TUGAS AKHIR – EE 184801

IMPLEMENTASI LIDAR SEBAGAI KONTROL KETINGGIAN QUADCOPTER

Sayyidul Aulia Alamsyah
NRP 07111540000094

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.

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

TUGAS AKHIR – EE 184801

IMPLEMENTASI LIDAR SEBAGAI KONTROL KETINGGIAN QUADCOPTER

Sayyidul Aulia Alamsyah
NRP 07111540000094

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.

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

FINAL PROJECT – EE 184801

LIDAR IMPLEMENTATION AS ALTITUDE CONTROL FOR QUADCOPTER

Sayyidul Aulia Alamsyah
NRP 07111540000094

Supervisor
Dr. Muhammad Rivai, ST., MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Implementasi Lidar Sebagai Kontrol Ketinggian Quadcopter**" 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, 9 Juli 2019

Sayyidul Aulia Alamsyah

NRP. 0711 15 4000 0094

(Halaman sengaja dikosongkan)

IMPLEMENTASI LIDAR SEBAGAI KONTROL KETINGGIAN QUADCOPTER

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

Dr. Muhammad Rivai, ST. MT.
NIP. 196904261994031003

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

SURABAYA
JULI, 2019

(Halaman sengaja dikosongkan)

IMPLEMENTASI LIDAR SEBAGAI KONTROL KETINGGIAN QUADCOPTER

Nama : Sayyidul Aulia Alamsyah
Pembimbing : Dr. Muhammad Rivai, ST., MT.

ABSTRAK

Quadcopter atau yang saat ini lebih di kenal dengan nama drone sudah menjadi hal yang dapat dimiliki dengan mudah. Telah banyak perusahaan perusahaan yang memproduksi *flight controller* yang sudah terintegrasi dengan banyak sensor di dalamnya, termasuk juga sensor ketinggian. Berbagai macam perusahaan tersebut sebagian besar menggunakan barometer yang sudah terintegrasi didalam flight controller sebagai sensor ketinggian. Barometer sendiri adalah sensor yang mendeteksi ketinggian berdasarkan nilai tekanan udara di sekitarnya, tekanan udara inilah yang digunakan barometer untuk menentukan ketinggian drone. Karena barometer mencari nilai ketinggian terhadap tekanan udara, maka pengaruh cuaca dan kecepatan angin akan memberikan efek pada nilai ketinggian barometer tersebut. Oleh karena itu Pada tugas akhir ini akan dirancang *quadcopter* dengan lidar sebagai sensor ketinggian dari permukaan tanah. Pada tugas akhir ini akan dilakukan percobaan untuk mengecek ketinggian *quadcopter* dengan menggunakan lidar dengan cara pengecekan hasil lidar dengan data sebenarnya. Sebelum mencoba mengontrol *quadcopter* menggunakan lidar, *quadcopter* harus di pastikan sudah berorientasi dengan benar dengan cara mengkalibarsi sensor dan memastikan waktu terbang terlama hingga baterai habis. Metode yang digunakan adalah dengan cara mengontrol sinyal pwm *input* pada *flight controller.* Sinyal pwm yang akan dikirimkan pada *flight controller* adalah sinyal hasil dari sistem *Proportional-Integral-Derivative* yang di olah pada mikrokontroler dengan *input* sistem berupa hasil pembacaan lidar. Mikrokontroler yang di gunakan adalah STM32F103C8T6. Hasil dari tugas akhir ini didapatkan lidar memiliki error rata-rata sebesar 3.97875% dengan parameter kontrol PID yang cocok di gunakan pada sistem pengatur ketinggian *quadcopter* adalah *Proportional-Derivative.* Respon terbaik yang dihasilkan pada tugas akhir ini adalah respon dengan waktu tercepat menuju keadaan *steady state* selama 22.55 detik.

Kata kunci : Kontrol Ketinggian, Lidar, Quadcopter

(Halaman sengaja dikosongkan)

LIDAR IMPLEMENTATION AS ALTITUDE CONTROL FOR QUADCOPTER

Name : Sayyidul Aulia Alamsyah
Supervisor : Dr. Muhammad Rivai, ST., MT.

ABSTRACT

Quadcopter or currently known as drone has become something that can be easily owned. Many companies have produced flight controllers that have been integrated with many sensors inside, including altitude sensors. Most types of these companies use barometer that have been integrated inside the flight controllers as altitude sensors. The barometer itself is a sensor that detects the altitude based on value of surrounding air pressure, this air pressure that used by barometer to determaine the altitude of drone. Because of the how barometer looks for the altitude value, the influence of weather and wind will have an effect on the barometer's altitude value. Therefore in this final project a quadcopter with lidar will be designed as altitude sensor frome the ground level. In this final project an experiment will be held to check the altitude of the quadcopter by using lidar and compare the result with the actual data. Before trying to control the quadcopter using lidar, quadcotper must be sure it's oriented correctly bye calibrating the sensor and ensuring the longest flight time until the battery runs out. The method that used is by controlling the pwm input signal on the flight controllers. The PWM signal that will be sent to the flight controller is a signal that result from Proportional-Integral-Derivative system which is processed on the microcontroller with system input in the form of lidar data reading. The Microcontroller that used in this final project is STM32F103C8T6. The result of this final project found that lidar has an average error of 3.97875% with PID control parameters that are suitable for use in quadcopter altitude control systems are proportional-derivative. The best response produced in this final project is the response with the fastest time to the steady state area for 22.55 seconds.

Keywords : Altitude control, Lidar, Quadcopter

(Halaman sengaja dikosongkan)

KATA PENGANTAR

Puji dan syukur kepada Allah SWT atas berkat rahmat yang diberikannya penulis dapat menyelesaikan laporan tugas akhir dengan judul "**Implementasi Lidar Sebagai Kontrol Ketinggian QuadCopter**", sebagai salah satu persyaratan dalam menyelesaikan pendidikan program Strata-Satu di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.

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

1. Bapak Dr. Muhammad Rivai S.T.,M.T. selaku dosen pembimbing atas gagasan topik tugas akhir serta bimbingan dan arahan untuk penulis selama mengerjakan tugas akhir.
2. Bapak Dr. Astria Nur Irfansyah, S.T.,M.Eng., Ir. Haris Pirngadi M.T., Fajar Budiman, S.T., M.Sc., dan Ir. Tasripan M.T., selaku penguji yang memberikan berbagai macam masukan untuk tugas akhir ini.
3. Bapak Dr. Eng. Ardyono Priyadi, ST. M.Eng., selaku Kepala Departemen Teknik Elektro yang telah menyediakan berbagai macam fasilitas dan kenyamanan selama menjalankan tugas akhir.
4. Orang tua, yang tak henti-hentinya memberikan semangat dan kasih sayang yang luar biasa kepada penulis.
5. Seluruh dosen bidang studi elektronika Departemen Teknik Elektro ITS.
6. Teman-teman bidang studi elektronika yang tidak dapat disebutkan satu-persatu yang telah membantu dan memberikan semangat kepada penulis selama menjalani perkuliah di Departemen Teknik Elektro ITS.

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

Surabaya, 9 Juli 2019

Sayyidul Aulia Aamsyah

(Halaman sengaja dikosongkan)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Quadcopter atau yang saat ini lebih di kenal dengan nama drone sudah menjadi hal yang dapat dimiliki dengan mudah. Telah banyak perusahaan perusahaan yang memproduksi *flight controller* yang sudah terintegrasi dengan banyak sensor di dalamnya, termasuk juga sensor ketinggian. Sensor ketinggian adalah sensor yang dapat menghitung ketinggian objek. Saat ini ketelitian ketinggian dari drone sangat diperhatikan. Hal ini dikarenakan drone mulai banyak digunakan dalam berbagai bidang. Pada bidang olahraga, drone dapat digunakan untuk menangkap video jalannya pertandingan dari sudut yang tidak bisa di lakukan secara konvensional. Drone juga dapat di pakai pada monitoring lalu lintas di jalan-jalan besar dengan cara mengumpulkan data penting terkait situasi keramaian jalan untuk kemudian digunakan untuk efisiensi arus lalu lintas. Pada evakuasi bencana, drone juga dapat digunakan sebagai penyedia status awal bencana sehingga dapat menurunkan resiko bagi manusia dan juga meningkatkan efisiensi dari tim evakuasi [1]. Selain pada pemanfaatan di berbagai bidang tersebut kestabilan ketinggian *quadcopter* juga banyak digunakan sebagai penelitian penggabungan kamera dan *quadcoper* [2]. Hampir keseluruhan dari permasalahan pada tiap bidang tersebut membutuhkan ketinggian drone yang stabil.

Pada umumnya perusahaan *flight controller* menggunakan barometer sebagai sensor ketinggian dari *quadcopter*. Barometer sendiri adalah sensor yang dapat menghitung nilai tekanan udara di sekitarnya sebagai acuan nilai ketinggian objek. Tekanan udara sendiri akan selalu terpengaruh oleh cuaca sehingga pembacaan nilai ketinggian objek juga akan terpengaruh oleh cuaca meskipun berada pada ketinggian yang sama [3]. Meskipun pada *flight control* sudah di beri filter pada pembacaan nilai barometer, namun jika terdapat perubahan yang sangat tinggi secara konstan dapat mempengaruhi hasil filter. Sehingga dibutuhkan sensor lain yang dapat menghitung nilai ketinggian sebuah *quadcopter* terhadap permukaan secara *real-time*.

1.2 Rumusan Masalah

1. Bagaimana cara menjaga ketinggian *quadcopter* dengan akurat.
2. Metode kontrol yang digunakan untuk menstabilkan ketinggian quadcopter.

1.3 Tujuan

1. Penerapan lidar sebagai sensor ketinggian pada *quadcopter*.
2. Penggunaan PID pada sistem kontrol ketinggian *quadcopter*.

1.4 Batasan Masalah

1. Jarak penggunaan lidar 0.3m -12m.
2. Pengontrolan *quadcopter* hanya pada ketinggiannya.
3. Pengujian hanya pada satu jenis baterai.

1.5 Metodologi

1. Studi Literatur

Studi literatur bertujuan untuk mempelajari teori-teori dasar dan penelitian-penelitian sebelumnya yang sudah di lakukan yang relevan dengan tugas akhir yang akan dilakukan. Literatur yang di pakai bersumber dari jurnal-jurnal, buku, artikel dan paper yang terpercaya. Pada studi literatur ini teori dasar dan penelitian yang akan di pelajari mengenai quadcopter, kontrol gerakan quadcopter, dan penggunaan lidar.

2. Observasi dan Analisa Masalah

Pada tahap ini akan dilakukan pengkajian tentang cara gerak *quadcopter*. Bagaimana hubungan kecepatan sudut motor dengan pergerakan naik *quadcopter*. Observasi dan analisa masalah dilakukan dengan mengkaji paper-paper, jurnal dan buku.

3. Persiapan Alat dan Bahan

Tahap ini merupakan tahap pencarian informasi tentang alat dan bahan untuk pembuatan *quadcopter* dan sistem pengatur ketinggian. Alat dan bahan yang dibutuhkan Pada mekanik berupa *frame quadcopter* yang akan digunakan. Pada sisi elektronik PDB, motor, propeler , ESC, dan *Flight Controller* yang akan digunakan pada *quadcopter*.

4. Perancangan dan Pembuatan Alat

Perancangan ini adalah perancangan secara keseluruhan *quadcopter* yang akan di gunakan. Perancangan ini meliputi mekanik, elektronik, dan program. Pada perancangan elektronik ini juga menentukan peletakan baterai dan jalur kabel untuk mengurangi resiko *short-circuit* dan juga mengamankan kabel dari propeler yang berputar nantinya. Pada perancangan program disini berupa perancangan pada mission planner nantinya yang akan di *upload* pada *Flight Controller*.

5. Tahap Pengujian

Pengujian akan dilakukan secara bertahap dimulai dari orientasi *quadcopter*, pengujian sensor yang digunakan yaitu lidar, pengujian sistem pengatur ketinggian. Pada awalnya akan dilakukan secara terpisah, jika sudah di pastikan benar maka akan di satukan.

6. Analisa dan Evaluasi

Analisa akan dilakukan pada setiap respon ketinggian dan output PWM yang di dapatkan pada setiap perubahan parameter PID. Analisa ini bertujuan unutk menemukan kekurangan dari respon yang di dapakan dan meningkatkan sistem.

7. Penulisan Laporan Tugas Akhir

Penulisan laporan tugas akhir merupakan tahap pembuatan laporan hasil pelaksanaan tugas akhir. Laporan tugas akhir ini berisi dasar teori, perancangan alat, pembuatan alat, pengujian alat, hasil pengujian sistem.

1.6 Sistematika Penulisan

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

- **BAB I: Pendahuluan**
  Bab ini menjelaskan tentang latar belakang, rumusan masalah, tujuan masalah, metodologi, sistematika penulisan, dan relevansi.
- **BAB II: Tinjauan Pustaka**
  Bab ini berisi tentang teori yang mendasari penyusunan laporan tugas akhir secara umum, khususnya yang berhubungan dengan komponen yang akan digunakan.
- **BAB III: Perancangan Sistem**
  Bab ini berisi tentang penjelasan rancangan dari sistem yang akan dibuat, meliputi perangkat keras (*hardware*) dan perangkat lunak (*software*).
- **BAB IV: Pengujian dan Analisis**
  Bab ini menjelaskan tentang pengujian alat yang telah dibuat beserta analisanya.
- **BAB V: Penutup**
  Bab ini berisi tentang kesimpulan yang diperoleh dari pembuatan alat serta saran untuk pengembangan lebih lanjut.

1.7 Relevansi

Dengan dibuatnya tugas akhir ini akan membantu pengembang *quadcopter* khusunya pada bidang pengolahan citra berbasis *quadcopter* diaman sangat membutuhkan kestabilan ketinggian dari permukaan tanah.

BAB II
TINJAUAN PUSTAKA

2.1 Light Detection and Ranging

Light detection and ranging atau yang dikenal dengan nama Lidar adalah suatu teknologi yang memanfaatkan sinar laser untuk dapat menghitung jarak suatu objek. Teknologi lidar ini adalah metode yang paling efektif untuk digunakan sebagai pengambilan data jarak suatu objek [4]. Untuk lidar jenis TF mini Lidar (Gambar 2.1) data hasil pembacaan jarak dari lidar ini dikirimkan oleh lidar dengan cara komunikasi serial pada mikrokontroler.

Gambar 2.1 TF Mini Lidar [5]

Pada penerapannya, lidar sudah banyak digunakan dan di teliti oleh berbagai penelitian. Contoh pertama yaitu penggunaan lidar pada sebuah truk [6]. Pada penelitian tersebut lidar yang digunakan ialah lidar 360o. Lidar jenis ini memiliki tiga nilai output, yaitu jarak, sudut, dan intensitas. Ketiga nilai tersebut lah yang digunakan truk untuk memprediksi posisi dirinya dan dapat berjalan dengan otomatis. Contoh kedua yaitu mendeteksi rintangan menggunakan lidar [7]. Pada penelitian tersebut lidar digunakan untuk mendeteksi dan mengklasifikasikan rintangan. Contoh ketiga yaitu penggunaan Lidar untuk memetakan ruangan [8]. Pada penelitian tersebut digunakan 2D lidar untuk memetakan ruangan secara 3 dimensi.

