwa ketidakpastian pada pemodelan stokastik dapat mengurangi performa sistem secara signifikan. Terebih lagi pemodelan stokastik dari metode *optical flow* sulit untuk ditentukan secara pasti.

Dalam paper ini penulis berusaha meningkatkan performa filter pada pengintegrasian optical flow dengan sistem INS. Filter yang dibuat bertujuan untuk mengatasi ketidakpastian pada model stokastik *optical flow.* Dalam paper ini algoritma adaptasi berbasis pencocokan kovarians

telah diimplementasikan dengan extended kalman filter. Algoritma baru ini menunjukkan peningkatan performa dalam integrasi optical flow dengan INS.

2.2.3. Implementation of autonomous line follower robot[33]

Line follower adalah robot otonom yang mendeteksi dan bergerak mengikuti garis. Jalur yang digunakan dapat berupa garis yang dapat dilihat maupun yang tidak dapat dilihat sperti garis medan magnet. Robot *line follower* menggunakan *closed loop system.* Robot line follower harus dapat mendetksi garis dan bermanuver sedemikian sehingga dapat tetap berada pada jalur yang ditentukan. Robot line follower memiliki desain yang simple namun efektif dalam melakukan tugasnya untuk mengikuti garis.

Robot yang dibuat pada paper ini menggunakan rangkaian logika dan input sensor untuk mendapatkan karakteristik pergerakannya. Salah satu atribut yang signifikan adalah akurasi yang tinggi dalam mengikuti garis. Pada *line follower* ini kontrol tidak dilakukan oleh ikrokontroler, melainkan dengan menggunakan rangkaian elektronik sederhana. Pengembangan lebih lanjut pada robot ini dapat berupa penambahan sonar dan sensor inframerah, sehingga robot dapat menghindari halangan secara otomatis.

2.2.4. Line follower robot: Fabrication and accuracy measurement by data acquisition[34]

Robot *line follower* memiliki peranan yang sangat penting dalam industri manufaktur. Dala paper ini akan dibahas mengenai efisiensi robot, respon sensor, mendapatkan data sensor, koreksi error. Robot pada paper ini merupakan bentuk sederhana dari *line follower robot*. Bentuk yang lebih kompleks dapat diproduksi berdasarkan pada bentuk yang lebih sederhana. Paper ini lebih terfokus pada pengambilan data *line follower robot* selama pengujian sehingga robot dapat diproduksi secara massal dengan kebutuhan yang spesifik.

Salah satu aplikasi dari robot *line follower* adaah sebagai penuntun jalan bagi tunanetra. Robot line follower dapat mengintegrasikan buzzer untuk memperingatka pengguna. *Line follower robot* jug dapat digunakan untuk menggantikan manusia dalam menangani material berbahaya selama proses industri bahan kimia.

2.2.5. *Design of all color line follower sensor with auto calibration ability*[35]

Sistem yang dibuat pada paper ini terfokus untuk mendesain sebuah *line follower sensor* untuk berbagai aplikasi robotika. Tidak seperti sensor konvensional yang menggunakan pasangan IR dan fotodioda sebagai *transmitter* dan *receiver.* Sensor konvensional hanya bekerja dengan baik untuk mendeteksi garis berwarna putih pada latar hitam atau sebaliknya. Paper ini akan membahas desain dari *sensor array* menggunakan RGB LED sebagai transmitter dan fototransistor sebagai *receiver.* Sensor ini dapat digunakan pada permukaaan dengan warna apapun. Sensor ini dapat secara otomatis mengganti – ganti warna LED antara merah, hijau dan biru, bergantung pada warna permukaan dan garis yang dideteksi. Sensor ini dapat digunakan pada aplikasi dimana garis *line follower* digambar pada permukaan yang berbeda warna. Sensor dan kontroler pada sistem ini memiliki algoritma yang digunakan untuk mengganti warna LED bergantung pada warna latar dan garis yang sedang digunakan.

Pada paper ini telah dibuat suatu sistem yang dapat mengikuti garis apapun tanpa dipengaruhi oleh warna garis dan latar. LED dapat secara otomatis berganti warna bergantung pada warna garis dan latar. Sistem yang dibuat juga memiliki kemampuan *threshold* otomatis.

2.2.6. *Indoor Surveillance Security Robot with a Self-Propelled Patrolling Vehicle*[1]

Self-propelled patrolling vehicles dapat melakukan patrol secara periodic pada area yan ditentukan untuk memastikan keamanan layaknya manusia. Alat yang dibuat tidak hanya akan mengurangi tenaga manusia, juga memastikan tidak adanya kesalahan yang dapat disebabkan oleh *human error*. Alat ini berbeda dengan sistem patroli konvensional yang terbatas pada tenaga manusia dan kamera dengan posisi yang tetap. Untuk dapat memperbaiki situasi tersebut, paper ini akan membahas mengenai *Self-propelled patrolling vehicles* yang dapat bergerak secara otomatis dan merekan gambar denganmenggunakan IPCAM pada area patrol yang telah ditentukan. Pengguna dapat menggunakan telepon genggam untuk terkoneksi dengan alat ini dan mengontrol posisi dalam ruangan.

Sistem yang dibahas pada paper ini diimplementasikan pada computer server, *Self-propelled patrolling vehicles* dan jaringan untuk memberikan fitur pengawasan dan pengendalian jarak jauh. Alat yang dibuat menggunakan teknologi RFID untuk dapat bernavigasi pada daerah yang telah ditentukan. Pesan peringatan dapat dikirimkan pada

pengguna oleh server melalui *smart-phone* maupun MSN. Algoritma deteksi wajah digunakan pada sistem ini untuk mengenali penyusup. Alat ini juga memiliki kemampuan untuk melakukan proses *charging* secara otomatis jika batrai lemah terdeteksi.

2.2.7. Autonomous Flight Drone for Infrastructure (Transmission line) Inspection (3)[2]

Pada paper ini akan dikembangkan drone yang digunakan untuk melakukan inspeksi secara otomatis pada jalur transmisi daya tegangan tinggi. Pada umumnya drone melakukan navigasi dengan menggunakan GPS, namun GPS seringkali error pada daerah sekitar jalur transmisi tegangan tinggi. Pada drone ini digunakan beberapa *feature points* pada gambar untuk melakukan lokalisasi.

Paper ini membahas mengenai metode yang digunakan untuk mengstimasi posisi dan ketinggian drone berdasarkan *feature points* sebagai referensi. Paper ini bertujuan untuk mengembangkan sistem drone untuk keperluan inspeksi pada jalur transmisi.

2.2.8. Development and evaluation of drone mounted sprayer for pesticide applications to crops[4]

Penggunaan perstisida dalam agrikultur adalah salah satu proses yang sangat penting. *Drone mounted sprayer* terdiri dari motor BLDC, LiPo, tangki pestisida, pompa, dan rangka pendukung. Enam buah motor BLDC dipasangkan pada rangka *hexacopter* untuk memberikan kemampuan mengangkat beban hingga 5kg. Dua buah batrai LiPo 6 sel digunakan sebagai suplai arus pada sistem pendorong drone. Drone dirancang untuk dapat membawa 5 liter cairan pestisida. DC motor dan pompa digunakan untuk memberikan tekanan pada cairan perstisida. Cairan pestisida kemudian disemprotkan melalui empat buah *nozzle* pada drone. Pada drone ini juga digunakan FPV kamera untuk keperluan monitoring.

Teknologi ini sangat berguna untuk menyemprotkan pestisida pada tempat yang tidak dapat dijangkau oleh manusia. Teknologi ini juga dapat meningkatkan cakupan dan efektifitas penyebaran pestisida. Drone yang telah dibuat memiliki kemampuan untuk menampung pestisida hingga 5.5 liter dan memiliki waktu terbang hingga 16 menit.

2.2.9. Autonomous Drones for Disasters Management: Safety and Security Verifications[8]

Ketika dalam keadaan darurat seperti adanya bencana alam atau krisis, informasi menjadi peranan penting dalam melakukan menejemen

dalam penanganan situasi tersebut. UAV dapat memberikan dukungan pada pihak yang bertugas untuk mendapatkan informasi dan memberikan penilaian pada situasi yang sedang berlangsung. UAV atau *drone* dapat diimplementasikan untuk memberikan tiga macam dukungan kemanusiaan yaitu (1) sistem komunikasi dan koordinasi, (2) operasi pencarian dan (3) penjangkauan medan. Dalam melaukan penjangkauan medan yang sulit UAV dapat dioperasikan dengan cepat untuk melakukan identifikasi pada korban yang letaknya tersebar. Sedangkan pada operasi pencarian korban, UAV dapat mengenali gelombang elektromagnetik yang dipancarkan oleh perangkat elektronik yang dibawa oleh korban yang sedang terkubur dalam bangunan runtuh atau tersembunyi dalam hutan yang lebat.

Kemampuan UAV untuk beroprasi secara mandiri merupakan salah satu alasan utama penggunaan teknologi ini oleh tim penanganan bencana. Untuk dapat beroprasi secara mandiri, drone menggunakan sistem navigasi otomatis yang mampu mengenali medan di sekitar drone dalam bentuk rekonstruksi tiga dimensi. Sistem navigasi tersebut kemudian digunakan oleh UAV untuk menedeteksi dan menghindari halangan. Pada implementasi ini, UAV juga diharapkan dapat melakukan deteksi dan *tracking* pada korban.

Kemampuan UAV untuk beroprasi secara mandiri juga menimbulkan suatu permasalahan baru. Pengoprasian UAV secara mandiri harus memenuhi persyaratan keamanan dan keselamatan yang tinggi. Standar keamanan dan keselamatan yang tinggi berfungsi untuk mencegah adanya korban tambahan yang disebabkan oleh kegagalan sistem UAV. Teknologi UAV yang didukung dengan sistem keamanan dan keselamatan yang baik diharapkan dapat menjadi teknologi pendukung pihak – pihak yang bertugas dalam penanganan keadaan darurat.

2.2.10. Innovative Drone Selfie System and Implementation [9]

Dalam beberapa tahun terakhir, UAV mendapatkan banyak perhatian dan menjadi salah satu teknologi yang populer. Terdapat banyak jenis UAV yang dapat ditemui di pasaran dan sebagian besar dari drone tersebut dilengkapi dengan kamera. UAV dapat dioperasikan dengan menggunakan kontroler khusus, pada beberapa UAV juga dapat dioperasikan dengan menggunakan aplikasi antarmuka yang dijalankan pada komputer atau telepon pintar pengguna.

UAV dapat digunakan untuk mengambil gambar atau video. Video dan gambar yang diambil dengan menggunakan UAV dapat

memiliki sudut jangkauan yang lebih besar jika dibandingkan dengan gambar yang diambil oleh gambar yang diambil dengan menggunakan kamera genggam. Pengoprasian UAV untuk melakukan pengambilan gambar dan video mengharuskan pengguna untuk mengoprasikan UAV secara manual untuk mendapatkan jarak dan sudut pengambilan yang baik. Sistem pengoprasian tersebut dianggap tidak praktis karena pengguna harus melihat monitor gambar secara terus menerus.

Pada riset ini akan dikembangkan UAV yang digunakan untuk mengambil gambar selfie secara otonom. Pada UAV ini pengguna dapat memilih *template* foto pada aplikasi penduung, *drone* kemudian terbang pada posisi pengambilan gambar secara otomatis. UAV menggunakan *image-based visual servoing* untuk menentukan posisi pengambilan gambar. *Drone* inovatif ini terbukti dapat mengambil gambar secara efektif pada berbagai pengujian yang dilakukan.

2.2.11. The Utilization of Drones for Smart Governance Implementation: Tax Extensification Strategy for Transfer of Ownership in Real Estate Sales at Cibinong Primary Tax Office [10]

Pada tahun 2017, sektor perumahan dianggap menjadi salah satu sektor yang sedang berkembang pesat. Sektor perumahan di Indonesia menjadi sektor yang menarik, sektor ini terus mengalami peningkatan pasar yang besar. Industri perumahan juga didukung oleh banyaknya proyek pemerintah yang memungkinkan lokasi – lokasi baru untuk menjadi lahan perumahan.

Salah satu kantor pajak di Indonesia, cibinong menjadikan sektor perumahan menjadi salah satu sumber utama pendapatan pajak pada daerah tersebut. Namun pada kenyataannya, pertumbuhan pajak sektor perumahan pada tahun 2017 tidak mengalami pertumbuhan seperti yang diperkirakan. Berdasarkan data kantor pajak cibinong, pajak yang diperoleh dari sektor perumahan menurun hingga 19.57%. Penurunan pendapatan pajak pada sektor ini dicurigai disebabkan oleh kecurangan – kecurangan yang dilakukan baik oleh individu maupun korporat.

Pada studi ini, UAV diimplementasikan untuk pengambilan data area property. Data yang telah didapatkan kemudian digunakan untuk melakukan evaluasi peluang dan kemungkinan adanya kecurangan seperti yang telah dibahas sebelumnya. UAV digunakan untuk melakukan survey area pembangunan pada area kerja kantor pajak cibinong. Implementasi *drone* pada bidang ini juga diharapan dapat membantu pemerintah dalam

menerapkan program *smart governance* yang diusung oleh pemerintah pusat.

2.2.12. Fusion of Drone and Satellite Data for Precision Agriculture Monitoring [11]

Sistem pertanian presisi sudah menjadi objek penelitian selama setidaknya duapuluh tahun terakhir, namun bidang ini masih terus menghasilkan penemuan dan inovasi baru. Pada beberapa tahun terakhir, *drone* atau UAV berukuran kecil telah menjadi salah satu teknologi pendukung pada sistem pertanian presisi. *Drone* atau UAV bisa didapatkan dengan harga yang terjangkau dan memiliki kemampuan untuk mengambil gambar lokasi secara geografis. Kemampuan tersebut membantu pengguna untuk mendapatkan gambar data permukaan tanah yang menyeluruh dan lebih jelas. *Drone* dapat dilengkapi dengan kamera yang mampu menangkap cahaya pada spectrum mendekati infra merah, gambar yang dihasilkan kemudian dapat digunakan untuk mengetahui kesehatan tanaman agrikultur.

Pengambilan gambar dengan skala besar menggunakan satelit telah banyak dilakukan. Gambar yang dihasilkan kemudian dilakukan proses klasifikasi menggunakan berbagai metode, diantaranya *decision tree, thresholding,* SVM dan *neural network.* Klasifikasi tersebut digunakan untuk mengetahi berbagai macam permukaan tanah, seperti tumbuhan, permukaan air, perumahan dan lain- lain. Sistem klasifikasi tersebut kurang dapat digunakan pada aplikasi agrikultur karena gambar yang dihasilkan memiliki resolusi yang kurang memadai.

Pada penelitian ini akan ditawarkan suatu metode untuk dapat melakukan klasifikasi mengenai keadaan tumbuhan pertanian dengan menggunakan data yang diperoleh melalui satelit dan *drone.* Dengan melakukan penggabungan data dari satelit dan *drone*, penelitian ini berhasil melakukan klasifikasi hasil pertanian menjadi dua kelas yang berbeda.

2.2.13. Feasability Study of RFID-Mounted Drone Application in Management of Oyster Farms [7]

Peternakan tiram merupakan salah satu jenis peternakan biota laut yang populer pada pantai barat Taiwan. Peternakan tiram di kota tainan menggunakan sistem rakit dan *polystyrene foam* (PSF) untuk memelihara tiram. Setelah tiram dipanen, sering kali material PSF dibuang secara langsung di laut. Hal tersebut berkontribusi pada penumpukan limbah

sampah di laut, yang tidak hanya mencemari daerah di sekitar peternakan tiram. Oleh kareana itu, pemerintah Taiwan mulai memberlakukan regulasi yang mengharuskan peralatan yang digunakan pada sistem pertanian tiram untuk memiliki sistem pengenalan. Material PSF juga diharuskan untuk dikumpulkan pada tempat yang telah disediakan. Meskipun regulasi sudah diberlakukan, masih banyak ditemui pelanggaran – pelanggaran yang dilakukan oleh peternak tiram.

Pada penelitian ini akan dilakukan studi efektifitas aplikasi teknologi UAV untuk melakukan pengenalan tanda pengenal RFID pada PSF. Penelitian ini diharapkan dapat menghasilkan suatu metode untuk melakukan lokalisasi penanda RFID pada PSF. Dengan danya metode tersebut, pemerintah Taiwan dapat dengan mudah mengetahui sumber pencemaran limbah laut berupa PSF. Pemerintah kemudian dapat memberikan penindakan yang sesuai.

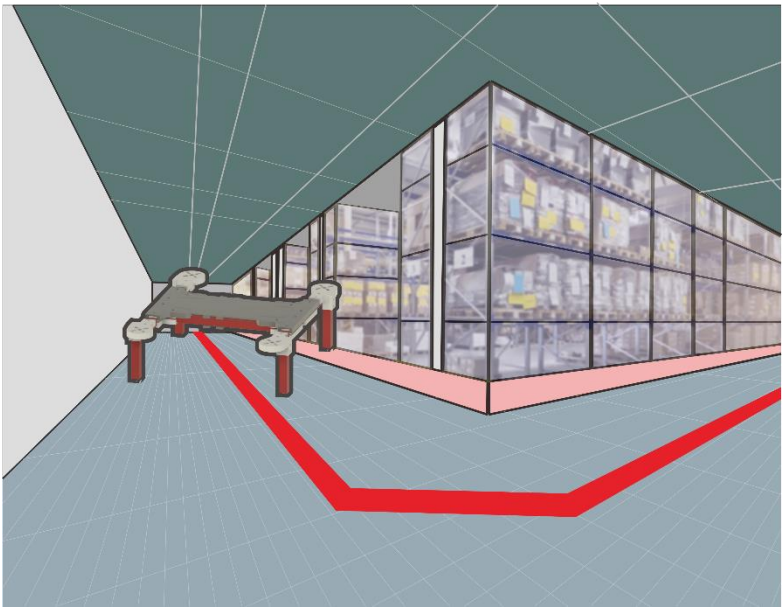
2.2.14. Development of a low cost and light weight UAV for photogrammetry and precision land mapping using aerial imagery [36]

Survey dan pemetaan permukaan tanah merupakan salah satu bagian penting dalam bidang pembangunan. Survey dan pemetaan permukaan tanah dapat menghasilkan model digital yang mendeskripsikan ketinggian permuakaan tanah dalam betuik tiga dimensi. Dengan menggunakan model tiga dimensi tersebut dapat dilakukan analisa lebih lanjut unutk melakukan pemetaan resiko banjir, pemetaan area rawan longsor dan banyak lainnya.

Hingga saat ini, kebanyakan survey yang dilakukan untuk kebutuhan pembangunan masih dilakukan dengan metode terrestrial. Pada penelitian ini akan dikembangkan metode untuk mendapatkan hasil pemetaan permukaan tanah dengan cepat, simpel dan murah. Metode ini memanfaatkan teknologi UAV yang dibuat secara khusus untuk melakukan *aerial imagery.*

Penelitian ini telah berhasil mengembangkan UAV yang mampu melakukan pemetaan permukaan tanah dengan kemampuan pemetaan area sebesar 25km^2 dalam waktu 25menit. Hasil pemetaan yang dihasilkan memiliki resolusi hingga 6cm/pixel.

BAB 3
PERANCANGAN SISTEM



Gambar 3.1 Skema dan ilustrasi *indoor parolling drone*

Bab tiga ini akan menjelaskan mengenai desain dan perancangan sistem navigasi penjejakan garis pada *indoor patrolling drone*. *Indoor patrolling drone* sendiri terdiri dari dua bagian utama, yaitu drone dan sistem navigasi penjejakan garis. Dalam perancangannya drone yang digunakan harus mampu mengakomodasi sistem navigasi penjejakan garis. Drone yang dibuat berjenis *multirotor quadcopter*.

Bagian kedua dari *indoor patrolling drone* adalah sistem navigasi penjejakan garis. Sistem navigasi penjejakan garis bertujuan untuk memberikan drone kemampuan untuk bernavigasi di dalam ruangan. Dengan menggunakan sistem penjejakan garis, drone dapat secara otomatis menentukan rute patroli dalam ruangan berdasarkan garis yang dibuat pada permukaan lantai. Sistem penjejakan garis pada *indoor patrolling drone* dibuat dengan menggunakan pengindraan visual komputer. Dalam melakukan take-off dan landing *indoor patrolling drone*

masih perlu dilakukan secara manual. Ketika *indoor patrolling drone* sudah mencapai ketinggian yang diinginkan, pengguna dapat beralih pada mode otomatis. Pada mode otomatis, drone dapat melakukan navigasi dalam ruangan dengan menggunakan sistem penjejakan garis. Rute terbang drone ditentukan berdasarkan garis yang terdeteksi.Spesifikasi *indoor patrolling drone* secara lengkap disajikan pada tabel 3.1 berikut.

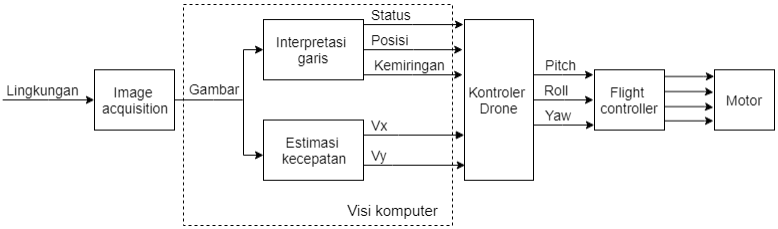
Tabel 3.1 Spesifikasi *indoor patrolling drone*

Parameter	Nilai
Flight Controler	Jenis = Omnibus F4 v6 mikrokontroler = STM32F405RGT6 Frekuensi clock = 168Mhz RAM = 192KB Flash = 1024KB Firmware = INAV
Single Board Computer	Jenis = Raspberry Pi 3b Frekuensi clock = 1.2GHz RAM = 1GB SDcard = 16GB OS = Raspbian
Mikrokontroler	Jenis = STM32F103C8T6 bluepill development board Frekuensi clock = 72Mhz Kapasitas flash = 64KB Kapasitas RAM = 20KB
Kamera	Jenis = Raspberry Pi camera Sensor = Sony IMX219 Resolusi video = 1080p 30fps, 720p 60fps dan 640x480p 90fps Konektor = CSI
Brushless motor	Jenis = Racerstar BR2305S Rotasi = 2xCW, 2xCCW KV rating = 2600KV LiPo = 2-5 sel Thrust = 1120gram Efisiensi = 1.7 (g/W)
ESC	Jenis = Brushless DC motor ESC LiPo = 2-4sel

	Rating arus Mikrokontroler Protokol	= 30A, = 32bit = DShot600, DShot300, Oneshot128, Oneshot42, PWM
Propeller	Diameter Pitch Jenis	= 6inci = 4 = Triblade
Take off weight	801 gram	
Dimensi frame	Lebar Panjang Tinggi	= 265mm (rotor to rotor) = 200mm (rotor to rotor) = 145mm
Tegangan Kerja	16.8V – 14V	
Baterai	Jenis Jumlah sel Kapasitas Tegangan total Tegangan sel	= Lithium Polymer = 4 sel = 1500 mAh = 16.8V – 14V = 4.2V – 3.5V

Gambar 3.2 menunjukkan diagram blok sistem penjejakan garis pada *indoor patrolling drone.* Pada diagram tersebut terdapat empat blok utama yaitu *image acquisition,* visi komputer, kontroler drone dan *flight controller* beserta motor. Pembahasan lebih spesifik dari masing – masing bagian tersebut akan dijelaskan lebih lanjut pada subbab selanjutnya.

Proses diawali dengan proses pengambilan gambar oleh kamera. Gambar yang dihasilkan pada proses tersebut terdiri dari tiga *channel* warna yaitu BGR (*Blue, Green, Red*) dengan resolusi 640x480 pixel. Gambar tersebut kemudian menjadi input dari visi komputer. Visi komputer terdiri dari dua blok yaitu interpretasi garis dan estimasi kecepatan. Visi komputer akan menghasilkan kecepatan drone pada sumbu x dan sumbu y, beserta parameter garis yang terdeteksi. Parameter garis meliputi status garis yaitu ada atau tidaknya garis yang terdeteksi, kemiringan garis dan posisi garis. Hasil output dari visi komputer akan menjadi input dari kontroler drone. Kontroler drone akan memberikan sinyal kontrol pada *flight controller* berupa *pitch, roll* dan *yaw*. Pada *flight controller* terdapat proses yang akan mengolah sinyal – sinyal input tersebut untuk mengatur kecepatan putar masing – masing motor yang terdapat pada drone. Dengan demikian *flight controller* dapat mengatur karakteristik terbang drone agar sesuai dengan sinyal input yang diberikan.



Gambar 3.2 Diagram blok sistem

3.1. Image acquisition

Image acquisition atau proses pengambilan gambar adalah suatu proses yang dilakukan pada sistem penjejakan garis untuk mendapatkan gambar garis dan lingkungan. Proses pengambilan gambar dilakukan dengan menggunakan kamera yang terhubung dengan perangkat Raspberry Pi. Kamera yang digunakan adalah Raspberry pi camera V2.0 dan terhubung dengan Raspberry Pi melalui konektor CSI. Proses ini akan menghasilkan gambar dengan resolusi 640x480pixel dengan tiga *channel* warna. Gambar yang diperoleh dari proses ini akan diproses lebih lanjut pada bagian visi komputer.

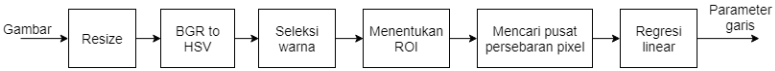
Kamera yang digunakan pada proses *image acquisition* ini memungkinkan pengambilan gambar pada tiga mode operasi, yaitu 3280 x 2464 pixel dan video dengan resolusi 1080p pada 30fps, 720p pada 60fps dan 640x480p pada 90fps. Pada proses ini mode yang digunakan adalah 640x480p pada 90fps. Meskipun pada mode operasi tersebut memungkinkan pengambilan gambar dengan *frame rate* sebesar 90fps, namun dikarenakan keterbatasan tenaga pemrosesan pada raspberry pi, proses *image acquisition* hanya dapat mencapai 22fps.

3.2. Visi komputer

Visi komputer terdiri dari dua proses utama yaitu interpretasi garis dan estimasi kecepatan. Visi komputer adalah suatu metode yang dilakukan untuk memproses gambar yang telah didapatkan dari *image acquisition* untuk mendapatkan bebrapa informasi kecepatan terbang drone dan parameter garis. Kedua proses tersebut akan dibahas lebih lanjut pada kedua subbab berikut.

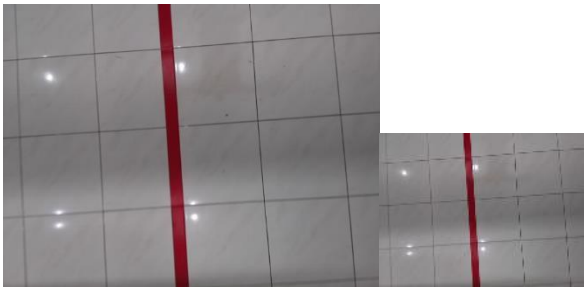
3.2.1. Interpretasi Garis

Proses interpretasi garis dilaukan untuk mendeteksi ada atau tidaknya garis pada suatu gambar. Selain untuk mendeteksi ada atau tidaknya garis, proses ini juga berfungsi untuk mengetahui posisi dan kemiringan garis yang terdeteksi. Proses interpretasi garis secara garis besar disajikan pada diagram berikut.



Gambar 3.3 Diagram proses interpretasi garis

Proses interpretasi garis diawali dengan memperkecil gamabr yang ditangkap oleh kamera. Raspberry pi kamera dapat menangkap gambar dalam beberapa pilihan resolusi yaitu 1080p, 720p dan 640x480p. Sistem penjejakan garis tidak memerlukan gambar dengan resolusi yang tinggi, sehingga pengambilan gambar dilakukan pada resolusi 640x480p.



(a) *(b)*

Gambar 3.4 *(a)* Gambar input dan *(b)* gambar setelah *resize*

Gambar yang diambil dengan resolusi 640x480 pixel selanjutnya diperkecil menjadi 160x120 pixel. Langkah tersebut bertujuan untuk mengurangi beban perhitungan yang yang harus dilakukan oleh Raspberry Pi.

Setelah gambar input diperkecil, langkah selanjutnya adalah mengkonversikan gambar dari domain warna RGB menjadi domain HSV. Langkah ini diperlukan karena domain warna RGB memiliki nilai yang dapat dipengaruhi oleh kamera yang digunakan dan tidak meratam . Domain warna HSV dipilih karena domain HSV mendeskripsikan warna

pada gambar menyerupai pengelihatan pada manusia. Pada domain HSV pemilihan warna dapat dilakukan pada satu komponen gambar yaitu Hue. Hal tersebut dapat mempermudah proses dalam melakukan seleksi warna pada langkah selanjutnya.

Gambar yang telah dikonversikan pada domain HSV akan diproses lebih lanjut untuk melakukan seleksi warna. Seleksi warna dilakukan untuk memisahkan garis dengan gambar latar. Proses seleksi warna dilakukan dengan metode thresholding. Metode thresholding akan menghasilkan gambar yang berisikan pixel-pixel penyusun garis yang terdeteksi.

Gambar 3.5 Gambar *binary* hasil seleksi warna

Setelah dilakukan proses seleksi warna, langkah selanjutnya adalah menentukan Region of Interest (ROI). Gambar yang telah dilakukan threshold akan dibagi menjadi 10 ROI. ROI pertama memiliki ukuran 160x12 pixel.

Gambar 3.6 **Pemilihan ROI pertama**

Gambar 3.7 menunjukkan seleksi daerah ROI. Selanjutnya, dilakukan analisa pada region yang terpilih untuk menentukan pusat

persebaran pixel. Untuk dapat menemukan pusat perseberan pixel menggunakan metode momen pixel yang telah dijelaskan pada bab 2.

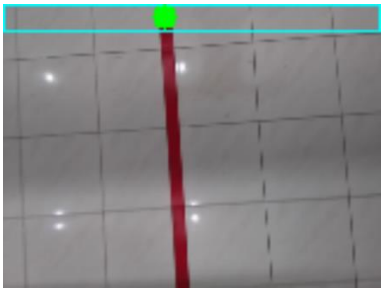
$$M_x = \sum_{i=1}^{160} \sum_{j=1}^{12} i \, . \, p_{(i,j)} \tag{3.6}$$

$$M_y = \sum_{j=1}^{12} \sum_{i=1}^{160} j \, . \, p_{(i,j)} \tag{3.7}$$

$$x = \frac{M_x}{\sum_{i=1}^{160} \sum_{j=1}^{12} p_{(i,j)}} \tag{3.8}$$

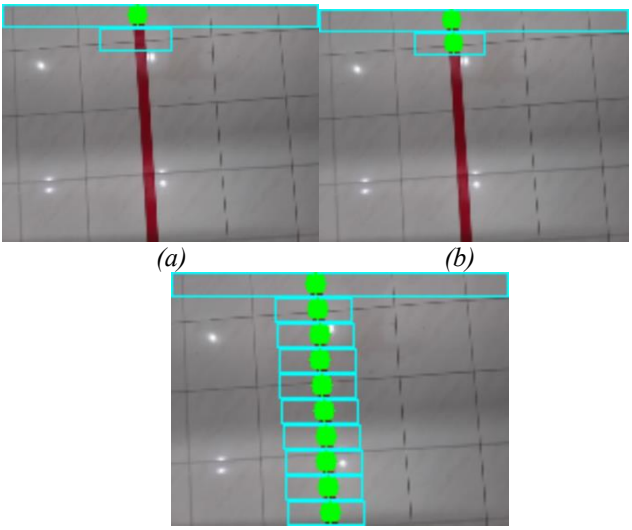
$$y = \frac{M_y}{\sum_{i=1}^{160} \sum_{j=1}^{12} p_{(i,j)}} \tag{3.9}$$

Dengan menggunakan persamaan tersebut didapatkan titik pusat dengan koordinat (x,y). Titik kooardiant tersebut ditampilkan pada gambar 3.7 berikut.



Gambar 3.7 Titik pusat persebaran pixel

Hingga langkah ini, telah didapatkan satu titik yang akan digunakan untuk melakukan estimasi garis. Titik yang sudah didapatkan selanjutnya digunakan untuk menentukan posisi ROI yang selanjutnya. ROI ke-dua, ke-tiga dan seterusnya memiliki ukuran 36x12 pixel. Persamaan berikut digunakan untuk menentukan posisi ROI baru berdasarkan pusat persebaran pixel pada ROI sebelumnya.

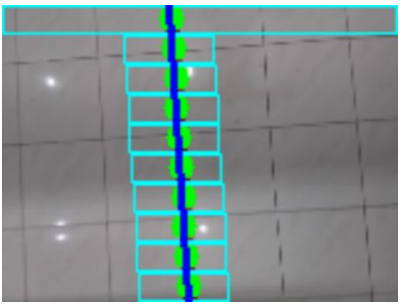


(a) *(b)*

(c)

Gambar 3.8 *(a)*Pemilihan lokasi ROI ke-dua *(b)*Penentuan pusat persebaran pixel ROI ke-dua *(c)*Kumpulan *datapoint* regresi

Kumpulan titik pusat persebaran pixel pada masing – masing ROI selanjutnya digunakan untuk melakukan estimasi garis dengan menggunakan metode regresi linier. Metode regresi linier menghasilakan nilai a dan b dimana persamaan garis y=ax+b.



Gambar 3.9 Hasil regresi linear

Proses interpretasi garis menghasilkan dua output berupa kemiringan garis dan jarak garis terhadap pusat frame. Jarak garis hanya diukur pada sumbu horizontal. Untuk mendapatkan jarak garis dari persamaan regresi y = ax+b, digunakan persamaan sebagai berikut, dimana w adalah lebar frame gambar dan h tinggi frame gambar.

$$jarak = \frac{w}{2} - \frac{y - b}{a}; y = \frac{h}{2} \tag{2.10}$$

Pada sistem ini kemiringan garis didefinisikan sebagai jarak x1 dan x2, dimana x1 dan x2 didefinisikan sebagai berikut.

$$x_1 = \frac{y_1 - b}{a};\ y_1 = 0 \tag{2.11}$$

$$x_2 = \frac{y_2 - b}{a};\ y_2 = \frac{h}{2} \tag{2.12}$$

Berdasarkan definisi diatas, kemiringan garis dapat dituliskan sebagai berikut.

$$kemiringan = jarak\ x_1 x_2 = x_1 - x_2 \tag{2.13}$$

$$kemiringan = \frac{y_1 - b}{a} - \frac{y_2 - b}{a} \tag{2.14}$$

$$kemiringan = \frac{y_1 - y_2}{a} \tag{2.15}$$

$$kemiringan = \frac{2y_1 - h}{2a} \tag{2.16}$$

3.2.2. Estimasi Kecepatan

Sistem estimasi kecepatan dilakukan untuk memperkirakan kecepatan terbang drone pada sumbu x dan sumbu y. Sistem estimasi kecepatan pada *indoor patroling drone* menggunakan metode lucas-kanade *optical flow.* Metode lucas-kanade diawali dengan menentuka titikyang akan digunakan pada *sparse optical flow*. Pemilihan titik – titik tersebut dilakukan secara otomatis. Pemilihan titik menggunakan *harris*

corner detection method. Sparse optical flow pada sistem ini dibatasi sebanyak sepuluh titik.