2.1.1 Prinsip Pengukuran Jarak

Pada prinsip kerjanya, Lidar TF Mini menghitung jarak benda berdasarkan ToF (Time of Flight). Lidar akan mentransmisikan gelombang laser (Light Amplication by stimulated emission of radiation). Laser adalah instrumen yang dapat menghasilkan energi radiasi yang kuat berupa emisi cahaya dengan cara mengalirkan arus kuat pada material penghasil cahaya seperti carbon dioxide, helium-neon, argon, rubies, ataupun material lain. Gelombang laser yang di hasilkan akan termodulasi

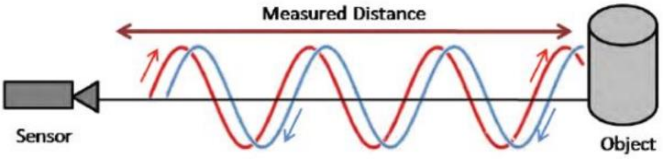
pada suatu periode waktu tertentu, dimana gelombang tersebut akan merefleksikan cahaya setelah mengenai suatu benda (Gambar 2.2) . Lidar akan menghitung waktu dengan cara menghitung perbedaan sudut kedatangan cahaya dan kemudian menghitung jarak relatif antara lidar dan benda.

$$T_L = nT + \frac{\varphi}{2\pi}T \quad (2.1)$$

Persamaan (2.1) adalah persamaan untuk menghitung time of travel (TL) berdasarkan beda fasa yang di dapat antara gelombang transmisi dan gelombang pantul. Dimana n adalah jumlah gelombang penuh, T adalah waktu tempuh cahaya untuk menempuh satu panjang gelombang dan φ adalah beda fasa.

$$R = \frac{T_L}{2}c \quad (2.2)$$

Setelah mengetahui time of travel (TL) maka selanjutnya dapat mencari Range (jarak). Persamaan (2.2) adalah persamaan untuk mencari jarak berdasarkan nilai TL yang sudah di ketahui. Dimana c adalah kecepatan cahaya pada medium antara lidar dan benda yang akan di ukur jaraknya [9].



Gambar 2.2 Cara kerja Lidar [9]

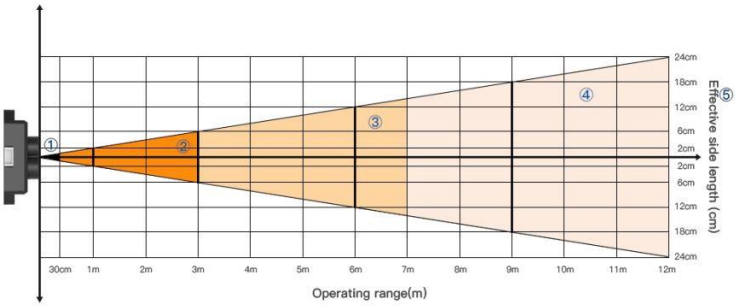
2.1.2 Karakteristik Parameter

Lidar memiliki parameter-parameter yang dapat di jadikan acuan dalam penggunaannya. Parameter-parameter ini berisi nilai-nilai seperti akurasi jarak yang di ukur, penggunaan optimal, dan sebagainya. Nilai-nilai ini biasanya diikutsertakan dengan buku panduan produk saat pembelian produk tersebut. Pada TF Mini Lidar berikut karakteristik parameter penggunaannya.

Tabel 2.1 Tabel Karakteristik Parameter

Deskripsi	**Nilai Parameter**
Operating Range (Indoor)	0.3m – 12m
Measurement accuracy	±4cm@ (0.3m – 6m)
	±6cm@(6m – 12m)
Default unit of distance	Cm
Range Resolution	5mm
Receiving half angle	1.15º
Transmitting half angle	1.5º
Frequency	100Hz

Selain karakteristik parameter pada tabel 2-1 terdapat karakteristik lain pada TF Mini Lidar.



Gambar 2.3 Karakteristik Jarak Pengukuran[10]

Karakteristik selanjutnya ialah karakteristik pada pengukuran jarak tertentu. Dapat dilihat pada gambar 2.3 bahwa semakin jauh jarak benda yang akan di ukur jaraknya, sinyal yang di pancarkan semakin melebar. Hal ini menyebabkan di setiap jarak pengukuran terdapat minimum lebar benda yang akan di ukur, nilai minimum ini dapat di lihat pada tabel 2-2. Apabila lebar benda tidak mencapai nilai minimum tersebut maka nilai pembacaan TF Mini Lidar akan bernilai diantara dua jarak yang berbeda.

Tabel 2.2 Tabel Minimum Lebar Benda

Jarak (m)	1	2	3	4	5	6	7	8	9	10	11	12
Lebar Minimum (cm)	4	8	12	16	20	24	28	32	36	40	44	48

2.1.3 Protokol Komunikasi dan Format Data

Seperti di jelaskan di awal bahwa komunikasi yang digunakan oleh lidar jenis TF Mini ialah menggunakan komunikasi serial. Komunikasi serial terdiri dari dua jalur data yaitu *transceiver* (Tx) dan *receiver* (Rx). Protokol komunikasi serial yang digunakan ialah UART dengan baudrate 115200 seperti terlihat pada tabel 2.3.

Tabel 2.3 Tabel Protokol Komunikasi Tf Mini Lidar

Comunication Interface	UART
Default Baudrate	115200
Data bit	8
Stopbit	1
Parity Check	None

Tabel 2.4 Tabel Format data setiap paket

Byte 0	0x59, “Y”
Byte 1	0x59, “Y”
Byte 2	Data jarak
Byte 3	Data jarak
Byte 4	Data kekuatan sinyal
Byte 5	Data kekuatan sinyal
Byte 6	Mode pengukuran jarak
Byte 7	Spare Byte, Default bernilai 0
Byte 8	Checksum

Format dari data yang dikirimkan oleh Tf Mini lidar adalah data dengan panjang 9 Byte untuk setiap paket data (Tabel 2.4) . 9 Byte data ini berisi nilai-nilai dari data jarak, kekuatan sinyal, mode pengukuran jarak yang di gunakan lidar, checksum, dan lain-lain. Bentuk data dari setiap nilai tersebut adalah hexadecimal (HEX). Checksum pada satu

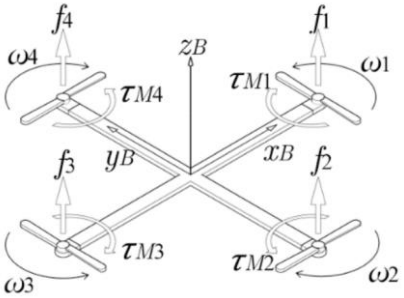
paket data berfungsi sebagai pengecek kebenaran dari setiap isi data dengan cara menjumlahkan semua data.

2.2 Unmanned Aerial Vehicle

Unmanned Aerial Vehicle (UAV) adalah pesawat terbang yang tidak membutuhkan manusia untuk mengendarainya. UAV sudah banyak di gunakan untuk berbagai penelitian sebelumnya. Contoh penelitian pertama ialah penelitian pada kontrol pergerakan UAV berjenis quadcopter [11]. Penelitian tersebut berfokus pada pengkontrolan pergerakan dan ketinggian dari quadcopter. Pada penelitian tersebut digunakan dua jenis pengkontrolan, yang pertama yaitu PID dan yang kedua yaitu kontrol prediksi model non-linear. Contoh penelitian kedua yaitu penelitian tentang pengintegrasian sensor untuk masalah pengisian bahan bakar dari UAV [12]. Pada penelitian tersebut digunakan Machine Vision dan GPS yang kemudian di gabungkan menggunakan Extended Kalman Filter.

Banyak sekali tipe dari UAV, salah satunya ialah Quadcopter. Quadcopter adalah UAV yang menggunakan empat motor untuk terbang dan bermanufer. Dua dari motor tersebut bergerak clockwise dan dua yang lain bergerak counterclockwise [13]. Banyak sekali penelitian untuk mendapatkan kontrol gerakan yang lebih baik pada quadcopter, salah satunya pada [14]. Pada penelitian tersebut di lakukan penelitian pada stabilitas quadcopter model V-tail dengan membandingkan kontrol PD, PID, dan *sliding-mode position*.

Rangka quadcopter terbuat dari 4 batang panjang, dimana di ujung-ujung batang tersebut di letakkan motor untuk menghasilkan thrusts tegak lurus dengan bidang rangka.



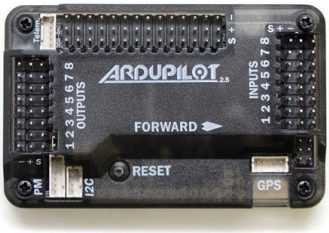
Gambar 2.4 Sistem pergerakan motor quadcopter[13]

Motor 1 dan 3 berputar secara counterclockwise, sedangkan motor 2 dan 4 berputar secara clockwise. Konfigurasi arah putaran dari keempat motor ini menjadi dasar dari tiga rotasi dari quadcopter yaitu yaw pada sumber Z, pitch pada sumbu X, dan roll pada sumbu Y. Pitch adalah gerakan menukik ke depan ataupun belakang. Yaw adalah gerakan menoleh ke kanan ataupun kiri. Roll adalah gerakan menukik ke kanan ataupun kiri. Pada pergerakan pitch positif kecepatan sudut motor 1 dan 2 akan meningkat dan motor 3 dan 4 akan menurun. Pada pergerakan roll positif kecepatan sudut motor 1 dan 4 akan meningkat sedangkan motor 2 dan 3 akan menurun. Pada yaw positif kecepatan sudut yang dinaikkan adalah motor 1 dan 3 sedangkan kecepatan sudut motor 2 dan 4 akan di turunkan. Gerakan naik turun pada quadcopter adalah hasil dari penambahan ataupun pengurangan kecepatan sudut secara bersamaan pada keempat motor [13].

2.3 Flight Controller

Flight Controller adalah mikrokontroller yang terintegrasi dengan berbagai macam sensor dan sudah diprogram sebagai pengontrol berbagai jenis UAV. Contoh-contoh flight controller ialah beta flight, ardupilot, PixHawk, APM, dan lain-lain.

Pada tugas akhir ini akan digunakan Flight controller jenis APM (Gambar 2.5) . APM adalah singkatan dari ArduPilot Mega, dimana APM adalah autopilot IMU berkualitas profesional yang berbasis pada platform arduino yaitu Arduino Mega. Jenis Flight Controller ini dapat mengkontrol fixed-wing aircraft dan helikopter multi motor. Pada APM ini sudah terintegrasi berbagai macam sensor seperti Gyroscope, Accelerometer, Magenetometer, dan Barometer [15].

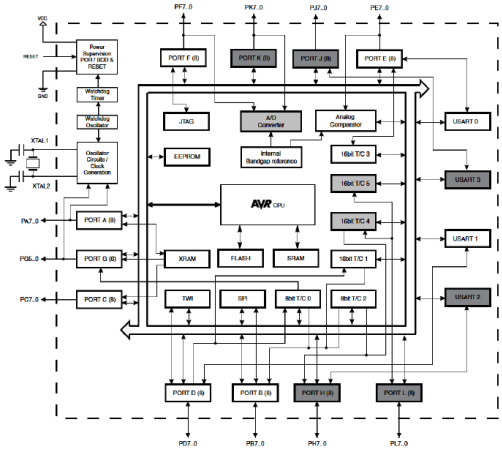


Gambar 2.5 ArduPilot Mega[16]

Seperti yang sudah di sebutkan sebelumnya bahwa APM berbasis pada mikrokontroller Arduino Mega, yaitu seri Atmega2560. Karena

APM berbasis Arduino maka APM dapat diprogram menggunakan aplikasi bawaan Arduino yaitu Arduino IDE. Arduino IDE adalah aplikasi buatan arduino yang bersifat Open Source dan juga mudah untuk digunakan. Fitur-fitur yang terdapat pada APM ialah sebagai berikut:

1. MPU-6000, merupakan sensor Gyroscope dan Accelerometer.
2. HMC5883L-TR, merupakan sensor Magnetometer.
3. IC data flash sebesar 4MB, untuk data logging otomatis.
4. Pin input PWM sebanyak 8, 4 diantaranya untuk input sinyal Throttle, yaw, pitch, dan roll dari receiver remote controller.
5. MS5611-01BA03, Sensor Barometer
6. Ouput sinyal PWM pada motor berjumlah 8 pin.
7. Atmega 2560, sebagai pemroses sinyal (Gambar 2.6).
8. Atmega 32, sebagai pendukung fungsi USB.

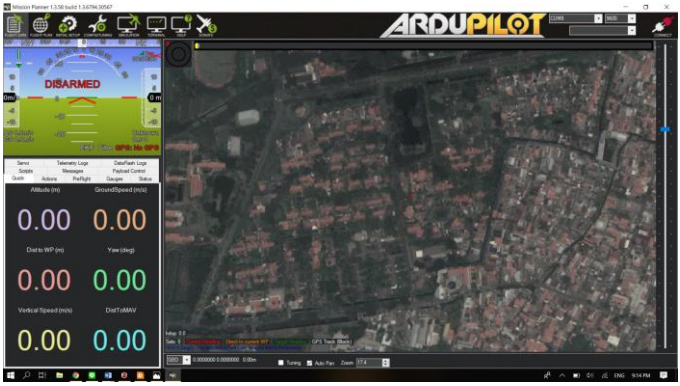


Gambar 2.6 Blok diagram Atmega2560[17]

Pada APM terdapat berbagai macam mode terbang yang dapat digunakan. Pada tugas akhir ini hanya akan digunakan mode stabilize. Stabilize adalah mode dimana APM hanya akan membantu untuk mengkontrol gerak pitch dan roll agar *quadcopter* tetap terbang secara stabil di udara. Pada mode stabilize ketinggian *quadcopter* akan dikendalikan secara manual oleh pilot tanpa ada kontrol dari APM sehingga kecepatan sudut motor akan sebanding dengan input pwm pada *channel* 3 yang diberikan.

2.4 Mission Planner Ardupilot

Mission planner adalah *software* buatan ardupilot yang dikhususkan untuk mengkonfigurasi *flight controller* ArduPilot. Tampilan pada mission planner dapat dilihat pada gambar 2.7. Terdapat berbagai macam fungsi didalam mission planner, seperti pengecekan *real-time* semua sensor yang ada, tracking jalur secara otomatis melalui GPS dan lain-lain. Pada Mission Planner ini juga dapat di atur kontrol output sinyal untuk setiap motor. Mission planner ini dapat di amati secara *real time* saat *quadcopter* terbang dengan cara penambahan modul telemetri pada flight controller dan juga komputer. Penggunaan telemetri ini banyak membantu dalam pengambilan data penerbangan *quadcopter* seperti kecepatan gerakan, posisi GPS, ketinggian, dan kompas [18]. Contoh dari penelitian yang menggunakan Mission Planner ini ialah Kapal pemantau lingkungan menggunakan WyePoint [19]. Pada penelitian tersebut mission planner digunakan untuk menentukan jalur dari kapal tersebut.



Gambar 2.7 Tampilan mission planner

2.5 STM32F103C8T6

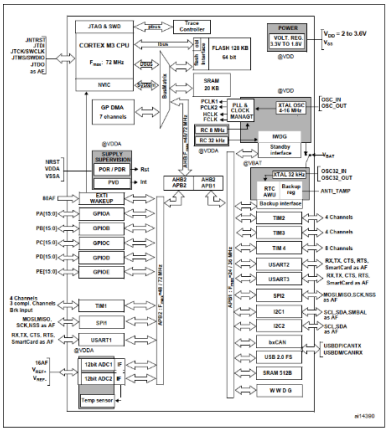
STM32 adalah mikrokontroler 32 bit yang banyak digunakan saat ini. Salah satu jenis dari STM32 adalah STM32F103C8T6. STM32F103C8T6 adalah mikrokontroler produk dari STMicroelectronics. Mikrokontroler jenis ini memiliki fitur sebagai berikut:

1. ARM 32-bit Cortex-M3 CPU Core
2. Frekuensi Maksimum 72 Mhz

3. 64 KB flash Memori dan 20 KB SRAM
4. 7 DMA *Controller*
5. 7 buah *Timer*
6. 16 *channel* ADC
7. 2 SPI
8. 2 buah *channel* I2C
9. 3 buah *channel* USART
10. Komunikasi CAN (2.0B)
11. USB 2.0

Gambar 2.8 STM32F103C8T6[20]

Development Board yang menggunakan mikrokontroler STM32F103C8T6 ini ialah *Development Board* keluaran dari Blue Pill (Gambar 2.8) . Produk keluaran Blue Pill ini sudah di lengkapi dengan *pin header* di setiap pinnya untuk mempermudah penggunaan mikrokontroler tersebut. Selain itu juga sudah di sediakan pin header untuk mempermudah pen-*download*-an program menggunakan ST-Link dari komputer. Mikrokontroler STM32F103C8T6 memiliki diagram blok seperti terlihat pada gambar 2.9.



Gambar 2.9 Blok Diagram STM32F103C8T6[21]

2.6 PID Controller

Kontrol PID ini berupa sinyal kontrol yang beraksi pada error. Aksi kontrol PID ini bertujuan untuk mereduksi error. Sistem PID ini juga berfungsi untuk menentukan kepresisian (kestabilan) suatu sistem. Sistem PID ini terdiri dari tiga parameter yaitu *proportional* (P), *derivative* (D),*integral*(I). PID ini merupakan parameter yang diatur sesuai terhadap output sistem yang diinginkan.

Sebuah standar kontrol PID juga di kenal sebagai *Three term controller*, dimana fungsi transfernya secara umum di tulis dalam bentuk paralel (1) atau bentuk ideal (2) seperti gambar di bawah ini.

$$G(s) = KP + KI\frac{1}{s} + KDs \quad (2.3)$$

$$= KP\ (1 + \frac{1}{TIs} + TDs) \quad (2.4)$$

Dalam waktu diskrit :

$$OutputPID = Kp[E + Ki \sum E\ \Delta t + \text{Kd}\ \frac{\Delta E}{\Delta t}] \quad (2.5)$$

Pada peramaan di atas, Kp adalah *Proportional gain*, KI adalah *Integral gain*, KD adalah *Derivative gain*, TI adalah *Integral Time Constant* dan, TD adalah *Derrivative Time Constant.* Fungsi dari *Three term controller* dapat disimpulkan sebagai berikut:

1. *Proportional Term*, mengkontrol error secara proporsional.
2. *Integral Term*, mengurangi error *steady-state* terhadap kompensasi frekuensi rendah oleh integrator.
3. *Derivative Term*, meningkatkan respon transien terhadap kompensasi frekuensi tinggi oleh differensiator [22].

Efek masing-masing dari ketiga term tersebut dapat diamati pada tabel 2.6.

Tabel 2.5 Tabel efek dari ketiga Term

Closed-Loop Response	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
Increasing Kp	Berkurang	Bertambah	Sedikit Bertambah	Berkurang	Menurun
Increasing KI	Sedikit Berkurang	Bertambah	Bertambah	Banyak Berkurang	Menurun
Increasing KD	Sedikit Berkurang	Berkurang	Berkurang	Perubahan kecil	Bertambah Baik

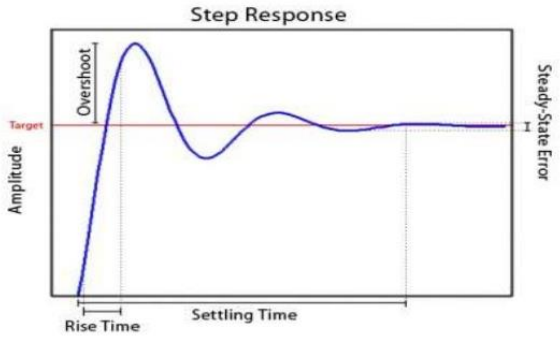
Untuk menentukan nilai dari setiap term tersebut haruslah diakukan tuning secara maual dan coba-coba namun terdapat juga berbagai macam aturan tuning. Salah satu aturan tuning adalah aturan tuning Ziegler Nichols. Pada aturan tuning ziegler nichols terdapat dua metode. Metode yang pertama adalah metode tuning untuk Open-Loop response dan metode kedua digunakan pada close-loop respons . Pada tugas akhir ini digunaan aturan ziegler nichols metode kedua.

Pada aturan ziegler nichols metode kedua, dibutuhkan dua nilai yaitu Kcr dan Pcr [23]. Kcr adalah nilai parameter Kp yang mengakibatkan osilasi biasa di sebut *critical gain*. Sedangkan Pcr adalah periode sinyal osilasi.

Tabel 2.6 Tabel Aturan Ziegler Nichols

Tipe kontroler	Kp	Ti	Td
P	0.5Kcr	0	0
PI	0.45Kcr	Pcr/1.2	0
PID	0.6Kcr	Pcr/2	Pcr/8

Setiap respon dari kontroler PID haruslah diamati untuk mengetahui parameter mana yang harus di tambah maupun di kurangi. Pada pengamatannya dibutuhkan beberapa karakteristik sinyal respon dari kontroler PID seperti terlihat pada gambar 2.10. Karakteristik sinyal tersebut terdiri dari overshoot, steady state error, settling time, dan rise time.



Gambar 2.10 Gambar penjelasan PID

2.7 Tinjauan Pustaka

Tinjauan pustaka bertujuan untuk mempelajari penelitian-penelitian sebelumnya yang dapat menjadi dasar dari tugas akhir ini. Berikut merupakan judul paper atau penelitian yang sudah dilakukan sebelumnya.

2.7.1 *Autonomous takeoff and landing of a quadcopter*[24]

Penelitian ini berfokus pada pengimplementasian metode untuk mengotomatisasi *take off* dan *landing* suatu *quadcopter*. Perhitungan ketinggian dari *quadcopter* menggunakan lidar, *intertial unit* dan kalman filter. Hasil dari penelitian didapatkan bahwa dengan *polynomial kinematic trajectories* dan *low cost sensors* berhasil untuk malukukan *take-off* dan *landing* secara otomatis.

2.7.2 *Autonomous quadcopter terrain-following with a laser rangefinder and gimbal system*[25]

Penelitian ini memperkenalkan metode untuk dapat membuat UAV mempertahankan ketinggian pada perbuahan medan terbang dari UAV tersebut. Metode tersebut menggunakan *laser rangefinder* yang dipasang pada gimbal. Pengkontrolan gimbal menggunakan arduino. Algoritma yang digunakan pada penelitan tersebut membutuhkan keakuratan sinyal GPS untuk mempertahankan posisi. Hasil dari penelitian tersebut didapatkan keberhasilan pada penerapannya dan dapat digunakan pada beberapa tujuan yang membutuhkan *terrain-following.*

2.7.3 *Low-cost, real-time obstacle avoidance for mobile robots*[26]

Penelitian ini berfokus pada pengimplementasian *low-cost hardware* dan algoritma untuk menghindari objek diam. Penelitian tersebut menggunakan modul lidar satu titik. Penelitian tersebut menghasilkan sebuah kemungkinan untuk menggunakan *low-cost hardware* untuk memetakan ruangan dan menghindar objek. sistem tersebut dapat di pasang pada badan robot termasuk juga *quadcopter.*

2.7.4 *An adaptive neuro PID for controlling the altitude*[27]

Penelitian ini berfokus pada pengimplementasian metode kontrol *adaptive* PID yang dapat menghasilkan koefisien untuk kontrol ketinggian secara adaptif terhadap keadaan *quadcopter.* Struktur dari kontrol PID ini mirip dengan *Artificial Neuron* yang digunakan pada *artificial neural network.* Kontrol ketinggian *quadcopter* oleh kontroler jenis ini di tampilkan dalam bentuk sinusoidal dan pengeliminasian gangguan yang masuk dilakukan oleh *adaptive neuro* PID juga diamati. Hasil yang didapatkan dari penelitian tersebut

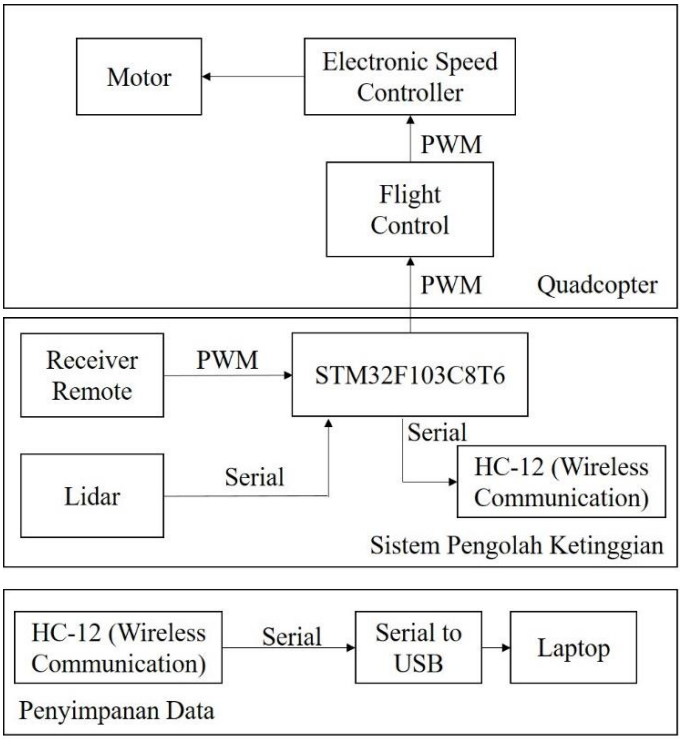
adalah kontroler yang digunakan yaitu *artificial neuron PID*, dapat mempertahankan koefisien kontrol secara adaptif berdasarkan keadaan *quadcopter*. Kelebihan dari kontroler ini adalah tidak dibutuhkannya analisa untuk mendefinisikan koefisien yang tepat. Kekurangan dari kontroler tersebut adalah koefisien yang dihasilkan tidak dapat di jamin akan berhasil.

(Halaman ini sengaja dikosongkan)

BAB III
PERANCANGAN SISTEM

3.1 Diagram Blok Sistem

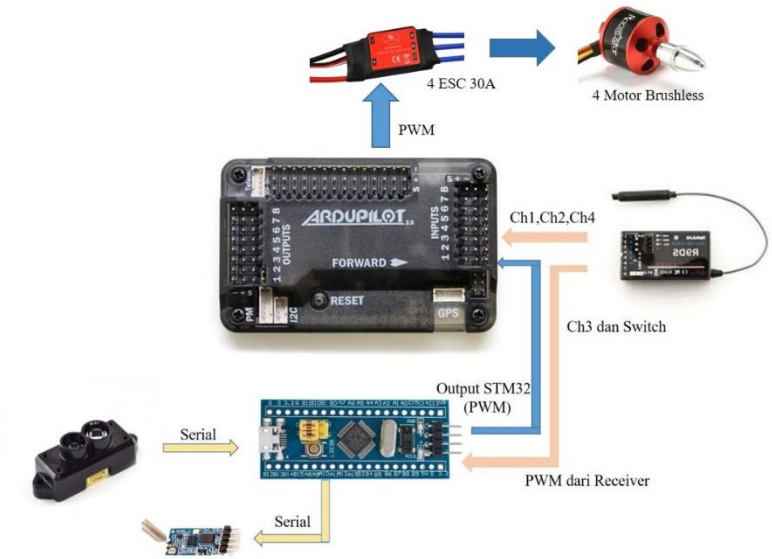
Pada tugas akhir ini terdapat tiga sub sistem yang memiliki fungsi masing-masing. Sub sistem tersebut adalah *quadcopter*, Sistem pengolah ketinggian, dan Penyimpanan data. Diagram blok sistem keseluruhan dapat dilihat pada gambar 3.1.



Gambar 3.1 Diagram Blok Sistem Keseluruhan

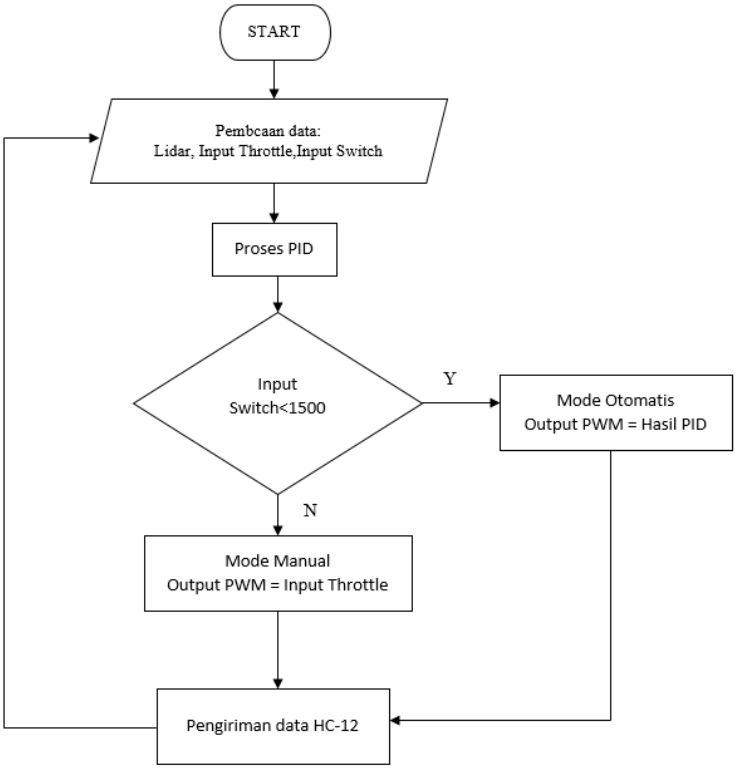
ArduPilot Mega adalah sebuah mikrokontroler yang berinti Atmega2560. ArduPilot Mega di rancang khusus sebagai mikrokontroler untuk menggerakkan berbagai jenis *Autonomous Vehicle*. Pada tugas akhir ini ArduPilot Mega digunakan untuk mengkontrol keempat motor *brushless* pada *quadcopter*.

STM32F103C8T6 adalah mikrokontroler 32bit dengan berbagai macam fungsi didalamnya. Pada tugas akhir ini STM32F103C8T6 digunakan untuk mengolah berbagai macam data. Data-data yang di olah ialah data lidar dan data *input* PWM. Pemilihan STM32F103C8T6 ini dikarenakan memiliki lebih dari satu komunikasi serial, 4 timer, dan frekuensi sinyalnya yang tinggi yaitu 72Mhz. Jika dibandingkan dengan mikrokontroler sejenis yang memiliki fitur yang sama, STM32F103C8T6 memiliki harga yang lebih murah.



Gambar 3.2 Sketsa Sistem Keseluruhan

Lidar sebagai input dari sistem akan diolah oleh STM32F103C8T6 untuk menghasilkan *output* berupa sinyal PWM. Sinyal PWM yang dihasilkan oleh STM32F103C8T6 akan memiliki karakteristik sinyal seperti yang dihasilkan oleh *receiver remote. Duty cycle* dari Sinyal PWM *output* sistem tersebut adalah hasil dari pengolahan sistem berupa kontroler PID. *Input* sinyal dari receiver juga akan di olah oleh STM32F103C8T6. Terdapat dua *input* PWM dari *receiver remote* yaitu *channel* 3 (*Throttle*) dan *channel* 5 (*Switch A*).

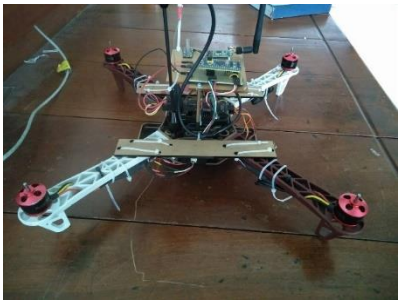


Gambar 3.3 Flow Chart Sistem Keseluruhan

Terdapat dua mode pada sistem ini seperti terlihat pada gambar 3.3 yaitu mode manual dan mode otomatis. Pada mode manual, sinyal *throttle* yang masuk pada STM32F103C8T6 akan di teruskan menuju ArduPilot Mega. Mode manual ditandai dengan sinyal *Switch* A memiliki lebar pulsa 1900 hingga 2000 us. Mode yang kedua adalah mode otomatis. Pada mode otomatis sinyal *input throtle* tidak akan di teruskan, namun akan di bangkitkan sinyal pengganti untuk diteruskan pada Ardu Pilot Mega. Sinyal pengganti ini adalah sinyal hasil pengolahan sistem berupa kontroler PID dengan input nilai pembacaan dari lidar.