Setelah kumpulan titik telah ditentukan, algoritma *lucas-kanade* akan menghasilkan vector perpindahan dari masing – masing titik tersebut. Untuk mendapatkan estimasi perpindahan drone, digunakan nilai rata – rata kumpulan vetor perpindahan yang telah didapatkan.

$$D_x = \frac{\sum_1^N d_{Nx}}{N} \tag{3.2}$$

$$D_y = \frac{\sum_1^N d_{Ny}}{N} \tag{3.3}$$

Persamaan diatas digunakan untuk mendapatkan estimasi perpindahan drone pada sumbu x (D_x) dengan menggunakan N titik yang telah ditentukan. Untuk mendapatkan estimasi kecepatan *drone* berasarkan perpindahan drone digunakan persamaan berikut :

$$V_x = D_x \,.\, FPS \tag{3.4}$$

$$V_y = D_y \,.\, FPS \tag{3.5}$$

Persamaan diatas digunakan untuk mendapatkan estimasi kecepatan pada sumbu x (V_x) dan kecepatan pada sumbu y (V_y) dengan menggunakan nilai perpindahan dan nilai *Frame per Second* (FPS) video.

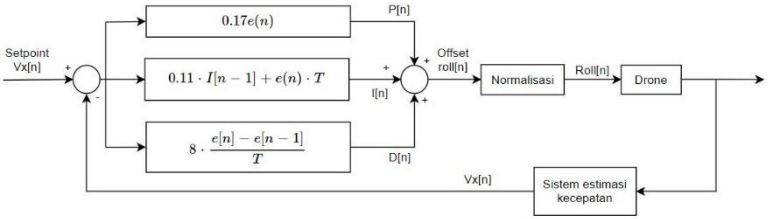
3.3. Kontroler drone

Kontroler drone terdiri dari dua bagian yaitu kontrol kecepatan dan kontrol penjejakan garis. Masing masing kontroler tersebut akan dibahas lebih lanjut pada subbab berikut.

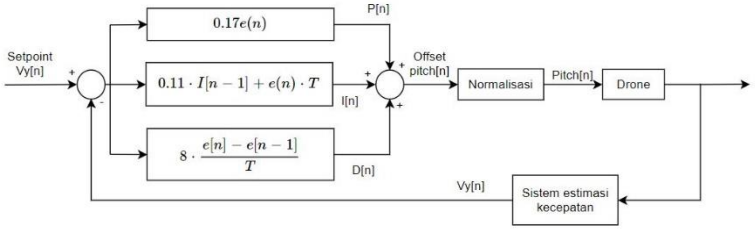
3.3.1. Kontrol kecepatan

Kontrol kecepatan berfungsi untuk mengatur karakteristik terbang drone agar dapat mempertahankan kecepatan pada nilai yang diinginkan. Vektor kecepatan pada *indoor patroling drone* terbagi menjadi dua komponen, yaitu kecepatan pada sumbu X dan kecepatan pada sumbu Y. Kontrol kecepatan pada pada *indoor patroling drone* direalisasikan dengan menggunakan PID kontroler. Terdapat dua PID kontroler yang

terdapat pada sistem kontrol kecepatan yaitu kontroler PID kecepatan terhadap sumbu x dan sumbu y. Kedua PID kontroler yang digunakan berdasarkan diagram blok yang sudah dijelaskan pada bab dua, dengan menambahkan sedikit modifikasi. Beberapa modifikasi tersebut adalah penambahan blok normalisasi. Diagram kontroler PID kecepatan pada *indoor patroling drone* disajikan dalam bentuk diagram sebagai berikut.

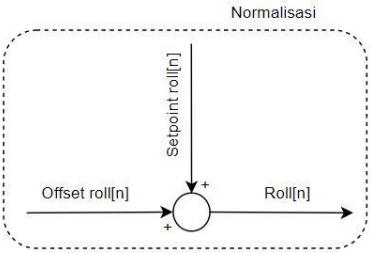


Gambar 3.10 Kontroler PID kecepatan sumbu x



Gambar 3.11 Kontroler PID kecepatan sumbu y

Pada kedua PID kontroler diatas, terdapat vaiabel T yaitu *sampling time* sebesar 30ms. Blok normalisasi pada kedua PID kontroler diatas berfungsi untuk menyesuaikan range nilai output PID kontroler agar sesuai dengan format nilai pada R/C system. Pada sebagian besar R/C system sinyal kontrol yang digunakan memiliki range nilai 1000 hingga 2000, dengan nilai tengah sebesar 1500. Nilai tengah tersebut tidak selalu berada pada posisi 1500, dalam beberapa kasus nilai tengah dapat digeser dengan menggunakan trimmer yang terdapat pada R/C transmitter. Range nilai pada R/C sistem berdeda dengan nilai output kontroler PID. Pada kontroler PID yang digunakan sinyal output memiliki range -200 hingga 200, dengan nilai tengah 0. Oleh karena itu, perlu adanya penyesuaian range nilai yang direalisasikan dalam bentuk blok normalisasi.



Gambar 3.12 Blok normalisasi kontroler PID kecepatan sumbu x

Pada gambar 3.12 diatas dapat dilihat bahwa sinyal referensi merupakan nilai tengah pada R/C system yang digunakan. Sinyal referensi akan dikirimkan oleh R/C receiver kepada mikrokontroler dan pada akhirnya diterima oleh Raspberry Pi. Secara singkat blok normalisasi dapat direpresentasikan dengan menggunakan persamaan sebagai berikut.

$$Norm\big(u(t)\big) = u(t) + ref(t)$$

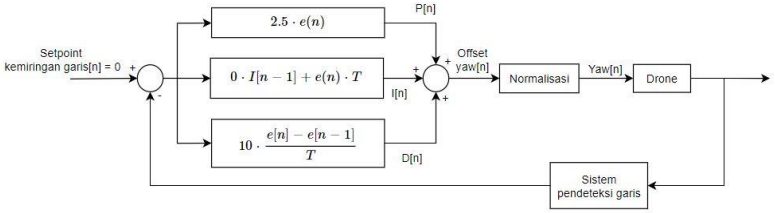
Pada gambar 3.11 juga dijelaskan bahwa sinyal feedback didapatkan dari speed estimation system. Speed estimation system adalah suatu sistem pada *indoor patroling drone* yang digunakan untuk mengetahui kecepatan drone pada sumbu x dan sumbu y. Speed estimation system memberikan sinyal keluaran kecepatan dalam satuan cm/s.

Output dari sistem kontrol kecepatan berupa sinyal pitch dan roll. Kontroler PID kecepatan x akan memberikan output sinyal roll dan kontroler PID kecepatan y akan memberikan output sunyal pitch. Sinyal roll dan pitch kemudian dikirimkan pada flight kontroler untuk mengatur sudut terbang drone pada sumbu pitch dan roll.

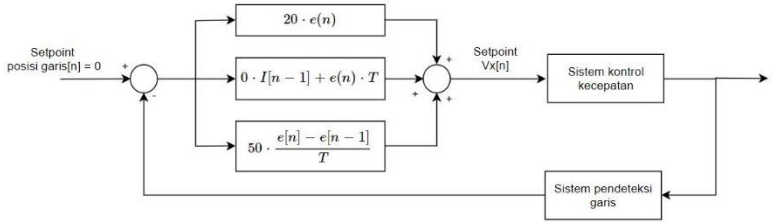
Dalam setiap kontroler PID tentunya perlu dilakukan proses tuning pada model kontroler yang dibuat. Proses tuning dilakukan untuk mendapatkan parameter – parameter yang ideal. Pada kontrole PID parameter tersebut meliputi Kp, Ki dan Kd. Terdapat beberapa metode yang dapat dilakukan untuk melakukan proses tuning pada kontroler PID. Dalam pembuata *indoor patrolling drone* metode tuning yang digunakan adalah metode *trial and error.* Metode trial and error telah dijelaskan pada bab dua.

3.3.2. Kontrol penjejakan garis

Kontrol penjejakan garis berfungsi untuk mengatur karakteristik penerbangan drone sehingga dapat bernavigasi pada lintasan garis yang telah ditentukan. Terdapat dua komponen dalam sistem kontrol penjejakan garis, yaitu posisi drone terhadap garis dan sudut drone terhadap arah garis. Kontrol penjejakan garis pada *indoor patroling drone* direalisasikan dengan menggunakan PID kontroler. Terdapat dua PID kontroler yang terdapat pada sistem kontrol penjejakan garis yaitu kontroler PID posisi dan sudut drone. Kedua PID kontroler yang digunakan berdasarkan diagram blok yang sudah dijelaskan pada bab dua, dengan menambahkan beberapa modifikasi. Modifikasi tersebut berupa penambahan blok normalisasi pada kontroler PID kemiringan garis. Kedua kontroler PID tersebut disajikan pada dua gambar berikut.



Gambar 3.13 Kontroler PID kemiringan garis



Gambar 3.14 Kontroler PID posisi garis

Gambar 3.13 menunjukkan diagram kontroler PID kemiringan garis. Kontroler PID ini hampir sama dengan konroler PID kecepatan. Perbedaan utama antara kontroler PID sudut dengan kontroler PID kecepatan adalah metode pengambilan sinyal feedback. Pada kontroler PID kecepatan sinyal feedback didapatkan dari proses estimasi kecepatan sedangkan pada kontroler sudut didapatkan dari proses interpretasi garis.

Proses interpretasi garis memiliki dua output yaitu sudut dan posisi garis. Sudut garis digunakan sebagai sinyal feedback pada kontroler PID ini dan posisi garis akan digunakan pada kontroler PID posisi drone terhadap garis.

Pada gambar 3.14 menunjukkan bahwa sinyal kontrol PID posisi akan digunakan sebagai setpoint pada kontroler PID kecepatan. Sinyal kontrol yang dihasilakan oleh kontroler PID posisi merupakan perintah kecepatan dengan satuan cm/s.

3.4. Flight controller

Flight controller adalah perangkat yang berfungsi untuk mengendalikan drone/UAV. *Flight controller* mendapatkan input dari berbagai sensor yang terdapat pada drone diantaranya; gyroscope, accelerometer, magnetometer dan barometer. *Flight controller* akan melakukan penyesuaian kecepatan putar masing – masing motor sehingga menghasikan daya dorong yang sesuai agar drone dapat memiliki karakteristik terbang yang bersesuaian dengan sinyal kontrol yang diterima. Perangkat *flight controller* yang digunakan pada drone ini adalah omnibus F4 V6. Omnibus F4 V6 merupakan salah satu *open source flight controller* yang dikembangkan oleh Airbot. Omnibus F4 V6 memiliki prosesor utama berupa mikrokontroler STM32F405RGT6.

Pada sistem ini, *flight controller* menerima input dari kontroler drone yang telah dibahas sebelumnya. Terdapat tiga sinyal kontrol utama yaitu pitch, roll dan yaw. Masing – masing sinyal kontrol tersebut mengatur rotasi drone pada sumbu yang bersesuaian.

BAB 4
PENGUJIAN DAN ANALISIS

Pada bab 4 ini akan dibahas mengenai pengujian dari berbagai bagina dari sistem *indoor patrolling drone.* Pengujian bertujuan untuk mendapatkan data yang selanjutnya akan dilakukan analisis terhadap data tersebut. Pengujian pada bab ini meliputi pengujian pembacaan ketinggian BMP280, pengujian *flight time,* pengujian regresi jarak, pengujian deteksi garis, pengujian kontroler PID kecepatan, pengujian kontroler PID posisi garis dan pengujian kontroler PID kemiringan garis.

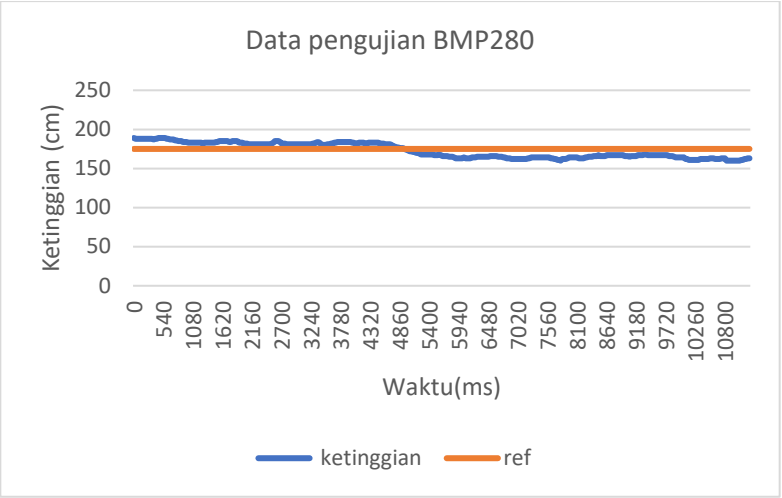
4.1. Pengujian Pengukuran Ketinggian BMP280

Sensor BMP280 digunakan mendapatkan ketinggian terbang drone. Data ketinggian drone kemudian digunakan pada proses regresi jarak. Ketinggian yang didapatkan dari pembacaan sensor BMP280 merupakan ketinggian relatif terhadap ketinggian saat dilakukan proses arming. Pengujian sensor BMP280 dilakukan untuk mengetahui akurasi pembacaan ketinggian yang diberikan oleh sensor BMP280.

Pengujian sensor BMP280 diawali dengan melakukan proses arming ketika drone berada diatas permukaan lantai, dengan demikian ketinggian referensi akan mengacu pada ketinggian diatas permukaan lantai. Langkah selanjutnya dilakukan dengan meletakkan drone pada ketinggian yang sudah ditentukan, nilai pembacaan ketinggian kemudian disimpan dalam bentuk file csv. Hasil pembacaan ketinggian disajikan dalam bentuk grafik dan dilakukan analisa pada grafik yang tersebut. Proses pengambilan data ketinggian dilakukan pada ketinggian 15 hingga 200cm diatas permukaan lantai.

Gambar 4.1 menunjukkan plot pembacaan ketinggian oleh sensor BMP820 pada ketinggian 50cm. Berdasarkan hasil pembacaan tersebut didapatkan nilai akurasi sebesar ±10cm. Percobaan diatas dilakukan kembali pada ketinggian yang berbeda – beda. Hasil pembacaan sensor BMP280 secara ringkas disajikan pada tabel 4.1.

Grafik data masing – masing ketinggian pada tabel 4.1 terlampir pada lampiran bagian A. Berdasarkan data yang telah diperoleh, pengukuran ketinggian dengan sensor BMP280 memiliki nilai tingkat maximum absolut error sebesar ±16.56cm.



Gambar 4.1 plot pembacaan ketinggian pada 50cm

Tabel 4.1 Tabel pengujian sensor BMP280

Ketinggian (cm)	Nilai minimum (cm)	Nilai maksimum (cm)	Max|error| (cm)
15	7	26	±19
25	19	39	±14
50	43	56	±7
75	70	93	±18
100	76	108	±24
125	118	145	±20
150	134	152	±16
175	160	189	±15
200	184	203	±16
Rata - rata			±16.56cm

4.2. Pengujian *Flight Time*

Pengujian *flight time* dilakukan dengan tujuan mengetahui daya tahan batrai yang digunakan. Dalam pengujian ini batrai yang digunakan berjenis LiPo 4 sel dengan kapasitas 1500mAh.

Gambar 4.2 Proses pengambilan data *flight time*

Gambar 4.2 menunjukkan proses pengambilan data *flight time*. Pengujian *flight time* diawali dengan mengisi daya batrai hingga penuh, kemudian drone diterbangkan dengan batrai tersebut dalam posisi *hovering*. Waktu penerbangan dimulai ketika drone mulai meninggalkan lantai hingga alarm penanda daya batrai lemah berbunyi yang menunjukkan tegangan batrai sudah mencapai 14V pada nilai tegangan tersebut daya batrai yang telah terpakai adalah sebesar 1258mAH atau 83.87% kapasitas maksimum. Percobaan tersebut dilakukan sebanyak lima kali. Waktu penerbangan pada masing masing percobaan disajikan pada tabel 4.2.