Hasil pengolahan dari lidar dan *input* PWM pada Mikrokontroler STM32F103C8T6 akan dikirimkan oleh HC-12. HC-12 adalah sebuah komunikasi serial yang mengirimkan data secara nirkabel dengan

frekuensi 915Mhz. Pengiriman data ini dilakukan untuk menyimpan data secara langsung pada laptop di permukaan tanah. Selain sebagai penyimpanan data, pengiriman secara langsung ini juga dilakukan untuk mempermudah pengolahan data karena tidak perlu mengakses mikrokontroler secara langsung dan dapat dilihat perubahan data secara *real-time.*

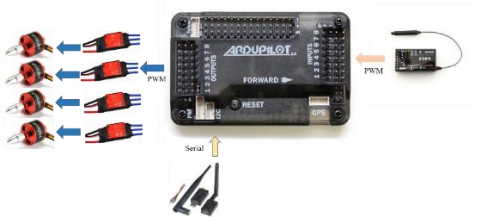


Gambar 3.4 Hasil Perancangan Keseluruhan

3.2 Perancangan Perangkat Keras

Pada subbab ini akan di rancang perangkat keras dari tugas akhir ini, perangkat keras ini meliputi elektronik dan mekanik. Perangkat keras pada Tugas akhir ini di bagi menjadi dua yaitu perangkat keras untuk pembuatan *quadcopter* dan perangkat keras untuk sistem pengatur ketinggian *quadcopter*. Perangkat keras untuk pembuatan *quadcopter* terdiri dari rangka (*frame*), *Electronic Speed Contro*l (ESC), motor *brushless*, *Power Distribution Board*, dan *ArduPilot Mega* (APM). Perangkat keras untuk perancangan sistem pengatur ketinggian ialah, sensor lidar, PCB mikrokontroler, dan modul HC-12.

3.2.1 Perancangan Quadcopter



Gambar 3.5 Perancangan Quadcopter

1. Rangka (*Frame*)

Frame yang akan digunakan pada tugas akhir ini adalah *frame* F450. *Frame* F450 memiliki lebar *frame* 450mm dengan empat lengan dimana panjang setiap lengan 200mm dan ketinggian 55mm. Pemilihan frame ini dikarenakan frame termasuk besar sehingga mempermudah dalam hal penataan kabel.

Gambar 3.6 *Frame Quadcopter* ukuran F450[31]

2. Motor *Brushless*

Gambar 3.7 Motor *Burshless*[32]

Pemilihan motor *brushless* pada tugas akhir ini dikarenakan mampu mengeluarkan *thrust* yang tinggi dengan ukuran motor yang kecil dan ringan. Pada tugas akhir ini digunakan motor brushless 1000 Kv.

3. *Electronic Speed Control* (ESC)

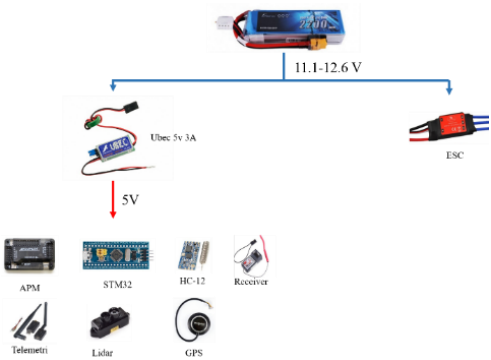
Gambar 3.8 Electronic Speed Control[33]

Electronic Speed Control adalah rangkain pengontrol kecepatan pada motor *brushless*. Kelebihan dari ESC adalah mampu mengeluarkan arus

tinggi. ESC yang digunakan pada tugas akhir ini adalah esc dengan output arus hingga 30A. pemilihan ESC ini berdasarkan kebutuhan arus pada motor. ESC ini membutuhkan sinyal input berupa sinyal PWM dengan lebar pulsa antara 1ms – 2ms dengan periode sinyal 20ms. Lebar pulsa dari sinyal input PWM inilah yang akan menentukan kecepatan dari motor *brushless*.

4. *Power Distribution Board* (PDB)

Gambar 3.9 Power Distribution Board pada frame F450[34]



Gambar 3.10 Distribusi listrik secara keseluruhan

Power Distribution board adalah sebuah papan yang bertujuan untuk mendistribusikan arus listrik dari baterai. *Power distribution board* akan mendistribusikan arus listrik menuju ke empat ESC dan menuju ubec. Ubec disini berfungsi untuk membangkitkan tegangan 5v. Ubec akan menjadi penyuplai tegangan 5v pada APM, STM32, Receiver, Lidar, Hc-12, dan telemetri.

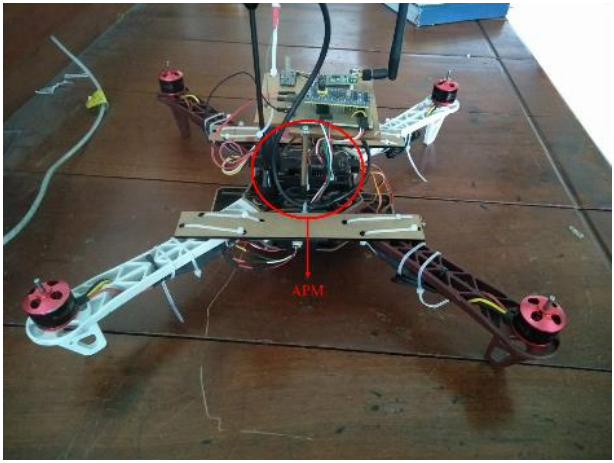
5. Baterai

Gambar 3.11 Battery Lipo 3 Cell[35]

Baterai yang digunakan pada tugas akhir ini adalah baterai lithium polimer 3 sel dengan kapasitas 2200 mah. Pemilihan baterai lipo 3 sel ini berdasarkan kebutuhan esc dan motor.

6. ArduPilot Mega (APM)

Pada perancangan mekanik, APM akan di letakkan ditengah *frame*. APM bekerja pada tegangan 5v sehingga APM akan disuplai oleh baterai melalui Ubec 5v.



Gambar 3.12 Peletakan APM pada Quadcopter

7. *Remote Control* dan *Receiver*

Remote Control yang digunakan pada tugas akhir ini adalah Fly-Sky FS-T6. *Remote Control* ini beroperasi pada frekuensi 2.40 Ghz – 2.48 Ghz dengan *channel* yang dapat di kirimkan sebanyak 6 buah *channel*.

Channel-channel tersebut memiliki fungsinya masing-masing pada *flight controler*. *Channel* 1 berfungsi untuk menggerakkan *quadcopter* maju dan mundur. *Channel* 2 berfungsi untuk menggerakkan *quadcopter* ke kanan dan ke kiri. *Channel* 3 berfungsi untuk menggerakkan *quadcopter* naik dan turun. *Channel 4* berfungsi untuk menggerakkan *quadcopter* untuk menghadap ke kanan dan ke kiri.

Gambar 3.13 Remote dan receiver[36]

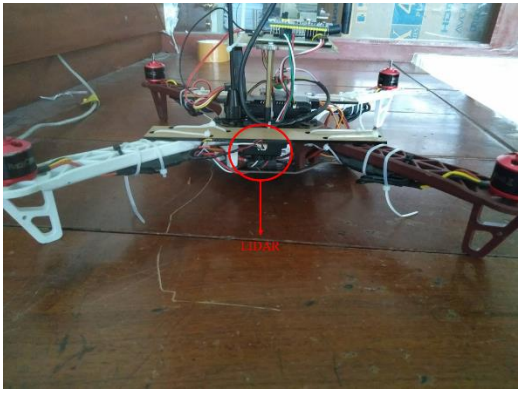
8. Telemetri

Gambar 3.14 Telemetri[37]

Telemetri adalah perangkat komunikasi serial. Telemetri pada tugas akhir ini digunakan untuk mempermudah mengamati kondisi APM secara *wireless*. Telemetri yang digunakan pada tugas akhir ini adalah telemetri dengan frekuensi sinyal 915 Mhz dikarenakan pada tugas akhir ini juga akan digunakan hc-12 yang bekerja pada frekeuensi sinyal 433Mhz. Hal ini dilakukan untuk meminimalisir interferensi sinyal antar kedua perangkat komunikasi tersebut

3.2.2 Perancangan Sistem Pengatur Ketinggian

1. Lidar



Gambar 3.15 Peletakan Lidar pada Quadcopter

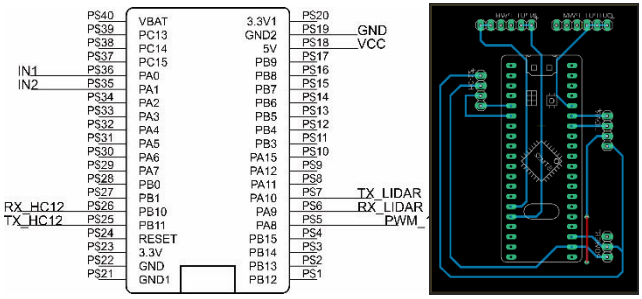
Pada perancangannya, lidar akan diletakkan pada sisi quadcopter bagian kiri namun tidak terlalu jauh dari titik tengah quadcopter (gambar 3.15). Karena lidar bekerja pada tegangan 5v maka suplai tegangan pada lidar akan di ambil dari baterry melalui ubec. Sedangkan kabel tx dan rx akan disambungkan dengan tx dan rx serial pada mikrokontroler.

2. HC-12

Gambar 3.16 HC-12[38]

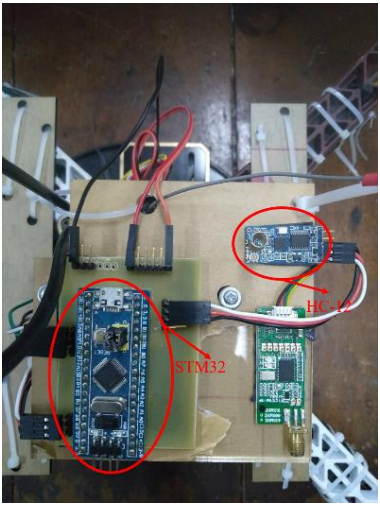
HC-12 adalah modul komunikasi serial *wireless*. Pada perancangan pengatur ketinggian, HC-12 berfungsi sebagai pengirim data-data yang diolah didalam mikrokontroler. Data-data ini berupa data pembacaan lidar, input sinyal, pembacaan pwm, hasil PID, dan lain-lain. Hc-12 bekerja pada frekuensi sinyal 433 Mhz.

3. Board Mikrokontroler



Gambar 3.17 Perancangan board mikrokontroler

Board mikrokontroler adalah papan sirkuit yang sudah di desain menggunakan aplikasi EAGLE. Pada tugas akhir ini board mikrokontroler yang dibuat dapat dilihat pada gambar 3.17. Board mikrokontroler ini digunakan untuk mensederhanaan koneksi antara mikrokontroler dan komponen-komponen seperti HC-12, lidar, dan sebagainya. Penggunaan board mikrokontroler ini juga untuk mengurangi resiko salah pasang yang dapat menyebabkan kerusakan pada berbagai komponen.



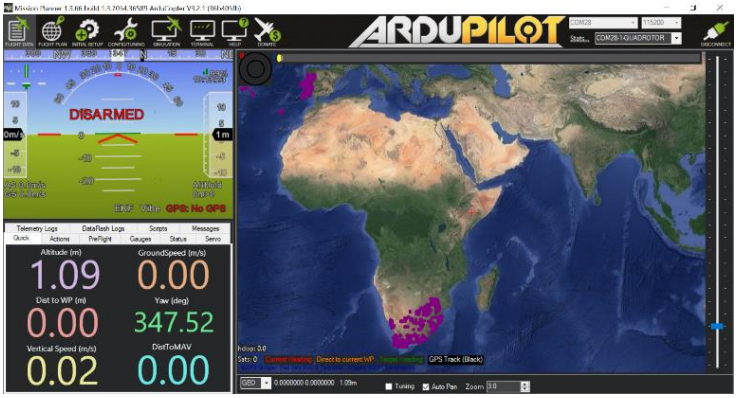
Gambar 3.18 Peletakan STM32 dan HC-12

3.3 Perancangan Perangkat Lunak

Perangkat lunak yang di maksud disini ialah *software-software* yang di gunakan pada Tugas Akhir ini. *Software-software* tersebut terdiri dari Mission Planner untuk mengatur APM, STM32CubeMX dan Atollic True Studio untuk mengatur Mikrokontroler STM32.

3.3.1 Mission Planner

Mission planner berfungsi sebagai media untuk mengamati nilai-nilai yang berada di dalam APM. Nilai-nilai yang dapat diamati dari APM ini ialah, nilai sensor, kemiringan quadcopter, sinyal input pwm yang masuk, nilai output motor yang di keluarkan di setiap motornya. Selain mengamati nilai-nilai di dalam APM, mission planner juga berfungsi sebagai jembatan untuk mengkalibrasi berbagai macam sensor yang tersedia didalam APM.



Gambar 3.19 Tampilan Mission Planner

Sensor-sensor di dalam APM harus di kalibrasi saat pertama kali pemakaian. Sensor-sensor yang harus di kalibrasi tersebut adalah sensor magnetometer/kompas, Acceleromter dan Gyroscope. Kedua sensor tersebut sangat berperan dalam pengendalian keempat motor dari *quadcopter*.

1. Pengkalibrasian APM

Kalibrasi adalah hal yang harus dilakukan sebelum APM digunakan. Kalibrasi ini memiliki tujuan untuk menyeseuaikan nilai sensor dengan keadaan sebenarnya. Pengkalibrasian sensor di dalam APM sudah

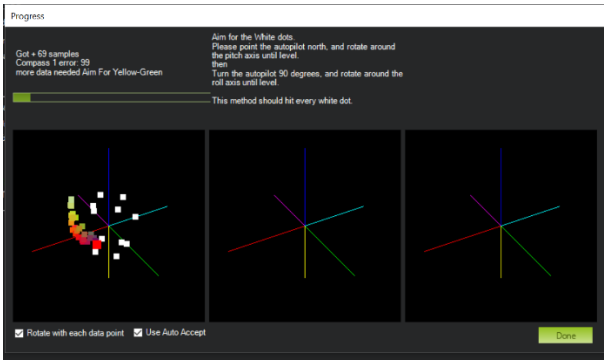
dipermudah dengan menggunakan *interface* dari mission planner. Hal-hal yang harus di kalibrasi pada APM ialah Sensor Accelerometer, Gyroscope, Sensor Magnetometer/Kompas , dan input PWM.

Gambar 3.20 Menu Kalibrasi Accelerometer

Sensor Accelerometer dan Gyroscope di kalibrasi dengan memilih sub-menu Accel Calibration pada menu initial setup (Gambar 3.20) . Kalibrasi sensor ini dilakukan dengan menggerakkan APM menuju sudut-sudut tertentu seperti yang sudah mission planner arahkan. Hasil dari kalibrasi sensor ini akan diolah untuk mengetahui orientasi dari *quadcopter*.

Gambar 3.21 Menu Kalibrasi Magnetometer

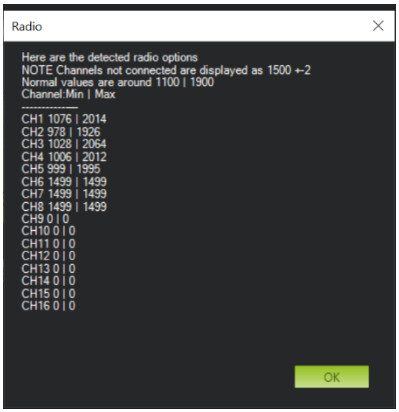
Sensor Magnetometer dikalibrasi dengan memilih sub-menu Compass pada initial setup (Gambar 3.21) . Kalibrasi sensor ini dilakukan dengan memutar-mutar APM hingga mendapatkan semua titik putih yang muncul di mission planner. Hasil dari kalibrasi sensor ini berupa nilai offset yang muncul pada mission planner.



Gambar 3.22 Proses Kalibrasi Sensor Magnetometer

Kalibrasi input PWM ini berupa kalibrasi nilai-nilai yang dikirimkan oleh *receiver remote* menuju APM. Untuk masuk pada mode kalibrasi ini dapat memilih sub-menu Radio Calibration pada menu initial setup, kemudian di pilih calibrate radio. Pada mode kalibrasi remote di gerakkan untuk menuju nilai maksimal dan minimal untuk setiap channel (Gambar 3.23). Hasil dari kalibrasi input pwm ini ialah nilai pwm maksimal dan minimal untuk setiap channel yang digunakan (Gambar 3.24).

Gambar 3.23 Proses Kalibrasi Input PWM dari remote



Gambar 3.24 Hasil kalibrasi input pwm remote

2. Pengamatan nilai-nilai pada APM

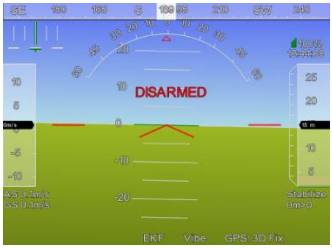
Pengamatan nilai-nilai pada APM berfungsi sebagai pengecekan APM sebelum take-off untuk menjalankan perintah di udara. Nilai pertama yang di cek ialah nilai input pwm pada APM. Input PWM pada APM terdiri dari lima channel yaitu Roll pada channel 1, Pitch pada channel 2, Throttle pada channel 3, Yaw pada channel 4, dan Switch pada channel 5 untuk menentukan mode APM.

Gambar 3.25 Pengecekan nilai input PWM

Pengecekan selanjutnya yaitu pada mode yang di pakai. Seperti di jelaskan sebelumnya, channel 5 digunakan untuk menentukan mode yang di pakai. Pada APM tediri banyak sekali mode terbang yang tersedia, namun pada tugas akhir ini hanya di gunakan tiga mode terbang yaitu Stabilize, Loiter, dan Auto.

Gambar 3.26 Pengecekan mode terbang yang digunakan

Pengecekan yang ketiga yaitu pengecekan sensor-sensor yang digunakan. Sensor-sensor yang digunakan pada suatu keadaan APM tergantung pada mode apa yang digunakan. Untuk mode *Altitude Hold* digunakan semua sensor yaitu Barometer, Magnetometer, Gyroscope dan Accelerometer. Sedangakan pada mode *Stabilize* hanya digunakan tiga sensor yaitu magnetometer, accelerometer dan Gyroscope. Barometer adalah sensor untuk mengetahui ketinggan dari *quadcopter* berdasarkan nilai tekanan udara di sekitarnya. Magnetometer adalah sensor kompas untuk mengetahui orientasi dari *quadcopter*. Gyroscope dan accelerometer digunakan sebagai sensor kemiringan untuk mengetahui kemiringan dari *quadcopter*. Pengecekan sensor sudah terkalibrasi atau belum dapat di amati dari tampilan pada gambar 3.24.



Gambar 3.27 Tampilan untuk pengecekan sensor

Pada gambar 3.27 dapat di amati nilai arah yang merupakan hasil dari sensor magnetometer, kemiringan quadcopter yang merupakan hasil dari sensor accerometer dan gyroscope, dan ketinggian yang merupakan hasil dari sensor barometer.

Pengecekan yang terakhir yaitu pengecekan output APM pada motor brushless. Pada pengecekan ini motor akan di uji tanpa menggunakan propeler. Hal ini dilakukan untuk keamanan. Saat APM sudah di *arm* maka dapat dilihat pada mission planner nilai output dari setiap motor. Nilai-nilai output motor ini akan tergantung pada input pwm dari remote dan pembacaan sensor-sensor.

Gambar 3.28 Output PWM pada motor

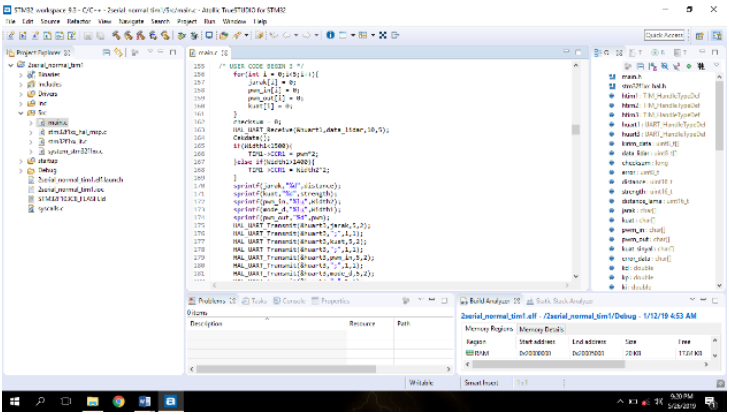
Pada gambar 3.28 dapat dilihat output pwm pada ESC motor untuk menggerakkan motor. Empat nomor teratas menunjukkan output pwm dari APM pada esc motor. Gambar tersebut di dapat saat throttle di naikkan hingga 100%.

Saat semua pengecekan sudah selesai maka *quadcopter* sudah siap untuk diterbangkan.

3.3.2 STM32CubeMX dan Atollic True Studio

STM32CubeMX dan Atollic True Studio adalah dua software yang digunakan untuk mengkonfigurasi Mikrokontroler jenis STM32. Meskipun sama-sama berfungsi untuk mengkonfigurasi STM32, kedua software tersebut memiliki fungsi yang berbeda. STM32CubeMX berfungsi untuk mengkonfigurasi fungsi apa saja yang akan digunakan pada STM32, hasil dari konfigurasi pada STM32CubeMx adalah program

dengan berbagai *library* dan fungsi di dalamnya yang akan di gunakan pada Atollic. Atollic True Studio berfungsi untuk mengolah program berdasarkan konfigurasi yang sudah di tetapkan pada STM32Cube MX. Pada konfigurasi tersebut fungsi-fungsi yang di aktifkan ialah, Penggunaan Timer untuk pembacaan panjang sinyal PWM, Penggunaan timer untuk interrupt pada PID, penggunaan timer pada pembangkitan sinyal PWM, Penggunaan komunikasi USART untuk pembacaan data sensor lidar, penggunaan komunikasi usart untuk pengiriman data secara wireless menggunakan HC-12.



Gambar 3.29 Tampilan Atollic

1. Pembacaan data sensor lidar

Seperti dijelaskan di bab sebelumnya bahwa lidar menggunakan komunikasi USART untuk mengirimkan nilai pembacaan jarak. Lidar menggunakan komunikasi USART dengan baudrate 115200 bit/s. Sehingga pada STM32CubeMX akan dikonfigurasi untuk menggunakan USART3 dengan baudrate 115200.

Setelah baudrate antara lidar dan mikrokontroler sudah di samakan selanjutnya ialah pembacaan data lidar. Setiap cyclenya Lidar mengirimkan 9 byte data berisi 2 byte nilai jarak, 2 byte nilai kekuatan sinyal, 2 byte nilai header, dan 1 byte nilai checksum. Nilai header dan checksum digunakan untuk memastikan data-data penting di setiap sekali pengiriman data lidar bernilai benar.

```
int Cekdata(void){
      if(data_lidar[0] == 'Y'){
      if(data_lidar[1] =='Y'){
      checksum += data_lidar[1];
      if(data_lidar[2]=='Y'){
      for(int i=2;i<9;i++){
      checksum += data_lidar[i];}
      checksum &= 0xff;
      if(checksum==data_lidar[9]){
      distance = data_lidar[4]<<8;
      strength = data_lidar[6]<<8;
      distance = distance|data_lidar[3];
      strength = strength|data_lidar[5];
}}}}}}
```

Pada data nilai jarak dan nilai kekuatan sinyal memiliki data terpisah menjadi dua byte data sehingga setelah data di terima maka data tersebut harus di satukan menjadi data 16 bit untuk mendapatkan nilai data sebenarnya.

2. Penggunaan *timer* untuk membaca sinyal PWM

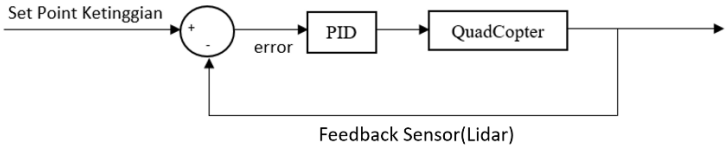
Timer adalah sebuah *clock* yang sudah di perkecil frekuensinya sesuai yang diinginkan pengguna. Pada STM32F103C8T6 terdapat 4 *timer* yang dapat di gunakan untuk berbagai macam tujuan. Salah satu penggunaan *timer* adalah untuk membaca sinyal PWM yang masuk pada pin-pin STM32F103C8T6. PWM adalah suatu sinyal kotak yang dimana panjang sinyal kotak tersebut merepresentasikan nilai input masuk. Pada umumnya PWM bekerja pada periode 20ms, dengan sinyal high berada pada kisaran 1ms – 2ms. Logika dari pembacaan sinyal PWM ini ialah melalui mendeteksi perubahan polaritas dari sinyal PWM. Pada setiap perubahan polaritas ini akan di ambil sebagai tanda mulai dan berhenti *counter*. Nilai *counter* inilah yang akan menjadi hasil pembacaan panjang sinyal PWM.

Pada tugas akhir ini PWM yang akan di hitung berupa sinyal PWM berasal dari *receiver remote controller*. *Receiver remote controller* memiliki 6 *channel* PWM yang dapat dibaca, namun untuk tugas akhir ini hanya di baca 2 sinyal pwm. Sinyal PWM yang akan di baca ialah sinyal PWM untuk *Throttle* dan *Switch 2*.

Mode timer yang akan digunakan untuk membaca nilai PWM ini ialah mode input capture. Input capture adalah sebuah mode timer dimana STM32F103C8T6 akan merespon nilai input apakah dia rising up atau falling down. Pada logika pembacaan PWM, dibutuhkan mode yang dapat meresepon nilai rising up dan falling down secara bersamaan. Namun STM32F103C8T6 tidak memiliki mode yang dapat merespon kedua hal tersebut bersamaan, sehingga akan dilakukan bergantian dalam pengecekan. Saat timer mendapatkan nilai rising up, maka mode timer tersebut akan langsung di ubah menjadi dalam mode falling down untuk dapat merespon nilai akhir dari panjang pulsa tersebut.

Pada pembacaan sinyal PWM menggunakan timer ini juga akan di gunakan fungsi interrupt agar pembacaan sinyal PWM menjadi prioritas di bandingkan pengolahan data lain. Hal ini bertujuan agar *counter* panjang sinyal pwm tidak terdelay.

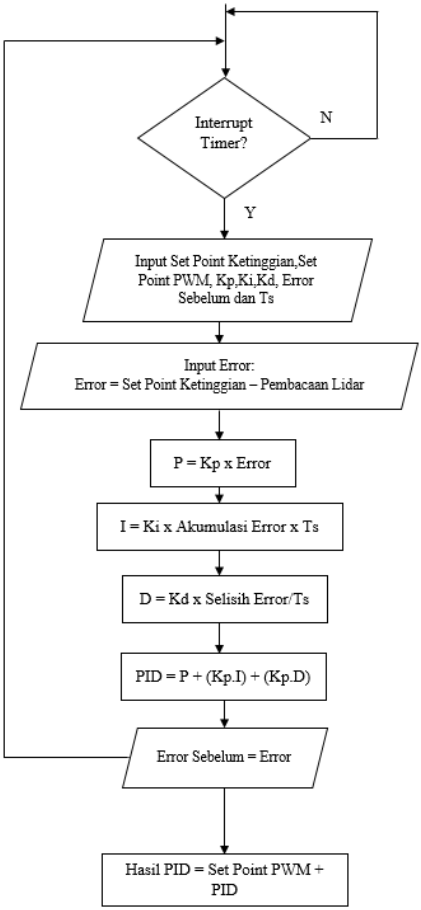
3. Penggunaan interrupt timer untuk PID



Gambar 3.30 Diagram Blok PID

PID pada tugas akhir ini akan digunakan sebagai kontrol *output* pwm yang akan dikirimkan pada APM. PID pada tugas akhir ini akan di buat sesuai diagram blok pada gambar 3.30 . Diagram blok tersebut kemudian di terjemahkan pada program sebagai berikut.

```
error_pid = set_point-distance;
akumulasi_error =+ error_pid;
selisih_error = error_pid - last_error;
p = kp*error_pid;
i = ki*akumulasi_error*Ts;
d = kd*rate/Ts;
pid = p+(kp*i)+(kp*d);
last_error = error_pid;
```



Gambar 3.31 Flow Chart PID

Interrupt timer adalah fungsi lain pada timer yang dapat menjalankan program secara terus menerus dalam periode waktu tertentu. Untuk menggunakan fungsi tersebut maka timer akan di set pada mode interrupt dengan periode waktu tertentu. Interrupt timer ini akan digunakan pada metode penghitungan nilai PID. Penggunaan pada PID ini dikarenakan PID diharuskan mempunyai Time Sampling yang konstan.

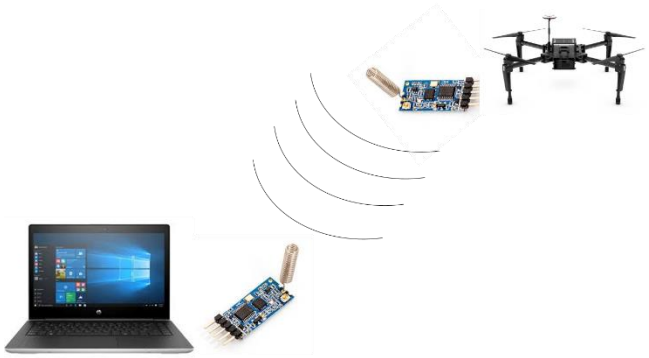
4. Penggunaan timer untuk membangkitkan sinyal PWM

Fungsi lain pada timer ialah untuk membangkitkan sinual PWM. Seperti dijelaskan sebelumnya sinyal PWM adalah sinyal kotak dengan panjang sinyal highnya menentukan data yang ingin dikirimkan. Jika pada fungsi timer digunakan untuk membaca nilai panjang pulsa, pada fungsi timer kali ini akan digunakna untuk membangkitkan sinyal pwm.

Setelah sinyal pwm input diolah di dalam STM32F103C8T6 dan digabungkan dengan pembacaan data lidar dan PID, STM32F103C8T6 akan membangkitkan sinyal PWM. Sinyal PWM yang akan dibangkitkan ini memliki karakter yang sama seperti nilai PWM yang masuk pada STM32F103C8T6, namun berbeda pada panjang pulsa.

Untuk membangkitkan sinyal PWM ini akan digunakan mode output compare pada timer. Input compare ini adalah suatu mode timer dimana nilai output akan bernilai high atau low tergantung sudah mencapai atau belum pada nilai yang sudah di tentukan.

5. Penggunaan komunikasi usart untuk HC-12



Gambar 3.32 Skema penggunaan HC-12

HC-12 adalah modul komunikasi *wireless* dengan menggunakan frekuensi 415 Mhz. Modul Hc-12 ini menggunakan *input* berupa *interface* serial sehingga di butuhkan dua jalur komunikasi yaitu Tx dan Rx. Secara default modul hc-12 ini menggunakan baudrate 9600. Pada penggunaannya dibutuhkan dua HC-12 untuk saling berkomunikasi. Pada tugas akhir ini HC-12 akan diletakkan pada *quadcopter* dan pada laptop di permukaan tanah.