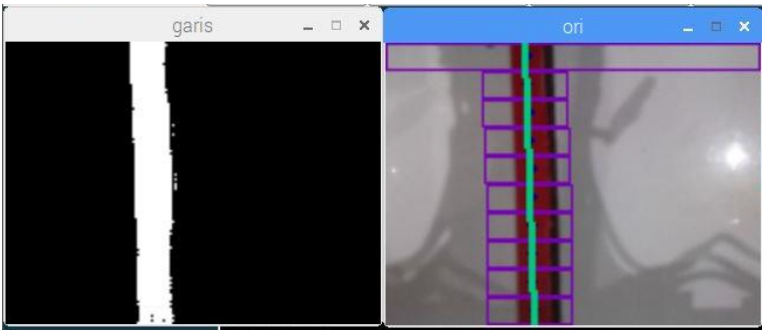
Tabel 4.2 Data pengujian waktu terbang drone

Percobaan	Waktu
1	6menit 59detik
2	7menit 05detik
3	6menit 49detik
4	6menit 30detik
5	7menit 18detik
Rata - rata	6menit 56.2detik

Berdasarkan data yang sudah diperoleh, didapatkan waktu terbang rata – rata yaitu selama 6menit 56.2detik.

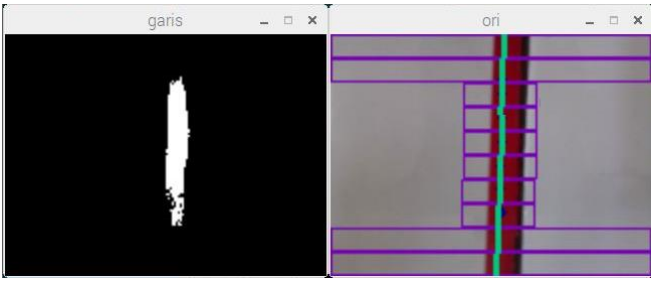
4.3. Pengujian Proses interpretasi garis

Dalam subbab ini akan dilakukan pengujian terhadap proses interpretasi garis dalam kondisi yang tidak optimal. Kondisi tersebut meliputi beberapa situasi yang mungkin ditemui ketika *indoor patrolling drone* melakukan navigasi penjejakan garis. Proses evaluasi kemudian dilakukan guna mengetahui performa proses interpretasi garis pada situasi keadaan ideal maupun pada beberapa situasi yang dianggap tidak ideal.



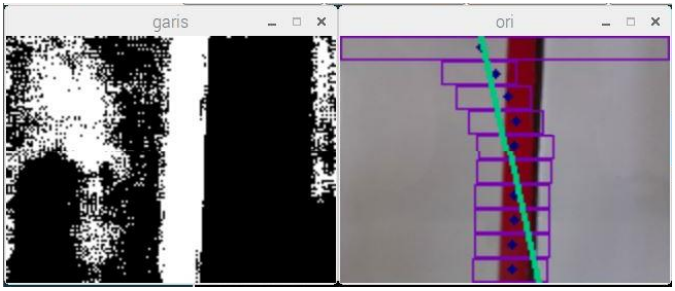
Gambar 4.3 kondisi optimal

Gambar 4.3 menunjukkan kondisi garis ideal. Pada kondisi ideal, garis dapat terdeteksi secara sempurna pada gambar binary hasil seleksi warna. Pada kondisi ideal ini proses interpretasi garis dapat melakukan estimasi garis dengan baik.



Gambar 4.4 kondisi tidak optimal 1

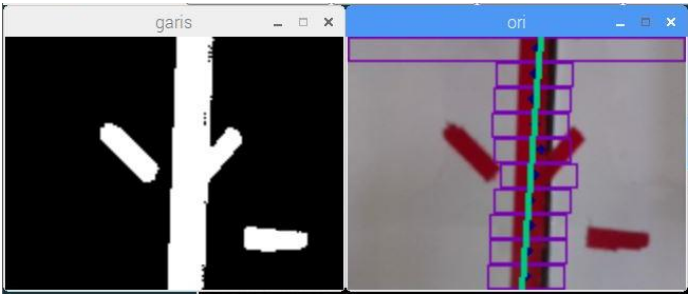
Gambar 4.4 menunjukkan kondisi tidak optimal yang pertama. Dalam kondisi ini gambar garis tidak terdeteksi secara sempurna. Pada gambar *binary* terlihat tepian garis terkikis dan tidak ditampilkan secara sempurna. Kondisi ini dapat disebabkan oleh pemilihan range seleksi warna yang terlalu sempit. Kondisi tersebut juga dapat disebabkan oleh variasi perubahan pencahayaan garis. Proses interpretasi garis garis masih dapat memberikan estimasi garis dengan cukup baik pada konsisi ini. Meskipun pada beberapa ROI sistem tidak dapat menemukan pusat persebaran pixel, regresi garis masih bisa memberikan prediksi garis yang memadai.



Gambar 4.5 kondisi tidak optimal 2

Gambar 4.5 menunjukkan kondisi tidak optimal yang kedua. Dalam kondisi ini terdapat banyak noise pada gambar hasil seleksi warna. Kondisi ini dapat disebabkan karena range seleksi warna terlalu lebar. Kondisi tersebut juga dapat disebabkan oleh perubahan konsi pencahayaan. Berdasarkan hasil pada gambar 4.6, pada kondisi ini proses

interpretasi garis kurang bisa mengestimasi garis dengan benar.



Gambar 4.6 kondisi tidak optimal 3

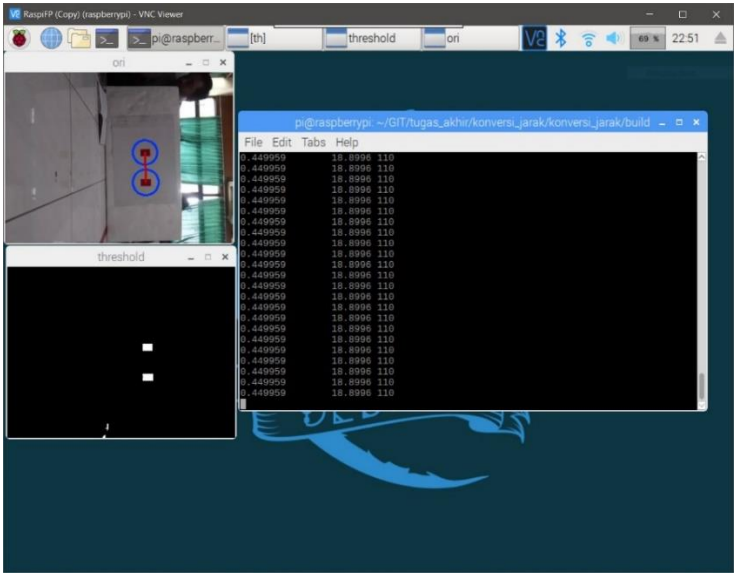
Gambar 4.6 menunjukkan kondisi tidak optimal yang ketiga. Kondisi ini menggambarkan keadaan ketika terdapat object atau benda lain yang memiliki warna menyerupai warna garis yang digunakan. Pada kondisi ini object tersebut juga akan muncul pada gambar hasil seleksi warna. Dalam kasus seperti ini ROI juga berperan dalam membantu memastikan area pencarian garis hanya terfokus pada area disekitar garis yang diinginkan. Hal tersebut membantu proses interpretasi garis untuk menghiraukan objek diluar garis, meskipun memiliki warna yang sama. Berdasarkan hasil yang ditunjukkan pada gambar 4.7, algoritma pendeteksi garis mampu memberikan estimasi garis yang baik meskipun terdapat objek yang berpotensi mengganggu sistem pendeteksi objek.

4.4. Pengujian Regresi Jarak

Pengujian regresi jarak diawali dengan pengambilan data. Proses pengambilan data bertujuan untuk mendapatkan data point yang akan digunakan dalam regresi. Dalam melakukan regresi jarak dibutuhkan dua macam data yaitu input jarak dalam pixel dan ketinggian drone. Dengan melakukan pemrosesan terhadap kedua input tersebut bisa didapatkan output berupa jarak dalam satuan centimeter.

Gambar 4.7 proses pengambilan data

Proses regresi jarak dimulai dengan pengambilan data kedua input yang telah disebutkan. Gambar 4.7 menunjukkan proses pengambilan data regresi. Data yang diambil berupa jarak (dalam pixel) antara dua obeject yang terpisah sejauh 28.8cm.



Gambar 4.8 pengukuran dua objek

Gambar 4.8 menunjukkan proses pengukuran jarak dengan menggunakan kamera. Proses pengukuran tersebut dilakukan sebanyak beberapa kali dengan melakukan variasi jarak objek terhadap kamera.

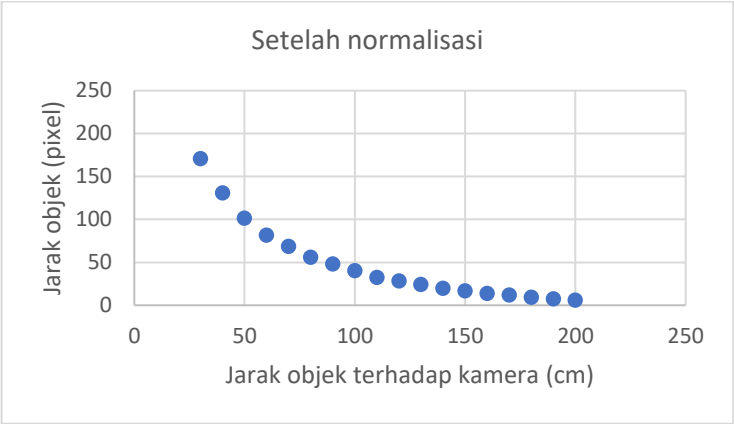
Tabel 4.3 Data pengukuran jarak

Jarak objek terhadap kamera	Pengukuran jarak kedua objek
30 cm	200.94 pixel
40 cm	161.00 pixel
50 cm	131.70 pixel
60 cm	111.94 pixel
70 cm	98.70 pixel
80 cm	86.15 pixel
90 cm	78.16 pixel
100 cm	70.63 pixel
110 cm	62.60 pixel
120 cm	58.58 pixel
130 cm	54.61 pixel
140 cm	50.10 pixel
150 cm	47.05 pixel
160 cm	44.07 pixel
170 cm	42.09 pixel
180 cm	39.55 pixel
190 cm	37.58 pixel
200 cm	36.04 pixel

4.4.1 Pemilihan model regresi

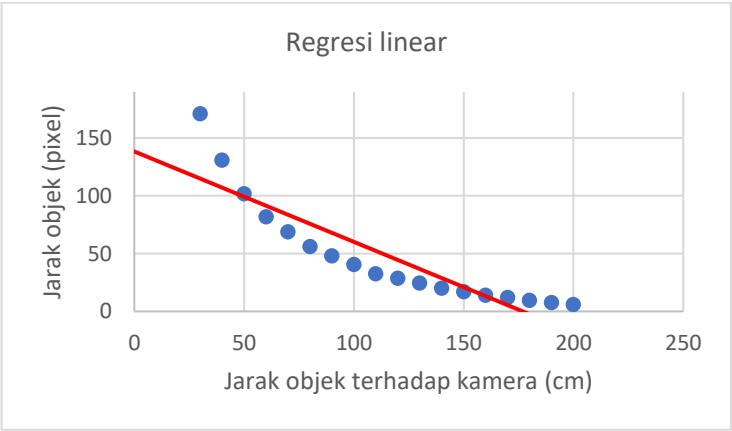
Langkah selanjutnya dalam regresi adalah menentukan model regresi. Regresi dapat dilakukan dengan beberapa model matematis yang berbeda. Dalam menentukan model regresi terbaik maka dilakukan percobaan dengan menggunakan pada masing – masing model matematis. Untuk melakukan evaluasi terhadap model matematis yang sedang diujikan, dapat menggunakan nilai Sum of Residual Error (SSR). Model dengan nilai SSR terkecil menggambarkan estimasi terbaik. Sebelum dilakukan pengujian ini, terlebih dahulu dilakukan proses normalisasi pada data

yang telah didapatkan sebelumnya.



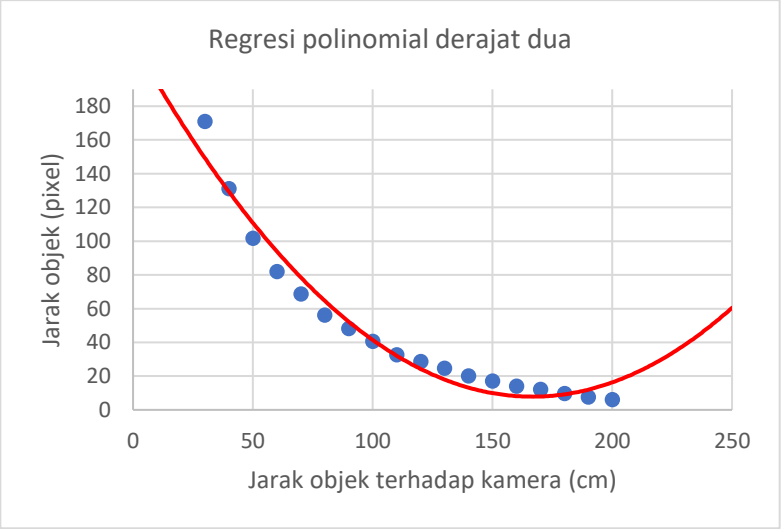
Gambar 4.9 Plot data setelah normalisasi

Gambar 4.9 menunjukkan data yang telah didapatkan sebelumnya setelah proses normalisasi. Tujuan dan pengujian proses normalisasi akan dijelaskan pada langkah 4.4.2. Pengujian pertama dilakukan terhadap fungsi linear dengan bentuk umum y = ax+b.



Gambar 4.10 Regresi linear

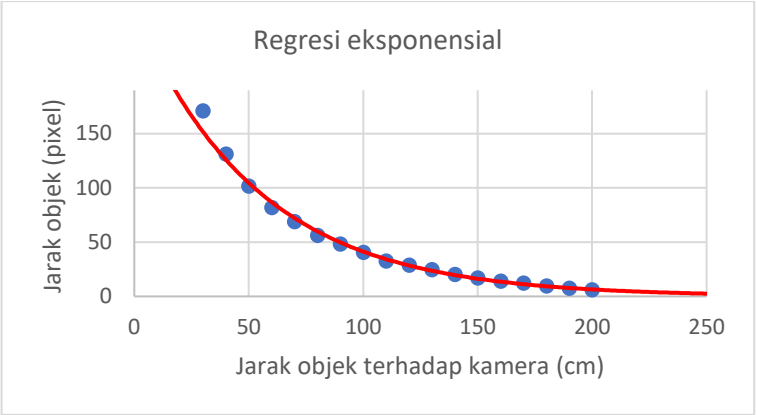
Berdasarkan gambar 4.10, dapat dilihat bahwa fungsi linear tidak dapat melakukan estimasi pada data secara akurat. Hal tersebut juga dapat ditinjau pada nilai SSE yang cukup besar yaitu 7159.58. Pada pengujian selanjutnya akan dilakukan regresi polynomial derajat dua dengan bentuk umum $y = ax^2+bx+c$.



Gambar 4.11 regresi polynomial derajat dua

Berdasarkan gambar 4.11, dapat dilihat bahwa fungsi polynomial derajat dua juga belum dapat melakukan estimasi pada data secara akurat. Regresi polynomial derajat dua menghasilkan nilai SSE sebesar 1227.16. Pada pengujian selanjutnya akan dilakukan regresi eksponensial dengan bentuk umum $y = \beta e^{\alpha x}$.

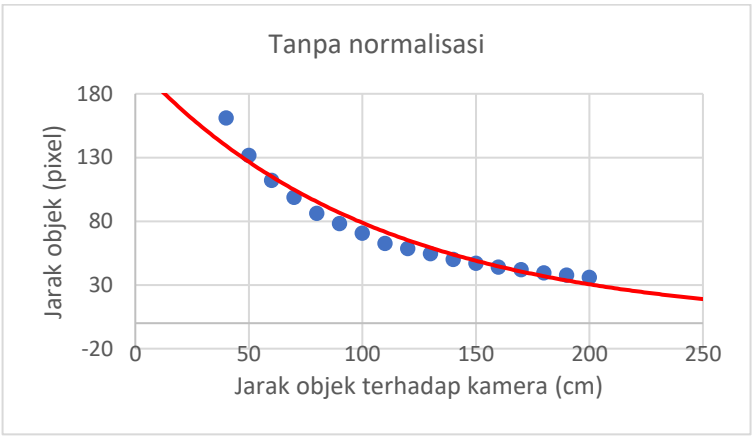
Berdasarkan gambar 4.12, secara kasat mata regresi polynomial mampu memberikan estimasi terhadap data point dengan cukup baik. Regresi eksponensial memberikan nilai SSE sebesar 318.12. Dengan membandingkan nilai SSE dari ketiga percobaan yang telah dilakukan, regresi eksponensial memberikan nilai SSE terkecil, sehingga dapat ditarik kesimpulan bahwa diantara ketiga metode regresi yang dilakukan, regresi eksponensial mampu memberikan estimasi terbaik.



Gambar 4.12 regresi eksponensial

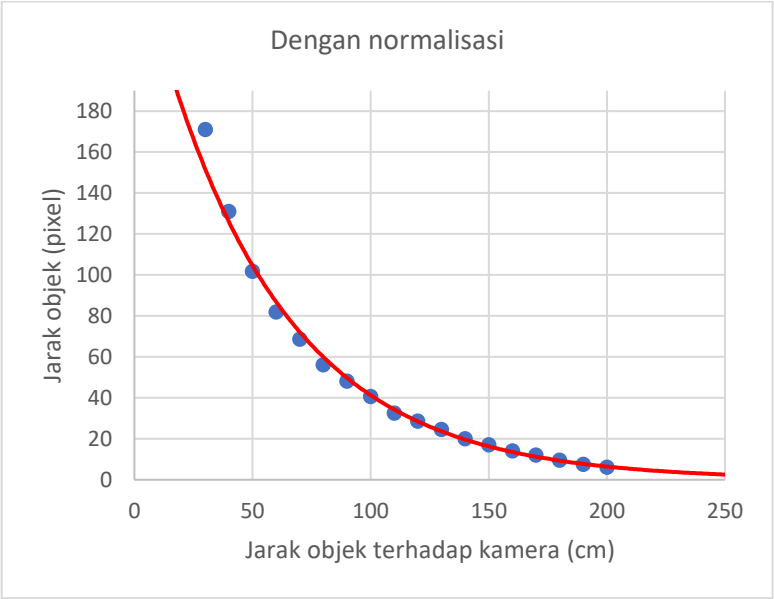
4.4.2 Pengujian efek bias pada data

Pada percobaan ini akan dilakukan percobaan untuk mengetahui efek bias pada regresi polynomial. Hasil percobaan ini akan mendasari proses normalisasi dan denormalisasi pada tahap perancangan. Pada dasarnya dipercobaan ini akan dilakuk regresi eksponensial pada dataset yang sama, namun pada salah satu dataset tidak dilakukan proses normalisasi terlebih dahulu.



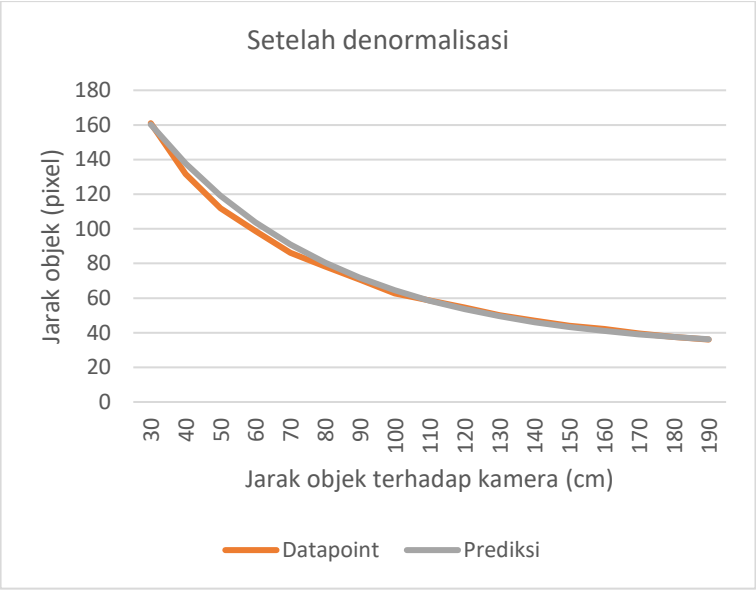
Gambar 4.13 regresi eksponensial tanpa normalisasi

Gambar 4.13 menunjukkan regresi eksponensial pada data yang memiliki bias. Regresi eksponensial dengan persamaan dasar $y = \beta e^{\alpha x}$ tidak dapat melakukan estimasi yang optimal terhadap data yang memiliki bias. Hal tersebut dibuktikan dengan nilai SSR yang tinggi yaitu 2135.26.



Gambar 4.14 regresi eksponensial dengan normalisasi

Gambar 4.14 menunjukkan hasil regresi eksponensial terhadap data yang telah melalui proses normalisasi. Berdasarkan gambar tersebut fungsi eksponensial dapat mengestimasi data dengan baik. Pada aplikasinya perlu dilakukan proses denormalisasi pada fungsi eksponnsial yang dihasilkan. Hal tersebut perlu dilakukan agar fungsi eksponensial yang dihasilkan dapat mengestimasi data input dengan baik. Dengan melakukan denormalisasi pada fungsi eksponensial maka fungsi dasar akan menjadi $y = \beta e^{\alpha x}+c$. Fungsi yang telah didapatkan, selanjutnya ditampilkan dalam bentuk grafik pada gambar 4.15.



Gambar 4.15 Denormalisasi fungsi eksponensial

4.4.3 Pengujian model regresi

Pengujian regresi digunakan untuk mengevaluasi model regresi yang telah dibuat. Pengujian dilakukan dengan konfigurasi kerja yang sama seperti sebelumnya. Modifikasi yang dilakukan hanya terdapat pada jarak kedua objek yang ditekesi. Pada langkah pengambilan data jarak kedua objek yang didetesi adalah 28.8cm pada langkah ini jarak tersebut diubah menjadi 19.5cm. Hasil pengujian tersebut disajikan pada tabel berikut.

Tabel 4.4 Hasil pengujian model regresi

Jarak	Sebenaranya	Prediksi	|Error|
30 cm	19.5 cm	20.4 cm	0.9 cm
40 cm	19.5 cm	19.4 cm	0.1 cm
50 cm	19.5 cm	18.9 cm	0.6 cm
60 cm	19.5 cm	18.8 cm	0.7 cm
70 cm	19.5 cm	18.0 cm	1.5 cm
80 cm	19.5 cm	18.5 cm	1.0 cm
90 cm	19.5 cm	18.8 cm	0.7 cm

100 cm	19.5 cm	19.2 cm	0.3 cm
110 cm	19.5 cm	18.9 cm	0.6 cm
120 cm	19.5 cm	19.4 cm	0.1 cm
130 cm	19.5 cm	19.5 cm	0 cm
140 cm	19.5 cm	19.6 cm	0.1 cm
150 cm	19.5 cm	19.8 cm	0.3 cm
160 cm	19.5 cm	20.0 cm	0.5 cm
170 cm	19.5 cm	20.1 cm	0.6 cm
180 cm	19.5 cm	20.3 cm	0.8 cm
190 cm	19.5 cm	19.6 cm	0.1 cm
200 cm	19.5 cm	19.7 cm	0.2 cm
		Rata - rata	0.51 cm

Tabel 4.4 menunjukkan hasil pengujian terhadap model regresi eksponensial $y = 278.5246148e^{-1.911796479.10^{-2}x} + 30$. Berdasarkan hasil yang telah didapatkan, model tersebut mampu memberikan estimasi dengan nilai rata - rata absolute error sebesar 0.51cm.

4.5. Pengujian Kontroler PID Kecepatan

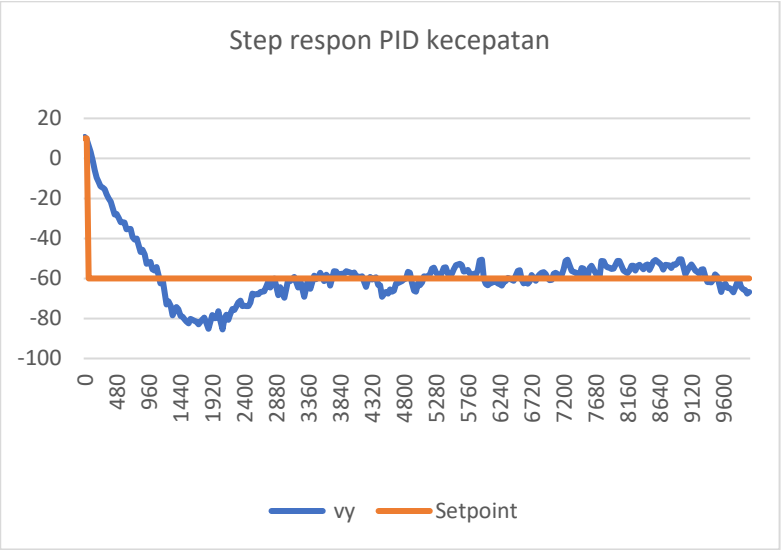
Pengujian kontroler PID dilakukan dengan mengamati step respon pada model yang dibuat. Dengan melakukan anaisa terhadap step respon, kita dapat mengevaluasi parameter – parameter yang terdapat pada kontroler tersebut.

Gambar 4.17 adalah step respons dari kontroler PID kecepatan pada *indoor* patrolling drone. Step response didapatkan denga memberikan input kecepatan sebesar -60cm/s ketika drone dalam keadaan diam. Pada pengujian ini akan dilakukan evaluasi terhadap beberapa parameter step response sebagai berikut.

- *Overshoot*
  Overshoot mendeskripsikan keadaan dimana suatu sinyal memiliki nilai yang melebihi target yang ditentukan. Nilai overshoot didefinisikan sebagai perbandingan overshoot dengan target dalam bentuk persen.
- *Steady state error*
  Steady state error adalah selisih antara sinyal dengan target ketika keadaan steady state tercapai.
- *Rise time*
  Rise time adalah waktu yang dibutuhkan sinyal dari 10% hingga

mencapai 90% steady state.

- *Setling time*
  Setling time adalah waktu yang dibutuhkan agar sinyal mencapai keadaan stabil. Keadaan stabil didefinisikan ketika sinyal memiliki error sebesar 2% dari steady state.



Gambar 4.16 Step respon kontroler PID kecepatan

Dengan melakukan analisa terhadap grafik step respons pada gambar 4.17 maka dapat diperoleh beberapa nilai parameter sebagai berikut.

Tabel 4.5 Parameter step respon PID kecepatan

Parameter	Nilai	Satuan
Overshoot	35	%
Steady state error	Mendekati 0	%
Setling time	2668	ms
Rise time	860	ms

4.6. Pengujian Kecepatan

Pengujian kecepatan dilakukan untuk mengetahui akurasi kecepatan

pada drone. Pengujian dilakukan dengan menghitung waktu yang dibutuhkan drone untuk menempuh jarak 3.5 meter. Waktu yang didapatkan kemudian digunakan untuk mengetahui kecepatan drone sebenarnya, nilai tersebut akan dibandingkan dengan nilai kecepatan yang diprogramkan pada drone. Nilai absolute error didapatkan dari selisih kecepatan drone yang terukur dengan kecepatan yang diprogramkan. Dengan melakukan pengujian ini, secara tidak langsung juga melakukan pengujian pada kontroler PID yang telah diuji sebelumnya. Pengujian kecepatan juga dapat memberikan gambaran seberapa akurat model kontroler PID yang digunakan. Hasil dari percobaan tersebut disajikan dalam bentuk tabel berikut.

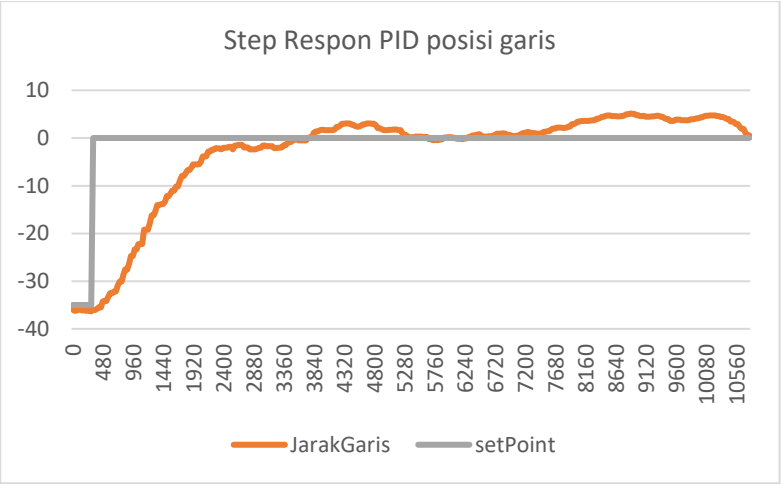
Tabel 4.6 Data pengujian kecepatan drone

Vset (cm/s)	Jarak (cm)	Tavg (s)	Vmea (cm/s)	|Error| (cm/s)
45.00	350	8.97	39.02	5.98
60.00	350	6.18	56.62	3.38
75.00	350	4.63	75.62	0.62
90.00	350	4.37	80.09	9.91
105.00	350	4.04	86.73	18.27
120.00	350	1.99	175.80	55.80
Rata - rata				15.66

Hasil pengujian menunjukkan nilai rata – rata absolute error adalah sebesar 15.66 cm/s. Berdasarkan hasil pengujian tersebut juga dapat diketahui bahwa nilai error cenderung semakin membesar ketika kecepatan yang diprogramkan semakin tinggi.

4.7. Pengujian Kontroler PID Posisi Garis

Pengujian kontroler PID dilakukan untuk mengetahui karakteristik kontroler yang sudah dibuat. Pengujian kontroler PID posisi garis dilakukan dengan melakukan pengamatan terhadap step respon. Dengan mengamati step respon, dapat diketahui beberapa parameter seperti overshoot, steady state error, settling time dan rise time. Dengan mengamati parameter – parameter tersebut dapat dilakukan evaluasi terhadap kinerja kontroler PID yang dibuat.



Gambar 4.17 step respon kontroler PID posisi garis

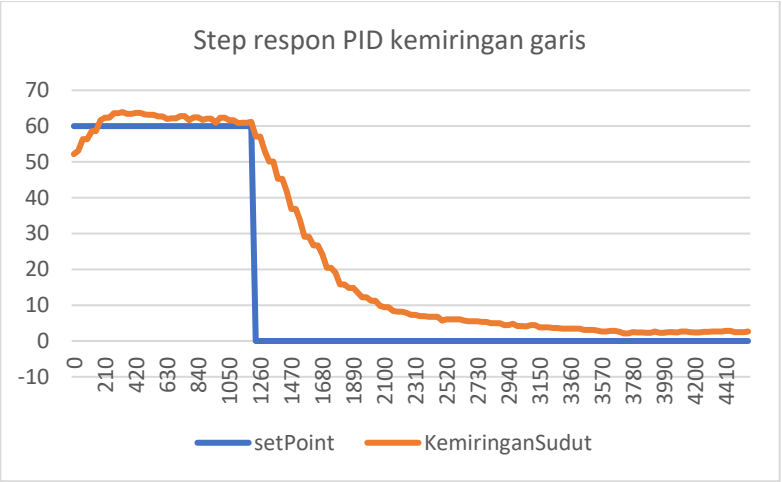
Gambar 4.17 adalah step respons kontroler PID posisi garis pada *indoor patrolling drone.* Step response didapatkan dengan memposisikan drone pada jarak 36cm dari garis. Dengan melakukan pengamatan terhadap grafik yang tersebut, maka didapatkan parameter step response sebagai berikut.

Tabel 4.7 Parameter step response PID posisi garis

Parameter	Nilai	Satuan
Overshoot	Mendekati 0	%
Steady state error	Mendekati 0	%
Setling time	2848	ms
Rise time	1500	ms

4.8. Pengujian Kontroler PID Kemiringan Garis

Kontroler PID kemiringan garis adalah kontroler PID yang berfungsi agar drone dapat memposisikan sejajar dengan daris yang terdeteksi. Sama seperti pengujian kontroler PID sebelumnya, pengujian kontroler PID kemiringan garis dilakukan dengan melakukan pengamatan terhadap step respons.



Gambar 4.18 **step respons kontroler PID kemiringan garis**

Gambar 4.18 merupakan grafik plot step respons kontroler PID kemiringan garis. Dengan melakukan pengamatan terhadap grafik tersebut, didapatkan parameter – parameter step response sebagai berikut.

Tabel 4.8 Parameter step response PID kemiringan garis

Parameter	Nilai	Satuan
Overshoot	Mendekati 0	%
Steady state error	3.79	%
Setling time	1984	ms
Rise time	1333	ms

4.9. Pengujian Keseluruhan Sistem

Pengujian sistem secara menyeluruh dilakukan untuk mengetahui mengetahui kinerja drone pada beberapa parameter kecepatan. Pegujian drone dilakukan pada tempat yang terlebih dahulu telah diberikan garis jalur navigasi drone. Pengujian ini dilakukan dengan mengikuti beberapa langkah pengoprasian drone.

Langkah pertama dalam pengoprasian drone adalah melakukan takeoff. Pada *indoor* patrolling drone yang dibuat, proses take-off masih harus dilakukan secara manual. Drone diterbangkan untuk mencapai posisi hovering dengan ketinggian kurang lebih 2m. Ketika pertama kali

melakukan proses take-off, pengguna juga dapat melakukan proses trimming pada R/C transmitter jika diperlukan. Setelah drone dapat melakukan hovering dengan stabil dan mencapai ketinggian yang diinginkan, pengguna mengaktifkan switch pada transmitter untuk mengaktifkan mode otomatis.

Setelah pengguna mengaktifkan mode otomatis melalui switch pada R/C transmitter maka drone akan memulai navigasi secara otomatis. Jika drone mendeteksi adanya garis pada kamera, maka drone akan menyesuikan posisi dan arah terbang sesuai dengan garis yang terdeteksi.

Jika drone telah mencapai ujung garis pengguna dapat menonaktifkan mode otomatis dengan mengembalikan posisi switch pada posisi sebelumnya. Untuk melakukan landing di ujung garis, *indoor* patrolling drone masih harus dioprasikan secara manual, sebagaiman telah dijelaskan pada batasan masalah. Langkah tersebut kemudian diulangi untuk beberapa nilai kecepatan. Hasil pengujian pada subbab ini disajikan dalam bentuk tabel berikut.

Tabel 4.9 Pengujian *indoor patrolling drone*

Kecepatan (cm/s)	Trial1	Trial2	Trial3
30	Berhasil	Berhasil	Berhasil
39	Berhasil	Berhasil	Berhasil
48	Berhasil	Berhasil	Berhasil
57	Berhasil	Gagal	Gagal
60	Gagal	Berhasil	Berhasil
69	Gagal	Gagal	Gagal
78	Gagal	Gagal	Gagal

Berdasarkan data yang telah didapatkan, navigasi drone dapat dilakukan secara optimal pada kecepatan 48cm/s.