(Halaman sengaja dikosongkan)

BAB IV
PENGUJIAN DAN ANALISIS

4.1 Pengujian Quadcopter

Pada pengujian ini quadcopter yang sudah di rangkai sebelumnya akan di uji orientasi, kestabilan, dan lama terbang. Selama pengujian baterai yang akan di pakai adalah baterai dengan konfigurasi 3 sel 2200 mah. Hasil yang ingin dicapai ialah kestabilan orientasi dan waktu seberapa lama quadcopter dapat terbang.

Gambar 4.1 Pengujian Quadcopter

Dari hasil pengujian didapatkan quadcopter sudah berorientasi dengan benar dan stabil dapat terlihat pada gambar 4.1. pergerakan yang di uji adalah pergerakan *yaw, pitch,* dan roll. Lama waktu terbang dari quadcopter hingga baterai habis rata-rata dari tiga kali percobaan adalah 4 menit 3 detik seperti terlihat pada tabel 4.1.

Tabel 4.1 Tabel waktu lama terbang quadcopter

No.	Lama waktu terbang
1	4 menit 37 detik
2	3 menit 14 detik
3	4 menit 20 detik
Rata-rata	4 menit 3 detik

4.2 Pengujian Lidar

Pada pengujian lidar ini akan di tampilkan hasil dari pengujian sensor lidar pada berbagai nilai jarak yang berbeda. Pengujian ini dilakukan untuk mengetahui karakteristik error dari berbagai macam jenis pembacaan lidar.

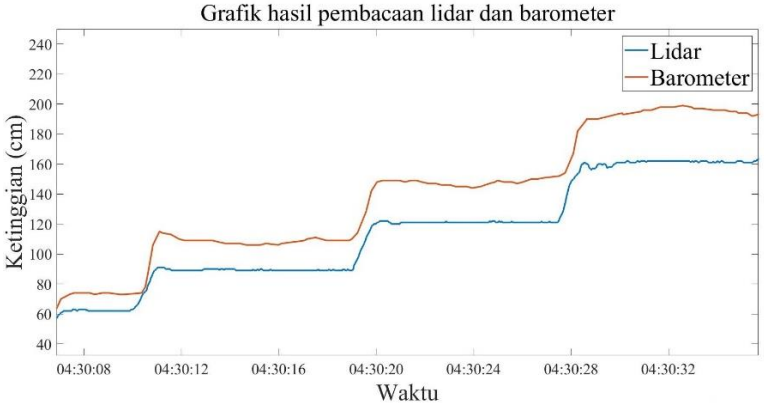
Tabel 4.2 Tabel Hasil pembacaan lidar

Jarak Sebenarnya (cm)	Jarak Pembacaan Sensor (cm)	Error (%)
60	52	13.33
70	66	5.71
80	76	5
90	86	4.44
100	103	3
150	155	3.33
200	192	4
250	247	1.2
300	316	5.33
314	320	1.91
320	336	5
350	367	4.86
400	417	4.25
500	504	0.8
600	604	0.67
	Rata-rata	3.97875

Pada tabel 4.2 di dapatkan data pembacaan lidar dan error dari setiap pembacaan lidar. Rata-rata error dari data di atas adalah 3.97875%. Dari data tersebut juga dapat di lihat bahwa nilai error berubah-ubah tidak beraturan pada setiap pembacaan. Namun dari data tersebut dapat juga di lihat nilai error lebih rendah pada saat pembacaan nilai di atas 100 cm. Sehingga pengujian lidar selanjutnya akan di uji pada nilai mulai dari 100 cm.

4.3 Perbandingan Lidar dan Barometer

Pada pengujian ini hasil pembacaan lidar akan di bandingkan dnegan hasil pembacaan barometer.. Pengujian ini dilakukan untuk mengetahui perbedaan dari hasil pengukuran lidar dan barometer.

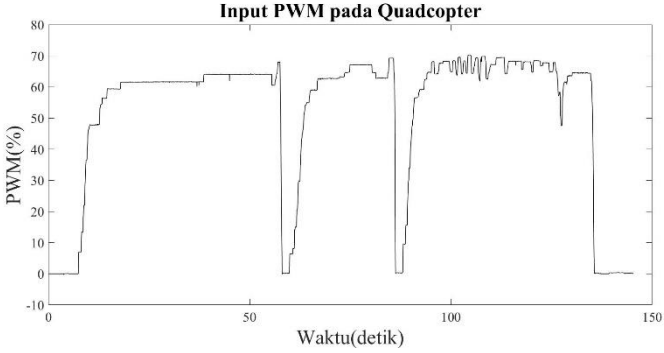


Gambar 4.2 Perbandingan hasil pembacaan lidar dan barometer

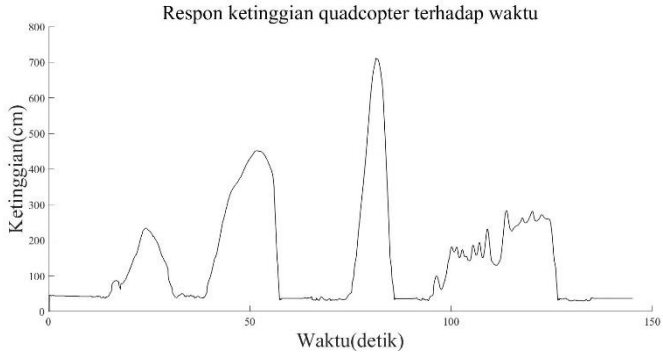
Pada gambar 4.2 di dapatkan data grafik pembacaan lidar dan pembacaan barometer dengan ketinggian yang berubah-ubah. Berdasarkan analisa pada gambar tersebut didapatkan kesimpulan bahwa pembacaan lidar memiliki kestabilan yang lebih baik dari pada barometer.

4.4 Pengujian Lidar pada Quadcopter

Pada pengujian ini lidar akan di pasang pada quadcopter, namun masih belum mempengaruhi sistem quadcopter itu sendiri. Pada pengujian ini hasil yang ingin di capai ialah mengetahui respon ketinggian dengan input pwm *throttle* pada setiap mode yang akan di gunakan pada tugas akhir.



Gambar 4.3 Grafik *input* PWM *quadcopter*



Gambar 4.4 Grafik ketinggian *quadcopter*

Pada pengujian ini didapatkan lidar memiliki pembacaan yang stabil. Dari grafik *input* PWM (gambar 4.2) dan respon ketinggian (gambar 4.3) dapat dilihat bahwa pada *input* PWM yang sama selama beberapa detik ketinggian *quadcopter* akan naik namun akan kembali turun sehingga untuk mempertahankan ketinggian pada mode stabilize dibutuhkan nilai *input* yang berubah-ubah.

4.5 Penerapan Sistem Kontrol Ketinggian pada Quadcopter

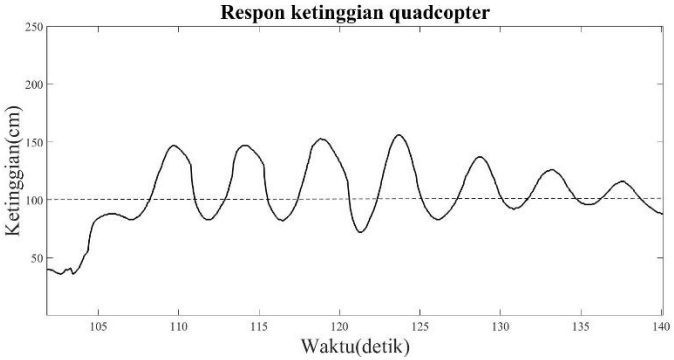
Pada pengujian ini lidar akan di pasang pada mikrokontroler untuk membangkitkan sinyal PWM yang dapat mempengaruhi ketinggian *quadcopter*. Hasil dari pengujian ini ialah *quadcopter* berhasil untuk dapat menuju titik *set point* dan mempertahankan ketinggiannya selama

beberapa detik. percobaan pertama di lakukan pada *set point* bernilai 100 cm.

Pada sistem kontrol ketinggian, seperti yang sudah di jelaskan pada perancangan program, akan menggunakan PID sebagai kontrol nilai *output* PWM dengan *input* nilai pembacaan lidar. Penentuan nilai-nilai parameter PID pada tugas akhir ini akan menggunakan aturan tuning Ziegler Nichols metode kedua dan juga tuning secara manual.

4.4.1 Metode Ziegler Nichols pada *set point* 100 cm.

Pada tuning ziegler nichols metode kedua, pada permulaan tuning Ki dan Kd bernilai 0. Sehingga hanya menggunakan nilai dari kontrol proporsional. Nilai dari kontrol proposional di mulai dari 0 hingga mendapatkan respon sinyal yang berosilasi secara konstan (Kcr). Selanjutnya sistem dan Kp diubah-ubah hingga mencapai osilasi dengan *set point* berada di tengah osilasi. Pada *set point* 100 cm didapatkan Kp bernilai 1 untuk mendapatkankan osilasi *quadcopter* secara konstan.



Gambar 4.5 Grafik ketinggian saat Kp = 1

Grafik pada gambar 4.4 adalah nilai pembacaan lidar pada saat nilai-nilai parameter PID di atur dengan Kp = 1 Ki = 0 Kd = 0. Dari grafik tersebut dapat dilihat bahwa quadcopter berosilasi secara konstan dengan nilai tengah pada *set point* 100 cm.

Tahap selanjutnya dari metode kedua ziegler nichols adalah penentuan penggunaan kontroler P, PI, atau PID. Masing-masing kontroler tersebut memiliki rumus masing-masing untuk menentukan parameter PID. Rumus tersebut di atur pada tabel aturan ziegler nichols.

Pada data sebelumnya di dapatkan Kcr bernilai 1. Karena selisih puncak berbeda-beda maka untuk mendapatkan nilai Pcr (periode osilasi) akan dilakukan dengan cara mencari rata-rata dari selisih waktu antar puncak dari grafik yang didapatkan.

Tabel 4.3 Tabel Perhitungan rata-rata selisih waktu puncak

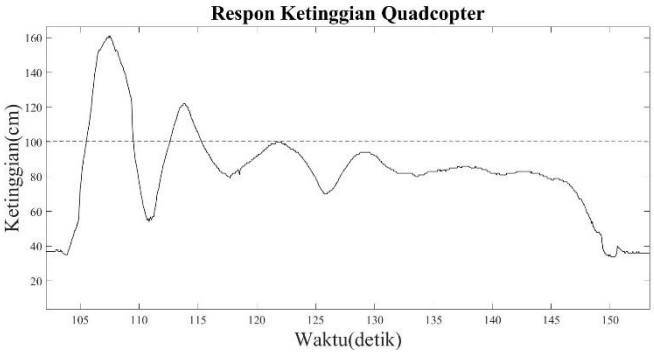
No.	Selisih waktu antar puncak (detik)
1	4.343
2	5.093
3	4.969
4	4.406
5	4.859
6	4.828
Rata-rata	4.749667

Rata-rata dari selisih waktu antar puncak didapatkan dari tabel 4.3 bernilai 4.749667. Setelah di ketahui Kcr dan Pcr maka di dapatkan nilai parameter dari aturan ziegler nichols sebagai berikut.

Tabel 4.4 Tabel hasil Aturan Ziegler Nichols

Tipe kontroler	Kp	Ki	Kd
P	0.5	0	0
PI	0.45	0.252649309	0
PID	0.6	0.421082181	0.593708333

Pada tabel 4.4 di dapatkan nilai parameter untuk setiap kontroler. Kontroler yang akan di uji adalah kontroler P, PI, dan PID.



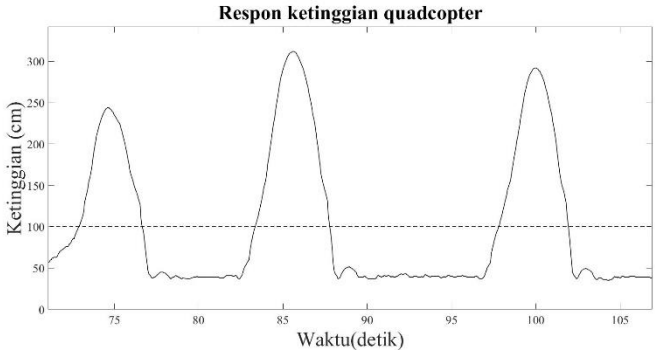
Gambar 4.6 Grafik ketinggian saat menggunakan kontroler P

Grafik kontroler P pada gambar 4.5 didapatkan dari parameter Kp = 0.5 Ki = 0 Kd = 0 dengan inisiasi PWM pada 50.5%. Grafik tersebut terlihat bahwa osilasi dari *quadcopter* berada di bawah nilai *set point*. Karakteristik dari respon kontroler diatas dapat dilihat pada tabel.

Tabel 4.5 Karakteristik respon kontroler P

Overshoot	61cm
Rise time	1.093 detik
Settling time	29.19 detik
Steady State Error	17cm

Pada tabel 4.5 dapat dilihat karakteristik dari grafik bahwa respon ketinggian *quadcopter* mencapai kestabilan ketinggian dalam waktu 29.19 detik. Kestabilan yang didapatkan masih memiliki error *steady state* sebesar 17 cm, dari analisa yang dilakukan didapatkan kesimpulan bahwa hal ini dikarenakan kurang tingginya inisiasi PWM yang dibutuhkan quadcopter utnuk mencapai nilai *set point*.



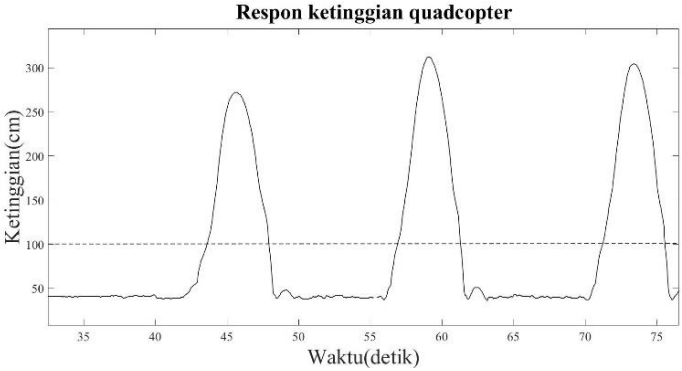
Gambar 4.7 Grafik ketinggian saat menggunakan kontroler PI

Grafik kontroler PI pada gambar 4.6 menggunakan parameter Kp = 0.6 Ki = 0.252649309 dan Kd = 0 . Grafik tersebut hanya di ambil selama 30 detik dikarenakan osilasi yang terus menerus dan menabrak tanah dapat mengakibatkan frame *quadcopter* patah. Dari grafik tersebut dapat diamati bahwa osilasi dari *quadcopter* sudah melewati *setpoint* namun terdapat *delay* waktu yang cukup lama antar setiap gelombang. *Delay* ini disebabkan parameter Ki yang terlalu besar sehingga menimbulkan *rise time* semakin cepat, *overshoot* semakin tinggi, *settling time* yang semakin lama, dan juga kestabilan yang semakin berkurang. Karakteristik dari sinyal tersebut dapat diamati pada tabel 4.6 .

Tabel 4.6 Karakteristik respon kontroler PI

Overshoot	144cm
Rise time	0.906 detik
Settling time	> 30 detik
Steady State Error	Tidak terlihat

Pada tabel 4.6 tentang karakteristik respon dapat dilihat bahwa nilai *settling time* tidak didapatkan secara pasti namun menggunakan pendekatan. Nilai lebih dari 30 tersebut berdasarkan pembacaan terakhir ketinggian setelah turun ketinggiannya pada 30 detik dan berdasarkan osilasi yang cukup tinggi maka di simpulkan bahwa pada waktu lebih dari 30 detik respon kontroler PI masih belum mencapai keadaan *steady-state.*



Gambar 4.8 Grafik ketinggian saat menggunakan kontroler PID

Grafik kontroler PID pada gambar 4.7 menggunakan parameter Kp = 0.6 Ki = 0.421082 dan Kd = 0.593708. Grafik tersebut hanya di ambil dalam waktu 35 detik dikarenakan osilasi yang mencapai permukaan tanah dapat mengakibatkan kerusakan pada *frame quadcopter*. Dari grafik tersebut dapat diamati bahwa osilasi sangatlah tinggi, hal ini dikarenakan parameter Ki yang terlalu besar.