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

5.1. KESIMPULAN

Berdasarkan berbagai pengujian dan pengamatan yang dilakukan pada *indoor patrolling drone* yang dibuat, didapatkan beberapa kesimpulan sebagai berikut.

1. Sensor tekanan barometric BMP280 dapat digunakan untuk mengetahui ketinggian drone dengan nilai rata – rata absolut error sebesar ±16.56 cm.
2. *Indoor patrolling drone* memiliki waktu terbang rata – rata selama 6menit 56.2 detik, dengan meggunakan batrai berjenis LiPo 4 sel dengan kapasitas 1500mAh.
3. Proses interpretasi garis mampu bekerja dengan baik pada kondisi tidak optima1 dan 3. Pada kondisi tidak optimal 2, proses interpretasi garis tidak dapat bekerja secara optimal. Kondisi tidak idel 1, 2 dan 3 didefinisikan pada bab 4.3.
4. Pada tugas akhir ini, telah berhasil dibuat *indoor patrolling drone* yang dapat melakukan navigasi dalam ruangan dengan menggunakan sistem penjejakan garis. Penjejakan garis dapat dilakukan secara optimal pada kecepatan 48cm/s.

5.2. SARAN

Dalam pembuatan dan pengujian *indoor patrolling drone* menggunakan penjejakan garis berbasis visual ini, terdapat beberapa saran dan masukan yang perlu dipertimbangkan.

1. Drone sebaiknya diberi *casing* atau penutup agar lebih kuat.
2. Kamera yang digunakan sebaiknya dapat mengambil gambar dengan resolusi yang lebih kecil.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] H.-T. Lee, W.-C. Lin, dan C.-H. Huang, "Indoor Surveillance Security Robot with a Self-Propelled Patrolling Vehicle," *Journal of Robotics*, vol. 2011, hlm. 1–9, 2011.

[2] M. Morita, H. Kinjo, S. Sato, dan T. S. T. Anezaki, "Autonomous Flight Drone for Infrastructure (Transmission line) Inspection (3)," hlm. 4.

[3] C. J. Li dan H. Ling, "High-resolution, downward-looking radar imaging using a small consumer drone," dalam *2016 IEEE International Symposium on Antennas and Propagation (APSURSI)*, Fajardo, PR, USA, 2016, hlm. 2037–2038.

[4] D. Yallappa, M. Veerangouda, D. Maski, V. Palled, dan M. Bheemanna, "Development and evaluation of drone mounted sprayer for pesticide applications to crops," dalam *2017 IEEE Global Humanitarian Technology Conference (GHTC)*, San Jose, CA, 2017, hlm. 1–7.

[5] A. Bitar, A. Jamal, H. Sultan, N. Alkandari, dan M. El-Abd, "Medical Drones System for Amusement Parks," dalam *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, Hammamet, 2017, hlm. 19–20.

[6] M. Sajid, Y. J. Yang, G. B. Kim, dan K. H. Choi, "Remote monitoring of environment using multi-sensor wireless node installed on quad-copter drone," dalam *2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, Tokyo, Japan, 2016, hlm. 213–216.

[7] J.-H. Yang dan Y. Chang, "Feasibility study of RFID-Mounted drone application in management of oyster farms," dalam *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Fort Worth, TX, 2017, hlm. 3610–3613.

[8] L. Apvrille, Y. Roudier, dan T. J. Tanzi, "Autonomous drones for disasters management: Safety and security verifications," dalam *2015 1st URSI Atlantic Radio Science Conference (URSI AT-RASC)*, Gran Canaria, Spain, 2015, hlm. 1–2.

[9] C.-Y. Lin dan C.-Y. Liang, "Innovative Drone Selfie System and Implementation," dalam *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, Budapest, 2017, hlm. 135–140.

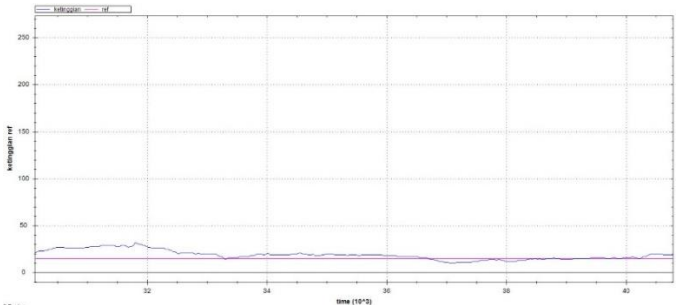
[10] A. R. Marhani dan Murwendah, “The Utilization of Drones for Smart Governance Implementation: Tax Extensification Strategy for Transfer of Ownership in Real Estate Sales at Cibinong Primary Tax Office,” dalam *2018 International Conference on ICT for Smart Society (ICISS)*, Semarang, 2018, hlm. 1–6.
[11] D. Murugan, A. Garg, T. Ahmed, dan D. Singh, “Fusion of drone and satellite data for precision agriculture monitoring,” dalam *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, Roorkee, 2016, hlm. 910–914.
[12] F. R. Ali dan A. T. Rashid, “Design and implementation of static and dynamic objects store systems using line follower robots,” dalam *2018 International Conference on Advance of Sustainable Engineering and its Application (ICASEA)*, Wasit, 2018, hlm. 37–42.
[13] M. Piras dan A. Cina, “Indoor positioning using low cost GPS receivers: Tests and statistical analyses,” dalam *2010 International Conference on Indoor Positioning and Indoor Navigation*, Zurich, Switzerland, 2010, hlm. 1–7.
[14] R. Austin, *Unmanned Aircraft Systems: UAVS Design, Development and Deployment*. Chichester, UK: John Wiley & Sons, Ltd, 2010.
[15] C. Noviello, *Mastering STM32*, 0.18. Leanpub, 2016.
[16] “STM32™ 32-bit MCU family brochure,” STMicroelectronics, Feb 2018.
[17] “STM32F103x8 STM32F103xB datasheet.” STMicroelectronics, 21-Agu-2015.
[18] “Raspberry Pi 3 Model B - Raspberry Pi.” [Daring]. Tersedia pada: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Diakses: 08-Mei-2019].
[19] “IMX219PQ | Sony Semiconductor Solutions.” [Daring]. Tersedia pada: https://www.sony-semicon.co.jp/products_en/new_pro/april_2014/imx219_e.html. [Diakses: 08-Mei-2019].
[20] “Camera Module V2,” *Raspberry Pi*. .
[21] “Omnibus F4 V6 Flight Controller – Unmanned Tech UK FPV Shop.” [Daring]. Tersedia pada: https://www.unmannedtechshop.co.uk/product/omnibus-f4-v6-flight-controller/. [Diakses: 08-Mei-2019].
[22] “INAV: Navigation-enabled flight control *software*.,” 26-Mei-2019. [Daring]. Tersedia pada: https://github.com/iNavFlight/inav.

[Diakses: 26-Mei-2019].
[23] D. Norris, *Build your own quadcopter: power up your designs with the Parallax Elev-8*. New York: McGraw-Hill Education, 2014.
[24] "FS-X6B Receiver User Manual." Flysky RC model technology, 2016.
[25] "Open Source Flight Controller Firmware. Contribute to betaflight/betaflight development by creating an account on GitHub," 26-Mei-2019. [Daring]. Tersedia pada: https://github.com/betaflight/betaflight. [Diakses: 26-Mei-2019].
[26] A. Williams, *C++ concurrency in action: practical multithreading*. Shelter Island, NY: Manning, 2012.
[27] G. B. Adrian Kaehler, *Learning OpenCV 3 Computer vision in C++ with the OpenCV library*. O'Reilly, 2017.
[28] J. R. Winkler, "Numerical recipes in C: The art of scientific computing, second edition," *Endeavour*, vol. 17, no. 4, hlm. 201, Jan 1993.
[29] "15-494/694 Cognitive Robotics Lab 7: OpenCV." [Daring]. Tersedia pada: https://www.cs.cmu.edu/afs/cs/academic/class/15494-s18/labs/lab7/lab7.html. [Diakses: 27-Mei-2019].
[30] Muh. A. Rahman, I. K. Edy Purnama, dan M. H. Purnomo, "Simple method of human skin detection using HSV and YCbCr color spaces," dalam *2014 International Conference on Intelligent Autonomous Agents, Networks and Systems*, BANDUNG, Indonesia, 2014, hlm. 58–61.
[31] B. Sensortec, "Digital, barometric pressure sensor," hlm. 2.
[32] W. Ding, J. Wang, dan A. Almagbile, "Adaptive Filter Design for UAV Navigation with GPS/INS/Optic Flow Integration," dalam *2010 International Conference on Electrical and Control Engineering*, Wuhan, China, 2010, hlm. 4623–4626.
[33] K. M. Hasan, Abdullah-Al-Nahid, dan A. Al Mamun, "Implementation of autonomous line follower robot," dalam *2012 International Conference on Informatics, Electronics & Vision (ICIEV)*, Dhaka, Bangladesh, 2012, hlm. 865–869.
[34] F. Kaiser, S. Islam, W. Imran, K. H. Khan, dan K. M. A. Islam, "Line follower robot: Fabrication and accuracy measurement by data acquisition," dalam *2014 International Conference on Electrical Engineering and Information & Communication Technology*, Dhaka, Bangladesh, 2014, hlm. 1–6.
[35] K. Khade, R. Naik, dan A. Patil, "Design of all color line follower sensor with auto calibration ability," dalam *2017 7th*

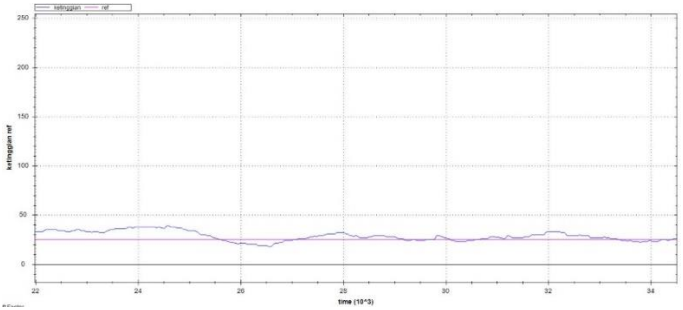
International Symposium on Embedded Computing and System Design (ISED), Durgapur, 2017, hlm. 1–5.
[36] A. Tariq, S. M. Osama, dan A. Gillani, "Development of a Low Cost and Light Weight UAV for Photogrammetry and Precision Land Mapping Using Aerial Imagery," dalam *2016 International Conference on Frontiers of Information Technology (FIT)*, Islamabad, Pakistan, 2016, hlm. 360–364.
[37] P. Ganesan, V. Rajini, B. S. Sathish, dan K. B. Shaik, "HSV color space based segmentation of region of interest in satellite images," dalam *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, Kanyakumari District, India, 2014, hlm. 101–105.

LAMPIRAN A

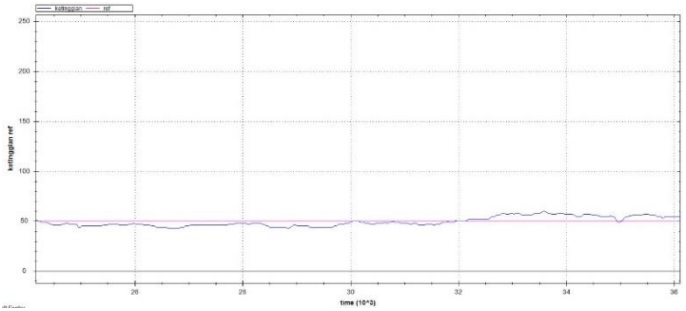
Grafik pengujian pengukuran ketinggian BMP280



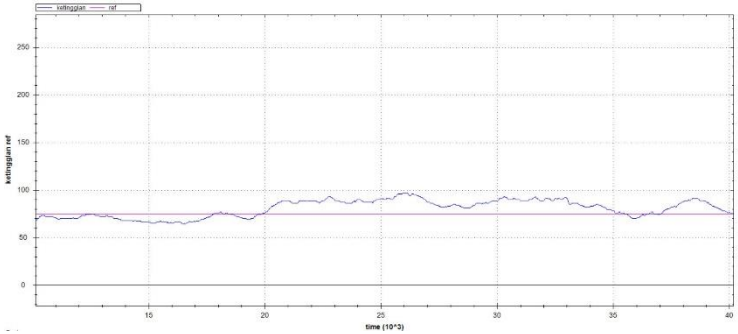
Grafik pengujian BMP280 pada ketinggian 15 cm



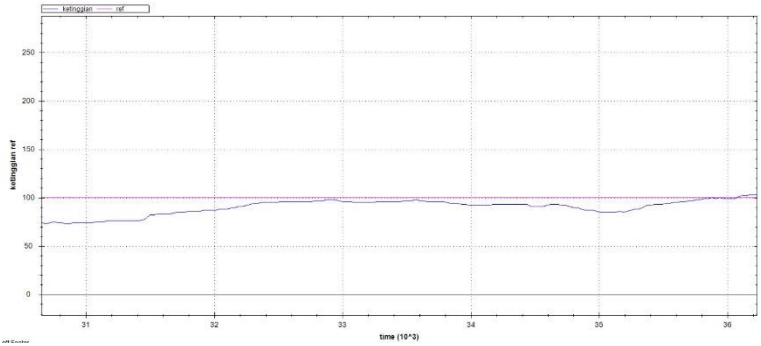
Grafik pengujian BMP280 pada ketinggian 25 cm



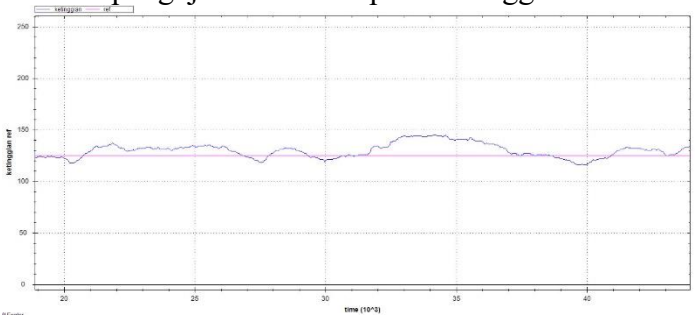
Grafik pengujian BMP280 pada ketinggian 50 cm



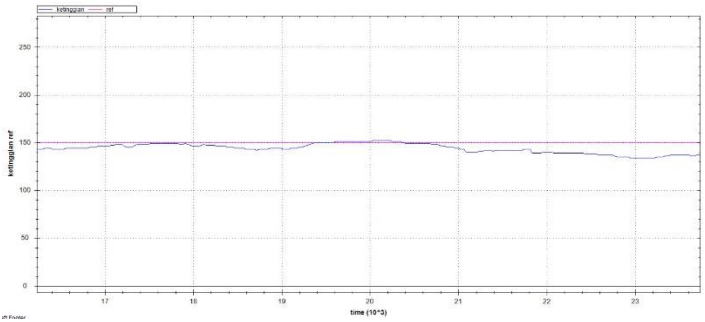
Grafik pengujian BMP280 pada ketinggian 75 cm



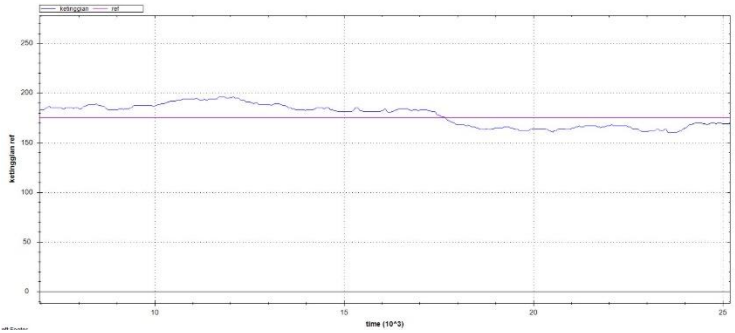
Grafik pengujian BMP280 pada ketinggian 100 cm



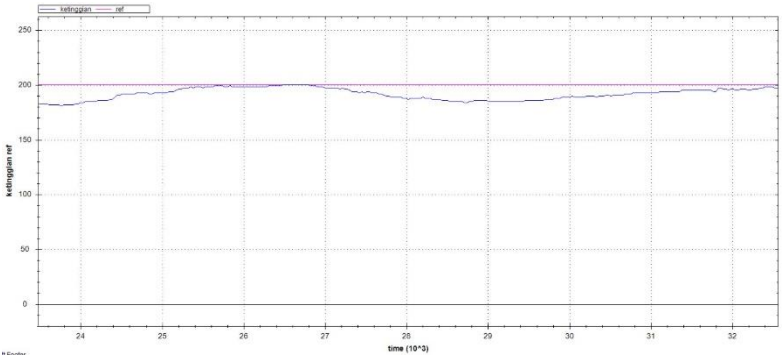
Grafik pengujian BMP280 pada ketinggian 125 cm