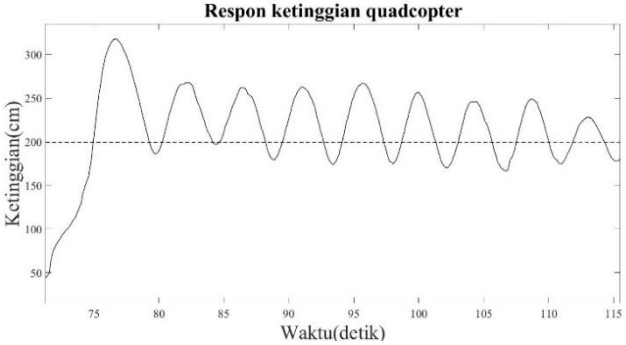
Tabel 4.7 Karakteristik respon kontroler PID

Overshoot	177cm
Rise time	0.672 detik
Settling time	> 35 detik
Steady State Error	Tidak terlihat

Pada tabel 4.7 dapat dilihat perubahaan nilai respon jika dibandingkan dengan parameter PI sebelumnya, nilai dari *overshoot* meningkat dan *rise time* semakin cepat, hal ini sesuai teori yang menyatakan dengan meningkatkan nilai Kp dan Ki maka *overshoot* akan semakin tinggi dan *rise time* semakin cepat.

4.4.2 Metode Ziegler Nichols pada *set point* 200 cm.

Selanjutnya di uji dengan *set point* yang berbeda. Pada pengujian kedua ini digunakan *set point* 200 cm. Pada *set point* kedua ini di dapatkan Kp bernilai 1 untuk menghasilkan respon *quadcopter* yang berosilasi secara konstan.



Gambar 4.9 Grafik ketinggian saat Kp bernilai 1

Pada grafik pada gambar 4.8 dapat dilihat bahwa osilasi sudah berada di tengah set point yang di tentukan yaitu 200 cm. Selanjutnya akan di cari nilai rata-rata selisih waktu antar puncak. Selisih waktu antar puncak dari grafik pada gambar 4.8 dapat dilihat pada tabel berikut.

Tabel 4.8 Tabel Perhitungan rata-rata selisih waktu puncak

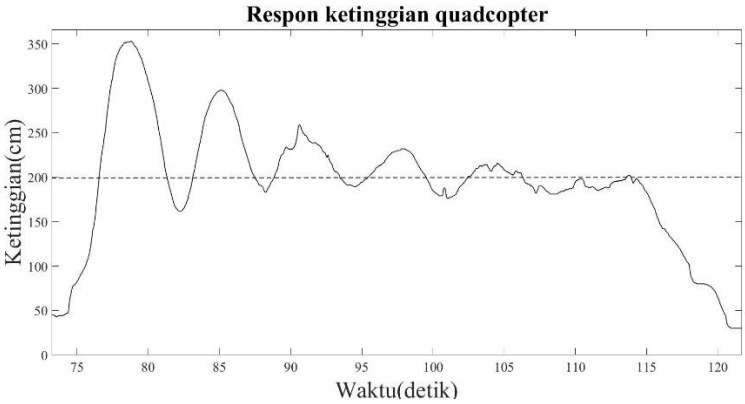
No	Selisih waktu antar puncak (detik)
1	4.718
2	4.386
3	4.687
4	4.406
5	4.281
6	4.562
7	4.235
8	4.234
Rata-rata	4.438625

Setelah di ketahui Kcr dan Pcr maka di dapatkan nilai parameter dari aturan ziegler nichols pada tabel 4.9.

Tabel 4.9 Tabel hasil Aturan Ziegler Nichols pada set point kedua

Tipe kontroler	Kp	Ki	Kd
P	0.5	0	0
PI	0.45	0.270354	0
PID	0.6	0.45059	0.554828

Dari masing-masing kontroler tersebut di dapatkan hasil grafik sebagai berikut.

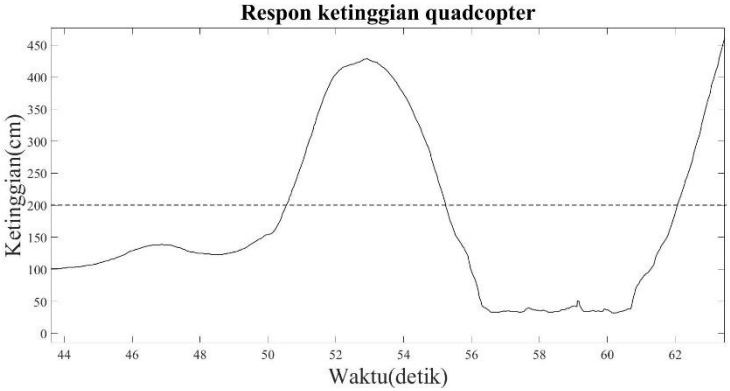


Gambar 4.10 Grafik ketinggian pada kontroler P

Pada grafik pada gambar 4.9 dapat dilihat bahwa *quadcopter* berosilasi dan sedikit demi sedikit menuju *set point*. Karakteristik dari respon ketinggian *quadcopter* pada *set point* 200 cm dan parameter Kp bernilai 1 dapat dilihat pada tabel 4.10.

Tabel 4.10 Karakteristik respon kontroler P

Overshoot	152cm
Rise time	0.766 detik
Settling time	35.486 detik
Steady State Error	12 cm



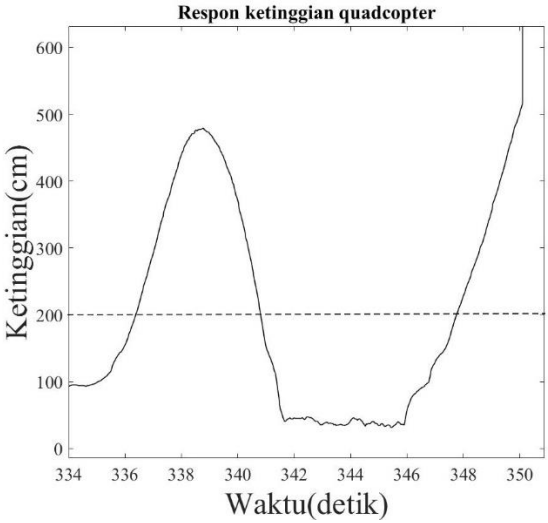
Gambar 4.11 Grafik ketinggian pada kontroler PI

Grafik respon ketinggian pada gambar 4.10 menggunakan parameter Kp = 0.45 dan Ki = 0.270354. Pada grafik tersebut dapat dilihat *quadcopter* memiliki *overshoot* 229 cm, dan juga pada osilasi kedua *quadcopter* berosilasi sangat tinggi sehingga harus segera di ubah ke mode manual untuk menghindari terbang secara tidak terkendali. Karakteristik lain dari respon ketinggian kontroler PI pada *set point* 200cm dapat dilihat pada tabel 4.11.

Tabel 4.11 Karakteristik respon kontroler PI

Overshoot	229 cm
Rise time	0.641 detik
Settling time	>20 detik
Steady State Error	Tidak terlihat

Karakteristik respon yang dihasilkan pada respon kontroler PI di atas dapat dilihat memiliki *overshoot* yang tinggi yaitu 229 cm. *Overshoot* yang tinggi ini di sebabkan oleh peningkatan nilai Kp dan penggunaan parameter Ki. Selain *overshoot* yang tinggi nilai *settling* time juga tidak dapat diketahui secara pasti namun dengan mengamati osilasi yang cukup tinggi di dapatkan kesimpulan bahwa waktu yang di butuhkan untuk mencapai *steady* state lebih dari 20 detik.



Gambar 4.12 Grafik ketinggian pada kontroler PID

Grafik respon ketinggian pada grafik 4.11 menggunakan parameter Kp = 0.6 , Ki = 0.45059 dan Kd = 0.554828. Pada grafik respon tersebut dapat terlihat *overshoot* bernilai 278 cm dengan *rise time* yang berkurang dibandingkan kontroler yang lain, namun pada osilasi kedua ketinggian *quadcopter* meningkat hingga lebih dari 500cm sehingga harus segera diubah menjadi mode manual. Karakteristik lain dari respon pada kontroler PID tersebut dapat dilihat pada tabel 4.12.

Tabel 4.12 Karakteristik respon kontroler PID

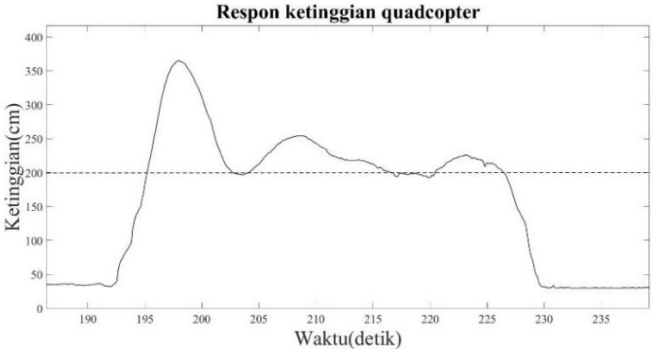
Overshoot	278 cm
Rise time	1.859 detik
Settling time	>30 detik
Steady State Error	Tidak terlihat

Pada dua pengujian menggunakan aturan ziegler-nichols metode kedua di atas di dapatkan kesimpulan kontroler dan parameter terbaik untuk mengontrol ketinggian *quadcopter* menggunakan lidar adalah pada kontroler P dengan parameter Kp = 0.5 Kd = 0 Ki = 0.

4.4.3 Tuning Manual

Tuning PID secara manual dilakukan dengan menambah secara bertahap setiap parameter Kp, Ki, dan Kd. Berdasarkan percobaan menggunakan aturan ziegler-nichols pada subbab sebelumnya, dimana dapat dilihat setiap respon kontroler yang menggunakan Ki memiliki *overshoot* yang sangat tinggi, maka pada tuning secara manual ini parameter yang akan di gunakan adalah parameter Kp dan Kd. Setelah dilakukan berbagai macam perubahan parameter pada Kp dan Kd, di dapatkan dua nilai parameter Kp dan Kd yang mendekati respon ketinggian stabil.

Parameter yang pertama adalah pada Kp bernilai 0.2 dan Kd bernilai 1.5. Pada parameter ini didapatkan grafik pada gambar 4.12.



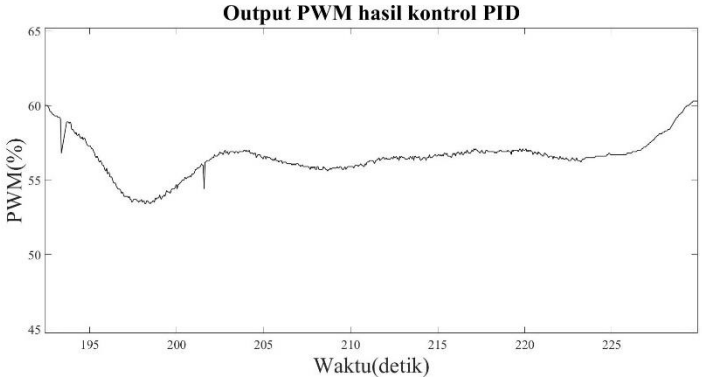
Gambar 4.13 Grafik ketinggian pada kontroler PD pertama

Pada grafik respon ketinggian gambar 4.12 dapat dilihat respon masih memiliki overshoot tapi terlihat stabil menuju *set point*. Karakteristik secara lengkap dari respon ketinggian pada kontroler PD pertama dapat dilihat pada tabel 4.13.

Tabel 4.13 Karakteristik respon kontroler PD pertama

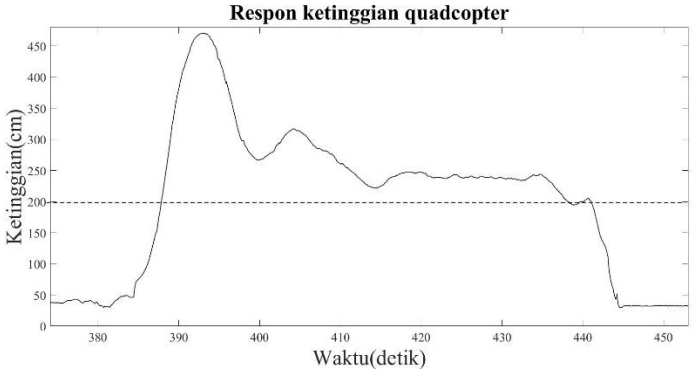
Overshoot	165 cm
Rise time	1.556 detik
Settling time	22.55 detik
Steady State Error	26 cm

Respon yang didapatkan pada parameter PD pertama menggunakan inisiasi PWM pada 56.9%, dengan sinyal PWM yang dihasilkan kontrol PID pada gambar 4.13.



Gambar 4.14 Grafik pwm hasil kontroler PD pertama

Parameter yang kedua adalah pada Kp bernilai 0.1 dan Kd bernilai 1.5. Pada parameter ini didapatkan grafik pada gambar 4.14.



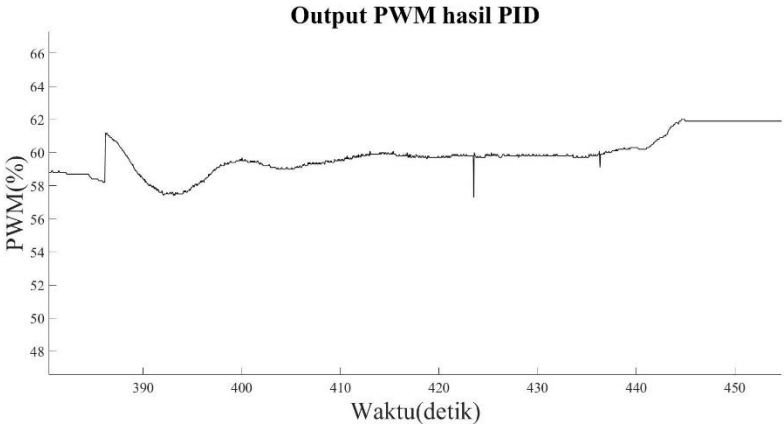
Gambar 4.15 Grafik pwm hasil kontroler PD kedua

Pada grafik gambar 4.14 dapat dilihat respon ketinggian *quadcopter* masih memiliki *overshoot* tapi terlihat stabil namun tidak menuju pada *set point*. Karakteristik respon ketinggian pada parameter PD kedua diatas dapat di lihat pada tabel 4.14.

Tabel 4.14 Karakteristik respon kontroler PD Kedua

Overshoot	270 cm
Rise time	1.601 detik
Settling time	35.653 detik
Steady State Error	39 cm

Respon yang didapatkan pada parameter PD kedua jika dibandingkan dengan parameter PD pertama terlihat cukup aneh. Pada teorinya penurunan nilai Kp seharusnya mengakibatkan *overshoot* berkurang, *rise time* semakin lambat, *settling time* semakin cepat dan error *steady state* bertambah namun pada karakteristik respon kontroler PD kedua terlihat bahwa *overshoot* bertambah dan *settling time* semakin lama. Keanehan ini dapat dijelaskan dengan mengamati *output* kontrol hasil PID di bawah ini.



Gambar 4.16 Grafik output PWM hasil kontroler PD kedua

Output PWM hasil kontroler PD pada gambar 4.15 menggunakan inisiasi PWM 60.2%, dengan adanya pertambahan dari kontroler PD, seperti terlihat pada gambar, *output* sistem naik secara drastis hingga hampir mencapai PWM 62% sehingga *overshoot* sangat tinggi.