Grafik pengujian BMP280 pada ketinggian 150 cm



Grafik pengujian BMP280 pada ketinggian 175 cm



Grafik pengujian BMP280 pada ketinggian 200 cm

Halaman ini sengaja dikosongkan

LAMPIRAN B

Program miktrokontroler

UserCode.h

```
#ifndef USERCODE_H_
#define USERCODE_H_

#include "main.h"
#include "stm32f1xx_hal.h"
#include <math.h>

#define HEADER_1 0x20
#define HEADER_2 0x40
#define CONTROL_DELAY 1000

#define RX  0
#define RPI 1

//////////////////////////////////////////////////////
typedef struct {
   uint8_t flag;
   uint32_t prev;
}perio;

typedef struct {
   uint16_t data;
   uint8_t cnt;
   perio tim;
}buzzer;
//////////////////////////////////////////////////////
I2C_HandleTypeDef hi2c1;
UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;
UART_HandleTypeDef huart3;

DMA_HandleTypeDef hdma_usart1_tx;
DMA_HandleTypeDef hdma_usart2_rx;
DMA_HandleTypeDef hdma_usart2_tx;
DMA_HandleTypeDef hdma_usart3_rx;
```

```
WWDG_HandleTypeDef hwwdg;

int setup();
int loop();

int decode_Ibus(uint8_t *data_in, uint16_t* data_out);
int encode_Ibus(uint16_t *data_in, uint8_t* data_out);
int send_mu2fc(uint16_t *channel_data, uint8_t *buffer);
int send_mu2rpi(uint16_t *channel_data, uint8_t *buffer);

int periodic_call(perio *perio_struct, uint32_t period);
void refresh_periodic_call(perio *perio_struct);
int initiate_periodic_call(perio *perio_struct);

int delayed_call(perio *perio_struct, uint32_t _delay);
void refresh_delayed_call(perio *perio_struct);

int input_changed(uint16_t *_data, uint16_t *_last_data,uint16_t _thr);

void buzzer_send(buzzer* buzzer_data);
void buzzer_callback(buzzer* buzzer_data);
//////////////////////////////////////////////////////

uint8_t rx2mu_buffer [32];
uint16_t rx2mu_data [14];
uint16_t prev_rx2mu_data [14];
uint8_t mu2fc_buffer [32];
uint16_t mu2fc_data [14];
uint16_t last_armed;

uint8_t mu2rpi_buffer [32];
uint16_t mu2rpi_data [14];
uint8_t rpi2mu_buffer [32];
uint16_t rpi2mu_data [14];

uint16_t rx_OK, rx_missmatch, rx_empty, rx_inv;
int16_t rx_error;
float rx_error_perc;
```

```
uint16_t rpi_OK, rpi_missmatch, rpi_empty, rpi_inv;
int16_t rpi_error;
float rpi_error_perc;

perio rx2mu_request_call, mu2fc_send_call, rpi2mu_request_call,
mu2rpi_send_call;
perio target_rate;
perio mu2raspi_fail;
perio control_delay;

buzzer buzzer_on;
buzzer buzzer_rpi, buzzer_rx;

uint32_t target_count;
uint32_t WWDG_periode, WWDG_prev_time;

uint8_t rx_lost_flag;
uint8_t rpi_rx_flag;
uint8_t control_flag, last_control_flag;

uint32_t debug_tx;
//////////////////////////////////////////////////////////////////
#define FIL_SZ 250
I2C_HandleTypeDef hi2c1;
typedef struct {
   uint16_t dig_T1;
   int16_t dig_T2, dig_T3;
} temp_param;

typedef struct {
   uint16_t dig_P1;
   int16_t dig_P2, dig_P3, dig_P4, dig_P5, dig_P6, dig_P7, dig_P8,
dig_P9;
} pres_param;

void bmp_init(uint8_t dev_addr, I2C_HandleTypeDef hi2c1);
pres_param get_pressure_parameter(uint8_t dev_addr,
I2C_HandleTypeDef hi2c1);
```

```
temp_param get_temperature_parameter(uint8_t dev_addr,
I2C_HandleTypeDef hi2c1);

double get_temperature(uint8_t dev_addr, temp_param temp_param,
int32_t* t_fine, I2C_HandleTypeDef hi2c1);
double get_pressure(uint8_t dev_addr, pres_param pres_param, int32_t
t_fine, I2C_HandleTypeDef hi2c1);
double get_altitude(double pressure, double ref_alt);
double mov_avg(double in);

pres_param pressure_param;
temp_param temperature_param;
double temperature, pressure, altitude, ref_altitude, altitude_fil;
int32_t t_fine;
//////////////////////////////////////////////////////////////////
#endif /* USERCODE_H_ */
```

UserCode.c

```
#include "userCode.h"
int a = 1;
uint8_t dev_addr = 0xEC;

int setup() {
   initiate_periodic_call(&rx2mu_request_call);
   initiate_periodic_call(&mu2fc_send_call);
   initiate_periodic_call(&target_rate);
   initiate_periodic_call(&rpi2mu_request_call);
   initiate_periodic_call(&mu2rpi_send_call);
   initiate_periodic_call(&mu2raspi_fail);

   bmp_init(dev_addr, hi2c1);
   pressure_param = get_pressure_parameter(dev_addr, hi2c1);
   temperature_param = get_temperature_parameter(dev_addr, hi2c1);

   for (int i = 0; i < FIL_SZ*2; i++) {
      temperature = get_temperature(dev_addr, temperature_param,
&t_fine, hi2c1);
      pressure = get_pressure(dev_addr, pressure_param, t_fine, hi2c1);
      altitude = get_altitude(pressure, 0);
```

```
        altitude_fil = mov_avg(altitude);
        HAL_Delay(5);
    }
    ref_altitude = altitude_fil;

    buzzer_on.data = 0B1001100110011001;
    buzzer_rpi.data = 0B1010100000001111;
    buzzer_rx.data = 0B1111000000010101;

    buzzer_on.cnt = 17;
    buzzer_rpi.cnt = 17;
    buzzer_rx.cnt = 17;

    buzzer_send(&buzzer_on);

    refresh_delayed_call(&control_delay);

    rx_lost_flag = 1;
    rpi_rx_flag = 0;
    debug_tx = 0;
    control_flag = RX;
    last_control_flag = RX;
    last_armed = 0;
    return 0;
}

int loop() {
    if (rx2mu_data[4] == 1000 && last_armed == 2000)
        NVIC_SystemReset();

    last_armed = rx2mu_data[4];
    if (rx2mu_data[9] > 1800) {
        rx_OK = rx_missmatch = rx_empty = rx_inv = 0;
        rpi_OK = rpi_missmatch = rpi_empty = rpi_inv = 0;
        target_count = 0;
        debug_tx = 0;
    } else {
        refresh_delayed_call(&control_delay);
        control_flag = RX;
```

```
    }

    if (input_changed(rx2mu_data, prev_rx2mu_data, 5)) {
        refresh_delayed_call(&control_delay);
        control_flag = RX;
        for (int i = 0; i < 14; i++)
            prev_rx2mu_data[i] = rx2mu_data[i];
    } else if (delayed_call(&control_delay, CONTROL_DELAY) &&
rx2mu_data[9] > 1800) {
        control_flag = RPI;
    }

    if (periodic_call(&rx2mu_request_call, 3)) {
        HAL_UART_Receive_DMA(&huart3, rx2mu_buffer, 32);
        HAL_UART_Receive_DMA(&huart2, rpi2mu_buffer, 32);
        rpi_rx_flag = 1;
    }

    if (periodic_call(&mu2raspi_fail, 20)) {
        HAL_UART_Abort(&huart2);
        debug_tx++;
    }

    if (periodic_call(&mu2fc_send_call, 6)) {
        if (rx_lost_flag == 1) {
            if (control_flag == RPI) {
                send_mu2fc(rpi2mu_data, mu2fc_buffer);
            } else
                send_mu2fc(rx2mu_data, mu2fc_buffer);
        }

        temperature = get_temperature(dev_addr, temperature_param,
&t_fine,
                hi2c1);
        pressure = get_pressure(dev_addr, pressure_param, t_fine, hi2c1);
        altitude = get_altitude(pressure, ref_altitude);
        altitude_fil = mov_avg(altitude);
    }
```

```
    if (periodic_call(&target_rate, 70)) {
        //a+= 1;
        //HAL_GPIO_TogglePin(onBoardLED_GPIO_Port,
onBoardLED_Pin);
        target_count++;
    }

    rx_error = rx_inv + rx_missmatch;
    rx_error_perc = ((float) rx_error / (float) rx_OK) * 100;

    rpi_error = rpi_inv + rpi_missmatch;
    rpi_error_perc = ((float) rpi_error / (float) rpi_OK) * 100;

    if (last_control_flag == RX && control_flag == RPI) {
        buzzer_send(&buzzer_rpi);
        last_control_flag = control_flag;
    } else if (last_control_flag == RPI && control_flag == RX) {
        buzzer_send(&buzzer_rx);
        last_control_flag = control_flag;
    }

    buzzer_callback(&buzzer_on);
    buzzer_callback(&buzzer_rpi);
    buzzer_callback(&buzzer_rx);
    return 0;
}

int decode_Ibus(uint8_t *data_in, uint16_t* data_out) {
    uint16_t buffSum = 0xffff, cksum;

    //HAL_UART_Transmit_DMA(&huart1, data_in, 32);
    if (*data_in == HEADER_1 && *(data_in + 1) == HEADER_2) {
        cksum = *(data_in + 31) << 8;
        cksum |= *(data_in + 30);

        for (uint16_t i = 0; i < 30; i++)
            buffSum -= *(data_in + i);

        if (buffSum != cksum) {
```

```
            data_in[0] = 0xfd;
            data_in[1] = 0xff;
            return -1; // checksum missmatch
        }

        for (uint16_t i = 0; i < 14; i++) {
            *(data_out + i) = *(data_in + (2 * (i + 1)) + 1) << 8;
            *(data_out + i) |= *(data_in + (2 * (i + 1)));
        }

        data_in[0] = 0xfd;
        data_in[1] = 0xff;
        return 0; // data 0K
    }

    else if (*data_in == 0xfd && *(data_in + 1) == 0xff)
        return -2; // empty data

    else {
        data_in[0] = 0xfd;
        data_in[1] = 0xff;
        return -3; // invalid header
    }

}
int encode_Ibus(uint16_t *data_in, uint8_t* data_out) {
    data_out[0] = HEADER_1;
    data_out[1] = HEADER_2;

    for (uint16_t i = 0; i < 14; i++) {
        data_out[2 * (i + 1) + 1] = data_in[i] >> 8;
        data_out[2 * (i + 1)] = data_in[i] & 255;
    }

    uint16_t buffSum = 0xffff;
    for (uint16_t i = 0; i < 30; i++)
        buffSum -= *(data_out + i);

    data_out[31] = buffSum >> 8;
```

```
    data_out[30] = buffSum & 255;
    return 0;
}

int send_mu2fc(uint16_t *channel_data, uint8_t *buffer) {
    encode_Ibus(channel_data, buffer);
    HAL_UART_Transmit_DMA(&huart1, buffer, 32);
    return 0;
}
int send_mu2rpi(uint16_t *channel_data, uint8_t *buffer) {
    channel_data[10] = (int16_t)altitude_fil;
    encode_Ibus(channel_data, buffer);
    HAL_UART_Transmit_DMA(&huart2, buffer, 32);
    return 0;
}

int periodic_call(perio *perio_struct, uint32_t period) {
    if (perio_struct->flag == 2) { // pertama dipanggil
        perio_struct->prev = HAL_GetTick();
        perio_struct->flag = 0;
        return 1;
    }

    if (HAL_GetTick() - perio_struct->prev >= period) { // jika sesuai
periode flag = 1
        perio_struct->flag = 1;
        perio_struct->prev += period;
    } else
        perio_struct->flag = 0;

    if (perio_struct->flag == 1) { // setelah eksekusi flag = 0
        perio_struct->flag = 0;
        return 1;
    } else
        perio_struct->flag = 0;

    return 0;
}
int initiate_periodic_call(perio *perio_struct) {
```

```
    perio_struct->flag = 2;
    return 0;
}
void refresh_periodic_call(perio *perio_struct) {
    perio_struct->prev = HAL_GetTick();
}
int delayed_call(perio *perio_struct, uint32_t _delay) {
    if (HAL_GetTick() - perio_struct->prev >= _delay) {
        return 1;
    }
    return 0;
}
void refresh_delayed_call(perio *perio_struct) {
    perio_struct->prev = HAL_GetTick();
}

int input_changed(uint16_t *_data, uint16_t *_last_data, uint16_t _thr) {
    if (_data[0] - _last_data[0] >= _thr || _data[0] - _last_data[0] <= -_thr)
{
        return 1;
    }

    if (_data[1] - _last_data[1] >= _thr || _data[1] - _last_data[1] <= -_thr)
{
        return 1;
    }

    if (_data[3] - _last_data[3] >= _thr || _data[3] - _last_data[3] <= -_thr)
{
        return 1;
    }

    if (_data[2] - _last_data[2] >= _thr || _data[2] - _last_data[2] <= -_thr)
{
        return 0;
    }
    return 0;
}
```

```
void buzzer_send(buzzer* buzzer_data) {
   buzzer_data->cnt = 65535;
   buzzer_data->tim.flag = 2;
}
void buzzer_callback(buzzer* buzzer_data) {
   uint16_t temp_data = buzzer_data->data >> buzzer_data->cnt;

   if (buzzer_data->cnt < 17)
      HAL_GPIO_WritePin(onBoardLED_GPIO_Port,
onBoardLED_Pin, temp_data & 1);

   if (periodic_call(&buzzer_data->tim, 70)) {
      buzzer_data->cnt++;
      //HAL_GPIO_TogglePin(onBoardLED_GPIO_Port,
onBoardLED_Pin);
      if (buzzer_data->cnt > 17)
         buzzer_data->cnt = 17;
   }

}

//////////////////////////////////////////
void bmp_init(uint8_t dev_addr, I2C_HandleTypeDef hi2c) {
   uint8_t I2C_tx_buff[3];
   I2C_tx_buff[0] = 0xF4;
   I2C_tx_buff[1] = 0x57;
   I2C_tx_buff[2] = 0x1C;
//  I2C_tx_buff[2] = 0x00;
   HAL_I2C_Master_Transmit(&hi2c, dev_addr, I2C_tx_buff, 3, 20);
}

pres_param get_pressure_parameter(uint8_t dev_addr,
I2C_HandleTypeDef hi2c) {
   uint8_t I2C_rx_buff[18];
   uint8_t I2C_tx_buff[1];
   uint16_t temp;

   pres_param ret_val;
```

```
I2C_tx_buff[0] = 0x8E;
HAL_I2C_Master_Transmit(&hi2c, dev_addr, I2C_tx_buff, 1, 20);
HAL_I2C_Master_Receive(&hi2c, dev_addr, I2C_rx_buff, 18, 100);

ret_val.dig_P1 = I2C_rx_buff[1] << 8;
ret_val.dig_P1 |= I2C_rx_buff[0];

temp = I2C_rx_buff[3] << 8;
temp |= I2C_rx_buff[2];
ret_val.dig_P2 = (int16_t) temp;

temp = I2C_rx_buff[5] << 8;
temp |= I2C_rx_buff[4];
ret_val.dig_P3 = (int16_t) temp;

temp = I2C_rx_buff[7] << 8;
temp |= I2C_rx_buff[6];
ret_val.dig_P4 = (int16_t) temp;

temp = I2C_rx_buff[9] << 8;
temp |= I2C_rx_buff[8];
ret_val.dig_P5 = (int16_t) temp;

temp = I2C_rx_buff[11] << 8;
temp |= I2C_rx_buff[10];
ret_val.dig_P6 = (int16_t) temp;

temp = I2C_rx_buff[13] << 8;
temp |= I2C_rx_buff[12];
ret_val.dig_P7 = (int16_t) temp;

temp = I2C_rx_buff[15] << 8;
temp |= I2C_rx_buff[14];
ret_val.dig_P8 = (int16_t) temp;

temp = I2C_rx_buff[17] << 8;
temp |= I2C_rx_buff[16];
ret_val.dig_P9 = (int16_t) temp;
```

```
    return ret_val;
}

temp_param get_temperature_parameter(uint8_t dev_addr,
I2C_HandleTypeDef hi2c) {
    uint8_t I2C_rx_buff[6];
    uint8_t I2C_tx_buff[1];
    uint16_t temp;

    temp_param ret_val;

    I2C_tx_buff[0] = 0x88;
    HAL_I2C_Master_Transmit(&hi2c, dev_addr, I2C_tx_buff, 1, 20);
    HAL_I2C_Master_Receive(&hi2c, dev_addr, I2C_rx_buff, 6, 100);

    ret_val.dig_T1 = I2C_rx_buff[1] << 8;
    ret_val.dig_T1 |= I2C_rx_buff[0];

    temp = I2C_rx_buff[3] << 8;
    temp |= I2C_rx_buff[2];
    ret_val.dig_T2 = (int16_t) temp;

    temp = I2C_rx_buff[5] << 8;
    temp |= I2C_rx_buff[4];
    ret_val.dig_T3 = (int16_t) temp;

    return ret_val;
}

double get_temperature(uint8_t dev_addr, temp_param temp_param,
int32_t* t_fine,
        I2C_HandleTypeDef hi2c) {
    uint8_t I2C_rx_buff[3];
    uint8_t I2C_tx_buff[1];

    I2C_tx_buff[0] = 0xFA;
    HAL_I2C_Master_Transmit(&hi2c, dev_addr, I2C_tx_buff, 1, 20);
    HAL_I2C_Master_Receive(&hi2c, dev_addr, I2C_rx_buff, 3, 20);
```

```
    int32_t temp_ADC;

    temp_ADC = I2C_rx_buff[0];
    temp_ADC <<= 8;
    temp_ADC |= I2C_rx_buff[1];
    temp_ADC <<= 8;
    temp_ADC |= I2C_rx_buff[2];
    temp_ADC >>= 4;

    int32_t var1, var2;

    var1 = ((((temp_ADC >> 3) - ((int32_t) temp_param.dig_T1 << 1)))
            * ((int32_t) temp_param.dig_T2)) >> 11;

    var2 = (((((temp_ADC >> 4) - ((int32_t) temp_param.dig_T1))
            * ((temp_ADC >> 4) - ((int32_t) temp_param.dig_T1))) >> 12)
            * ((int32_t) temp_param.dig_T3)) >> 14;

    *t_fine = var1 + var2;

    double temperature;
    temperature = (*t_fine * 5 + 128) >> 8;
    temperature /= 100;

    return temperature;
}

double get_pressure(uint8_t dev_addr, pres_param pres_param, int32_t
t_fine,
        I2C_HandleTypeDef hi2c) {
    uint8_t I2C_rx_buff[3];
    uint8_t I2C_tx_buff[1];

    I2C_tx_buff[0] = 0xF7;
    HAL_I2C_Master_Transmit(&hi2c, dev_addr, I2C_tx_buff, 1, 20);
    HAL_I2C_Master_Receive(&hi2c, dev_addr, I2C_rx_buff, 3, 20);

    int32_t pres_ADC;
```

```
    pres_ADC = I2C_rx_buff[0];
    pres_ADC <<= 8;
    pres_ADC |= I2C_rx_buff[1];
    pres_ADC <<= 8;
    pres_ADC |= I2C_rx_buff[2];
    pres_ADC >>= 4;

    int64_t var11, var22, p;

    var11 = ((int64_t) t_fine) - 128000;
    var22 = var11 * var11 * (int64_t) pres_param.dig_P6;
    var22 = var22 + ((var11 * (int64_t) pres_param.dig_P5) << 17);
    var22 = var22 + (((int64_t) pres_param.dig_P4) << 35);
    var11 = ((var11 * var11 * (int64_t) pres_param.dig_P3) >> 8)
          + ((var11 * (int64_t) pres_param.dig_P2) << 12);
    var11 = (((((int64_t) 1) << 47) + var11)) * ((int64_t)
pres_param.dig_P1)
          >> 33;

    double pressure;
    if (var11 != 0) {
       // avoid exception caused by division by zero
       p = 1048576 - pres_ADC;
       p = (((p << 31) - var22) * 3125) / var11;
       var11 = (((int64_t) pres_param.dig_P9) * (p >> 13) * (p >> 13)) >>
25;
       var22 = (((int64_t) pres_param.dig_P8) * p) >> 19;

       p = ((p + var11 + var22) >> 8) + (((int64_t) pres_param.dig_P7) <<
4);
       pressure = (double) p / 256;
    }

    else
       pressure = -1.0;
    return pressure;
}

double get_altitude(double pressure, double ref_alt) {
```

```
    double altitude;
    altitude = 4433000 * (1.0 - pow((pressure / 100) / 1013.25, 0.1903));
    altitude -= ref_alt;
    return altitude;
}

double mov_avg(double in) {
    static double ar[FIL_SZ];
    static int i = 0;

    ar[i] = in;
    i++;
    if (i >= FIL_SZ)
        i = 0;

    double result = 0;
    for (int j = 0; j < FIL_SZ; j++)
        result += ar[j];

    result /= FIL_SZ;
    return result;
}

//////////////////////////////////////////
```

LAMPIRAN C

Program main *thread*

Main.h

```
#ifndef MAIN_H_
#define MAIN_H_
#include <fstream>
#include <iomanip>
#include <iostream>
#include <string>
#include "PID.h"
#include "Save_and_load_settings.h"
#include "cam1_thread.h"
#include "compiler_setting.h"
#include "log_file.h"
#include "periodic_call.h"
#include "stm32_comm.h"

#define POS_HOLD 0
#define SELF_ALIGN 1
#define CRUISE_FORWARD 2
#define CRUISE_BACKWARD 3

using namespace std;
using namespace bical;

thread1 STM32_comm_thread;
thread2 camera1_thread;
periodic_call timer1;
periodic_call PID_timer;

PID pitch;
PID roll_stab, roll_line;
PID yaw;

int16_t pitch_PID_sum = 0;
int16_t roll_PID_sum  = 0;
int16_t yaw_PID_sum   = 0;
int16_t ketinggian    = 0;
```

```
uint32_t test_var  = 0;
uint32_t loop_time = 0;

uint16_t RC_in_data[14];
uint16_t RC_out_data[14];
uint16_t ref_pitch, ref_roll, ref_yaw;
uint16_t main_state;
uint16_t end_of_line;

double cam1_fps;
double off_center, off_vertical;
double off_center_cm, off_vertical_cm;
double pitch_lim, roll_l_lim, roll_s_lim, yaw_lim;
double roll_s_filsz, pitch_filsz;
double cruise_fw_spd, cruise_bw_spd;
double off_center_th, off_vertical_th, end_of_line_max_th,
end_of_line_min_th;
double divider;
double pos_hold_vx, pos_hold_vy;

vector<double> roll_stab_kPID, roll_line_kPID;
vector<double> pitch_kPID;
vector<double> yaw_kPID;
vector<double> settings_dat;
vector<string> settings_title;

vel velocity;
vel velocity_cm;
pos position;

void read_thread_data();
void write_thread_data();
void load_setting_file();
void open_log_file();
void save_log_file();

uint8_t line_detected = 0;
uint8_t line_status[REGION_SPLIT];
```

```
double roll_setPoint  = 0;
double pitch_setPoint = 0;

#if LOG_PID == 1
double log_time;
periodic_call log_timer;
vector<double>     PID_roll_val,     PID_pitch_val,     PID_roll_fil_val,
PID_pitch_fil_val;
log_file PID_log;
char file_name[50];
#endif

#endif
```

Main.cpp

```
#include "main.h"

using namespace std;

int main() {

   STM32_comm_thread.startThread();

   camera1_thread.startThread();


   load_setting_file();


   roll_stab = PID(roll_stab_kPID, roll_s_lim, PID_PERIODE);

   roll_line = PID(roll_line_kPID, roll_l_lim, PID_PERIODE);

   pitch     = PID(pitch_kPID, pitch_lim, PID_PERIODE);

   yaw       = PID(yaw_kPID, yaw_lim, PID_PERIODE);


   roll_stab.set_in_fil_sz(roll_s_filsz);
```

```
    pitch.set_in_fil_sz(pitch_filsz);

    cout << showpos << fixed;
    main_state = 0;

#if LOG_PID == 1
    open_log_file();
#endif

    while (true) {
        read_thread_data();
        loop_time = timer1.elapsed();
        position.x += (velocity.vx) * 10;
        position.y += (velocity.vy) * 10;

        if (RC_in_data[8] == 1200) {
            roll_stab.reset_Iterm();
            roll_line.reset_Iterm();
            pitch.reset_Iterm();
            yaw.reset_Iterm();

            ref_roll  = RC_in_data[0];
```

```
    ref_pitch = RC_in_data[1];
    ref_yaw   = RC_in_data[3];
  }

  end_of_line = 0;
  for (uint8_t i = REGION_SPLIT; i > 0; i--) {
    if (line_status[i - 1] == 255)
      break;
    end_of_line++;
  }

  ////////////////////////////////////main_program/////////////////////////////////
  if (PID_timer.call(PID_PERIODE)) {
    if (!line_detected)
      //jika tidak mendeteksi garis maka pos_hold
      main_state = POS_HOLD;
    else {
      if (abs(off_center) > off_center_th || abs(off_vertical) >
off_vertical_th)
        // jika belum sejajar dg garis maka self_align
        main_state = SELF_ALIGN;
      else if (end_of_line < end_of_line_min_th)
        // jika sejajar maka forward_cruise
```

```
            main_state = CRUISE_FORWARD;
        else if (end_of_line >= end_of_line_min_th && end_of_line
<= end_of_line_max_th)
            // jika diujung garis maka self_align
            main_state = SELF_ALIGN;
        else if (end_of_line > end_of_line_max_th)
            // jika mendahului garis maka reverse_cruise
            main_state = CRUISE_BACKWARD;
        else
            //--//
            main_state = POS_HOLD;
    }
    //overwrite main state here
    //main_state = POS_HOLD;
    //overwrite main state here

    yaw_PID_sum = (int)yaw.compute(0, off_vertical);
    switch (main_state) {
      case POS_HOLD:
        roll_setPoint  = pos_hold_vx;
        pitch_setPoint = pos_hold_vy;
        yaw_PID_sum    = 0;
        break;
```

```
            case SELF_ALIGN:
                roll_setPoint  = -(int)roll_line.compute(0, off_center);
                pitch_setPoint = 0;
                break;
            case CRUISE_FORWARD:
                roll_setPoint  = -(int)roll_line.compute(0, off_center);
                pitch_setPoint = cruise_fw_spd;
                break;
            case CRUISE_BACKWARD:
                roll_setPoint  = -(int)roll_line.compute(0, off_center);
                pitch_setPoint = cruise_bw_spd;
                break;
        }
        pitch_PID_sum = (int)pitch.compute(pitch_setPoint,
velocity.vy);
        roll_PID_sum  = (int)roll_stab.compute(roll_setPoint,
velocity.vx);
    }
    ////////////////////////////////////main_program//////////////////////////////////


    RC_out_data[0] = ref_roll + roll_PID_sum;
    RC_out_data[1] = ref_pitch - pitch_PID_sum;
    RC_out_data[2] = RC_in_data[2];
```

```
        RC_out_data[3] = ref_yaw + yaw_PID_sum;
        RC_out_data[4] = RC_in_data[4];
        RC_out_data[5] = RC_in_data[5];
        RC_out_data[6] = RC_in_data[6];
        RC_out_data[7] = RC_in_data[7];
        RC_out_data[8] = RC_in_data[8];
        RC_out_data[9] = RC_in_data[9];


        if (timer1.call(20) == 1) {
            //cout << ketinggian << " " << off_center << " " << off_vertical
<< " " << velocity.vx << " " << velocity.vy << " " << roll_setPoint <<
endl;
            cout << ketinggian << endl;
        }


#if LOG_PID == 1
        if (log_timer.call(PID_PERIODE / LOG_RATIO)) {
            save_log_file();
        }
#endif


        write_thread_data();
    }
```

```
#if LOG_PID == 1
   PID_log.close_log();
#endif
   return 0;
}

void read_thread_data() {
   STM32_comm_thread.basicLock();
   for (int i = 0; i < 14; i++) {
      RC_in_data[i] = STM32_comm_thread.RC_in_data[i];
      ketinggian    = STM32_comm_thread.RC_in_data[10];
      if (ketinggian < 30) ketinggian = 30;
      //if (ketinggian > 300) ketinggian = 300;
   }
   divider = 278.5246148 * exp(-0.01911796479 * ketinggian);
   divider += 30;
   divider = 28.8 / divider;
   STM32_comm_thread.basicUnlock();

   camera1_thread.basicLock();
   cam1_fps      = camera1_thread.fps;
   velocity      = camera1_thread.velocity;
```

```
    velocity_cm.vx = velocity.vx * divider;
    velocity_cm.vy = velocity.vy * divider;
    velocity_cm.vx *= cam1_fps;
    velocity_cm.vy *= cam1_fps;
    off_center    = camera1_thread.off_center_out;
    off_center_cm = 0.5 * divider * off_center;
    off_vertical  = camera1_thread.off_vertical_out;
    line_detected = camera1_thread.line_detected_out;
    for (uint8_t i = 0; i < REGION_SPLIT; i++) {
        line_status[i] = camera1_thread.line_status_out[i];
    }
    camera1_thread.basicUnlock();
}
void write_thread_data() {
    STM32_comm_thread.basicLock();
    for (int i = 0; i < 14; i++) {
        STM32_comm_thread.RC_out_data[i] = RC_out_data[i];
    }
    STM32_comm_thread.basicUnlock();

    camera1_thread.basicLock();
    camera1_thread.basicUnlock();
```

```
}
void load_setting_file() {
    settings_dat   = vector<double>(26);
    settings_title = vector<string>(26);
    load_settings("settings.ini", &settings_title, &settings_dat);

    roll_stab_kPID = vector<double>(3);
    roll_line_kPID = vector<double>(3);
    pitch_kPID     = vector<double>(3);
    yaw_kPID       = vector<double>(3);

    cout <<
"_____SETTINGS.INI_____
_____" << endl
        << endl
        << endl;

    roll_stab_kPID[0] = settings_dat[5];
    roll_stab_kPID[1] = settings_dat[6];
    roll_stab_kPID[2] = settings_dat[7];
    cout << "roll_stab_kPID " << roll_stab_kPID[0] << " " <<
roll_stab_kPID[1] << " " << roll_stab_kPID[2] << endl;
```

```
    roll_line_kPID[0] = settings_dat[10];

    roll_line_kPID[1] = settings_dat[11];

    roll_line_kPID[2] = settings_dat[12];

    cout << "roll_line_kPID " << roll_line_kPID[0] << " " <<
roll_line_kPID[1] << " " << roll_line_kPID[2] << endl;


    pitch_kPID[0] = settings_dat[0];

    pitch_kPID[1] = settings_dat[1];

    pitch_kPID[2] = settings_dat[2];

    cout << "pitch_kPID " << pitch_kPID[0] << " " << pitch_kPID[1] <<
" " << pitch_kPID[2] << endl;


    yaw_kPID[0] = settings_dat[14];

    yaw_kPID[1] = settings_dat[15];

    yaw_kPID[2] = settings_dat[16];

    cout << "yaw_kPID " << yaw_kPID[0] << " " << yaw_kPID[1] << "
" << yaw_kPID[2] << endl;


    roll_l_lim = settings_dat[13];

    roll_s_lim = settings_dat[8];

    pitch_lim  = settings_dat[3];

    yaw_lim    = settings_dat[17];

    cout << "roll_l_lim " << roll_l_lim << endl;
```

```
cout << "roll_s_lim " << roll_s_lim << endl;
cout << "pitch_lim " << pitch_lim << endl;
cout << "yaw_lim " << yaw_lim << endl;

pitch_filsz  = settings_dat[4];
roll_s_filsz = settings_dat[9];
cout << "pitch_filsz " << pitch_filsz << endl;
cout << "roll_s_filsz " << roll_s_filsz << endl;

cruise_fw_spd = settings_dat[18];
cruise_bw_spd = settings_dat[19];
cout << "cruise_fw_spd " << cruise_fw_spd << endl;
cout << "cruise_fw_spd " << cruise_fw_spd << endl;

off_center_th   = settings_dat[20];
off_vertical_th = settings_dat[21];
cout << "off_center_th " << off_center_th << endl;
cout << "off_vertical_th " << off_vertical_th << endl;

end_of_line_max_th = settings_dat[22];
end_of_line_min_th = settings_dat[23];
cout << "end_of_line_max_th " << end_of_line_max_th << endl;
```

```
    cout << "end_of_line_min_th " << end_of_line_min_th << endl;


    pos_hold_vx = settings_dat[24];
    pos_hold_vy = settings_dat[25];
    cout << "pos_hold_vx " << pos_hold_vx << endl;
    cout << "pos_hold_vy " << pos_hold_vy << endl;
    cout
        <<
"_____SETTINGS.INI_____
_____" << endl
        << endl
        << endl;
  }
void open_log_file() {
    vector<string> log_title;
    ////////////////////////////////////////////////////
    log_title.push_back(string("time"));
    log_title.push_back(string("ketinggian"));
    log_title.push_back(string("vx"));
    log_title.push_back(string("vy"));
    log_title.push_back(string("off_center"));
    log_title.push_back(string("off_vertical"));
    log_title.push_back(string("roll_setpoint"));
```

```
    log_title.push_back(string("pitch_setpoint"));
    ////////////////////////////////////////////////////
    // snprintf(file_name, 50, "%.2f_%.2f_%.2f_center_vertical.csv",
pitch_kPID[0], pitch_kPID[1], pitch_kPID[2]);
    snprintf(file_name, 50, "PID sudut_%.2f.csv", cruise_bw_spd);
    PID_log.open_log(file_name, log_title);
}
void save_log_file() {
    vector<double> data;
    ////////////////////////////////////////////////////
    data.push_back(log_time);
    data.push_back(ketinggian);
    data.push_back(velocity_cm.vx);
    data.push_back(velocity_cm.vy);
    data.push_back(off_center_cm);
    data.push_back(off_vertical);
    data.push_back(roll_setPoint);
    data.push_back(pitch_setPoint);
    ////////////////////////////////////////////////////
    PID_log.log(data);
    log_time += PID_PERIODE / LOG_RATIO;
}
```

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis bernama Muhammad Abizhar Multazam yang sering dipanggil Abizhar. Penulis menempuh pendidikan sekolah dasar di SD Hangtuah X Juanda, kemudian dilanjutkan dengan pendidikan menengah di SMP Al-Falah Deltasari selama tiga tahun dan di SMAN 1 Sidoarjo selama 3 tahun. Penulis kemudian melanjutkan pendidikan perguruan tinggi di Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2015. Selama masa perkuliahan, Penulis menjadi salah satu asisten pada praktikum elektronika dasar dan juga menjadi salah satu asisten laboratorium B202. Pada tahun 2018, penulis juga sempat melaksanakan kerja praktek di Lembaga Elektronika Nasional PT LEN Industri Persero.

E-mail : AbizharMultazam@gmail.com
Whatsapp : 08813151871
Line : abizhar_multazam