Setelah melakukan tuning dengan berbagai macam parameter didapatkan kesimpulan parameter kontrol PID terbaik untuk mendapatkan

kestabilan pada suatu ketinggian menggunakan lidar adalah menggunakan kontrol PD. Pada penelitian ini hasil terbaik yang didapatkan adalah dengan inisiasi PWM sebesar 56.9 % dengan respon yang di hasilkan memiliki *overshoot* sebesar 165 cm, *rise time* selama 1.566 detik, *settling time* selama 22.55 detik dan *steady state error* sebesar 26 cm.

(Halaman sengaja dikosongkan)

BAB V
PENUTUP

5.1 Kesimpulan

Pada tugas akhir ini telah dilakukan berbagai macam percobaan. Percobaan pertama yaitu pengetesan terbang quadcopter dan didapatkan hasil quadcopter yang dapat terbang stabil. Percobaan kedua yaitu pengujian sensor lidar dimana pengujian dilakukan pada beberapa nilai pembacaan dengan membandingkan hasil pembacaan lidar dengan pengukuran sebenarnya. Dari percobaan kedua didapatkan kesimpulan rata-rata error yang pada penggunaan lidar yaitu 3.97875 % dan error pada pengukuran di atas 1 meter memiliki error lebih kecil di bandingkan pengukuran kurang dari 1 meter. Percobaan selanjutnya adalah percobaan pengujian sistem pengatur ketinggian dan didapatkan hasil untuk mempertahankan ketinggiann *quadcopter* menggunakan lidar pada kontroler yang di pakai adalah kontroler PD dengan respon terbaik yang dihasilkan pada tugas akhir ini adalah respon ketinggian quadcopter dengan *overshoot* 165cm, *rise time* selama 1.566 detik, *settling time* selama 22.55 detik dan *steady state error* sebesar 26 cm.

5.2 Saran

Setelah dilakukan penelitian selama mengerjakan tugas akhir dengan judul "Implementasi Lidar Sebagai Kontrol Ketinggian Quadcopter" terdapat beberapa hal yang penulis dapat sarankan untuk mengembangkan penelitian ini. Metode kontrol yang digunakan yaitu PID dapat dilakukan tuning lanjutan, untuk mendapatkan hasil yang lebih baik. Penggunaan metode kontrol lain juga disarankan untuk dapat di bandingkan dengan metode-metode sebelumnya yang sudah dilakukan. Saran selanjutnya yaitu pada pemilihan *flight controller* disarankan untuk memilih *flight controller* yang masih dikembangkan sehingga tanya jawab dapat berlangsung lebih baik dengan pengembang *flight controler*. Saran terakhir dari penulis adalah penggunaan quadcopter yang lebih kecil untuk mendapatkan quadcopter yang lebih stabil.

(Halaman sengaja dikosongkan)

DAFTAR PUSTAKA

[1] F. Muñoz, I. González-Hernández, S. Salazar, E. S. Espinoza, dan R. Lozano, "Second order sliding mode controllers for altitude control of a quadrotor UAS: Real-time implementation in outdoor environments," *Neurocomputing*, vol. 233, hlm. 61–71, Apr 2017.

[2] D. H. S. De Mel, K. A. Stol, J. A. D. Mills, dan B. R. Eastwood, "Vision-based object path following on a quadcopter for GPS-denied environments," dalam *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, Miami, FL, USA, 2017, hlm. 456–461.

[3] D. E. Bolanakis, "Evaluating performance of MEMS barometric sensors in differential altimetry systems," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 32, no. 9, hlm. 34–39, Sep 2017.

[4] P. Denysyuk, V. Teslyuk, dan I. Chorna, "Development of mobile robot using LIDAR technology based on Arduino controller," dalam *2018 XIV-th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, Lviv, 2018, hlm. 240–244.

[5] "Benewake TFMINI Micro LIDAR." [Daring]. Tersedia pada: http://buaya-instrument.com/. [Diakses: 03-Jul-2019].

[6] M. Azizi dan E. Tarshizi, "Autonomous control and navigation of a lab-scale underground mining haul truck using LiDAR sensor and triangulation - feasibility study," dalam *2016 IEEE Industry Applications Society Annual Meeting*, Portland, OR, USA, 2016, hlm. 1–6.

[7] A. N. Catapang dan M. Ramos, "Obstacle detection using a 2D LIDAR system for an Autonomous Vehicle," dalam *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Penang, Malaysia, 2016, hlm. 441–445.

[8] M. R. Shahrin dan F. H. Hashim, "3D Indoor Mapping System Using 2D LiDAR Sensor for Drones," *Int. J. Eng.*, hlm. 5.

[9] I. Sunandar dan D. Syarifudin, "LiDAR : PENGINDERAAN JAUH SENSOR AKTIF DAN APLIKASINYA DI BIDANG KEHUTANAN," vol. 1, hlm. 11, 2014.

[10] www.elecrow.com, "Datasheet Lidar." .

[11] H. S. Khan dan M. B. Kadri, "Attitude and altitude control of quadrotor by discrete PID control and non-linear model predictive control," dalam *2015 International Conference on Information and*

Communication Technologies (ICICT), Karachi, Pakistan, 2015, hlm. 1–11.
[12] M. Mammarella, G. Campa, M. R. Napolitano, M. L. Fravolini, Y. Gu, dan M. G. Perhinschi, "Machine Vision/GPS Integration Using EKF for the UAV Aerial Refueling Problem," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 38, no. 6, hlm. 791–801, Nov 2008.
[13] D. Domingos, G. Camargo, dan F. Gomide, "Autonomous Fuzzy Control and Navigation of Quadcopters," *IFAC-Pap.*, vol. 49, no. 5, hlm. 73–78, 2016.
[14] jose J. Castillo-Zamora, K. A. Camarillo-Gomez, G. I. Perez-Soto, dan J. Rodriguez-Resendiz, "Comparison of PD, PID and Sliding-Mode Position Controllers for V–Tail Quadcopter Stability," *IEEE Access*, vol. 6, hlm. 38086–38096, 2018.
[15] M. F. Ridho, S. Si, dan J. Palembang, "PERANCANGAN PENGENDALI OCTOCOPTER BERBASIS ARDUPILOT MEGA SEBAGAI ROBOT PEMADAM API," hlm. 10.
[16] "Archived:APM 2.5 and 2.6 Overview — Copter documentation." [Daring]. Tersedia pada: http://ardupilot.org/copter/docs/common-apm25-and-26-overview.html. [Diakses: 03-Jul-2019].
[17] "ATmega2560 - 8-bit AVR Microcontrollers." [Daring]. Tersedia pada: https://www.microchip.com/wwwproducts/en/ATmega2560#datasheet-toggle. [Diakses: 03-Jul-2019].
[18] "Pemetaan Distribusi Gas Polutan Menggunakan Quadcopter Berbasis Autonomous Waypoint Navigation," vol. 5, no. 2, hlm. 6, 2016.
[19] F. R. Saputra dan M. Rivai, "Autonomous Surface Vehicle sebagai Alat Pemantau Lingkungan Menggunakan Metode Navigasi Waypoint," *J. Tek. ITS*, vol. 7, no. 1, Mar 2018.
[20] "Blue Pill - STM32duino wiki." [Daring]. Tersedia pada: https://wiki.stm32duino.com/index.php?title=Blue_Pill. [Diakses: 03-Jul-2019].
[21] "STM32F103C8 - Mainstream Performance line, ARM Cortex-M3 MCU with 64 Kbytes Flash, 72 MHz CPU, motor control, USB and CAN - STMicroelectronics." [Daring]. Tersedia pada: https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html. [Diakses: 03-Jul-2019].

[22] Kiam Heong Ang, G. Chong, dan Yun Li, "PID control system analysis, design, and technology," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 4, hlm. 559–576, Jul 2005.
[23] K. Ogata, *Modern control engineering*. Upper Saddle River, NJ: Prentice Hall, 2010.
[24] H. Beck *dkk.*, "Autonomous takeoff and landing of a quadcopter," dalam *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Arlington, VA, USA, 2016, hlm. 475–484.
[25] M. Clark dan R. C. Roberts, "Autonomous quadrotor terrain-following with a laser rangefinder and gimbal system," dalam *2017 IEEE SENSORS*, Glasgow, 2017, hlm. 1–3.
[26] S. Ricardo, D. Bein, dan A. Panagadan, "Low-cost, real-time obstacle avoidance for mobile robots," dalam *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, 2017, hlm. 1–7.
[27] M. Fatan, B. L. Sefidgari, dan A. V. Barenji, "An adaptive neuro PID for controlling the altitude of quadcopter robot," dalam *2013 18th International Conference on Methods & Models in Automation & Robotics (MMAR)*, Miedzyzdroje, 2013, hlm. 662–665.
[28] M. Takahashi, K. Kobayashi, K. Watanabe, dan T. Kinoshita, "Development of prediction based emergency obstacle avoidance module by using LIDAR for mobile robot," dalam *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS)*, Kita-Kyushu, Japan, 2014, hlm. 561–564.
[29] D. Gheorghita, I. Vintu, L. Mirea, dan C. Braescu, "Quadcopter control system," dalam *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, Cheile Gradistei, Romania, 2015, hlm. 421–426.
[30] Z. Mustapa, S. Saat, S. H. Husin, dan T. Zaid, "Quadcopter physical parameter identification and altitude system analysis," dalam *2014 IEEE Symposium on Industrial Electronics & Applications (ISIEA)*, Kota Kinabalu, Malaysia, 2014, hlm. 130–135.
[31] "Elecmake Q330 Glass Fiber Quadcopter Frame 330mm: Amazon.in: Industrial & Scientific." [Daring]. Tersedia pada: https://www.amazon.in/Elecmake-Glass-Fiber-Quadcopter-Frame/dp/B072MJFYCL. [Diakses: 04-Jul-2019].
[32] "Racerstar BR2212 1000KV 2-4S Brushless Motor For RC Models." [Daring]. Tersedia pada: https://www.racerstar.com/s/Racerstar-

BR2212-1000KV-2-4S-Brushless-Motor-For-RC-Models-p-53.html?customer_reviews=1&page=5. [Diakses: 04-Jul-2019].
[33] "CW / CCW LHI 4x 2212 920KV Brushless Motor 4x SIMONK 30A ESC For DJI Phantom LHI-2212-30ESC," *Orli Mautner*. .
[34] "DJI Flamewheel F450 Combo 2 V2.0 (NAZA V2 + Legs)," *Helipal.com*. [Daring]. Tersedia pada: http://www.helipal.com/dji-flamewheel-f450-combo-2-naza-v2-legs.html. [Diakses: 04-Jul-2019].
[35] "Rc hobby battery & uav battery & FPV lipo battery online shop - Gens ace and Tattu." [Daring]. Tersedia pada: https://www.genstattu.com/. [Diakses: 04-Jul-2019].
[36] "Helicopter Radio Control, Helicopter Radio Control - Newegg.com." [Daring]. Tersedia pada: https://www.newegg.com/p/pl?d=Helicopter+Radio+Control,+Helicopter+Radio+Control. [Diakses: 04-Jul-2019].
[37] "433Mhz 500mw Telemetry Set Compatible PX4 & APM - DJI Store Madrid & Support Center StockRc." [Daring]. Tersedia pada: https://tienda.stockrc.com/433Mhz-500mw-Telemetry-Set-Compatible-APM-amp-3DR-Radio/en. [Diakses: 04-Jul-2019].
[38] "CENTIoT HC-12 Wireless Serial Port Module Low Power: Amazon.in: Electronics." [Daring]. Tersedia pada: https://www.amazon.in/SI4463-wireless-serial-consumption-bluetooth/dp/B01MUCVL67. [Diakses: 04-Jul-2019].

LAMPIRAN A

```
#include "main.h"
#include "stm32f1xx_hal.h"
TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;

UART_HandleTypeDef huart1;
UART_HandleTypeDef huart3;

uint8_t kirim_data[10];
uint8_t data_lidar[10] = {0};
long checksum = 0;
uint8_t error;
uint16_t distance = 0;
uint16_t strength = 0;
uint16_t distance_lama = 0;
char jarak[5];
char kuat[5];
char pwm_in[5];
char pwm_out[5];
char kuat_sinyal[5];
char error_data[5];
double kd,kp,ki,pid,p,i,d;
int Ts,set_point,pwm;
uint32_t IC1Value,Width1,IC2Value,Width2;
int error_pid,rate,akumulasi_error;
int last_error = 0;
char mode_d[5];
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);
void HAL_TIM_MspPostInit(TIM_HandleTypeDef *htim);
```

```
int Cekdata(void);
void kontrol_pid(void);
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim);
int main(void)
{
 HAL_Init();
 SystemClock_Config();
 MX_GPIO_Init();
 MX_USART1_UART_Init();
 MX_USART3_UART_Init();
 MX_TIM1_Init();
 MX_TIM2_Init();
 MX_TIM3_Init();
 HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_1);
 HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_1);
 HAL_TIM_IC_Start_IT(&htim2,TIM_CHANNEL_2);
 HAL_TIM_Base_Start_IT(&htim3);

 set_point = 100;
 kp = 0.6;
 ki = 0.013342;
 kd = 18.7375;
 Ts = 10;
 while (1)
 {
        for(int i = 0;i<5;i++){
                jarak[i] = 0;
                pwm_in[i] = 0;
                pwm_out[i] = 0;
                kuat[i] = 0;
        }
        checksum = 0;
        HAL_UART_Receive(&huart1,data_lidar,10,5);
        Cekdata();
        if(Width1<1500){
                TIM1->CCR1 = pwm*2;
        }else if(Width1>1400){
                TIM1->CCR1 = Width2*2;
```

```
            }
            sprintf(jarak,"%d",distance);
            sprintf(kuat,"%d",strength);
            sprintf(pwm_in,"%lu",Width2);
            sprintf(mode_d,"%lu",Width1);
            sprintf(pwm_out,"%d",pwm);
            HAL_UART_Transmit(&huart3,jarak,5,2);
            HAL_UART_Transmit(&huart3,";",1,1);
            HAL_UART_Transmit(&huart3,kuat,5,2);
            HAL_UART_Transmit(&huart3,";",1,1);
            HAL_UART_Transmit(&huart3,pwm_in,5,2);
            HAL_UART_Transmit(&huart3,";",1,1);
            HAL_UART_Transmit(&huart3,mode_d,5,2);
            HAL_UART_Transmit(&huart3,";",1,1);
            HAL_UART_Transmit(&huart3,pwm_out,5,3);
            HAL_UART_Transmit(&huart3,"\r\n",2,2);
  }
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK

|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
```

```
  RCC_ClkInitStruct.SYSCLKSource =
RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);
HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK
);
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}
static void MX_TIM1_Init(void)
{

  TIM_MasterConfigTypeDef sMasterConfig;
  TIM_OC_InitTypeDef sConfigOC;
  TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig;

  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 36;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 39999;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload =
TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_PWM_Init(&htim1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode =
TIM_MASTERSLAVEMODE_DISABLE;
```

```
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1,
&sMasterConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 2000;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC,
TIM_CHANNEL_1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity =
TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.AutomaticOutput =
TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim1,
&sBreakDeadTimeConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  HAL_TIM_MspPostInit(&htim1);

}
static void MX_TIM2_Init(void)
{
```

```
  TIM_ClockConfigTypeDef sClockSourceConfig;
  TIM_SlaveConfigTypeDef sSlaveConfig;
  TIM_MasterConfigTypeDef sMasterConfig;
  TIM_IC_InitTypeDef sConfigIC;

  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 71;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 29999;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload =
TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sClockSourceConfig.ClockSource =
TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) !=
HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  if (HAL_TIM_IC_Init(&htim2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sSlaveConfig.SlaveMode = TIM_SLAVEMODE_RESET;
  sSlaveConfig.InputTrigger = TIM_TS_TI1FP1;
  sSlaveConfig.TriggerPolarity =
TIM_INPUTCHANNELPOLARITY_RISING;
  sSlaveConfig.TriggerFilter = 0;
  if (HAL_TIM_SlaveConfigSynchronization(&htim2, &sSlaveConfig)
!= HAL_OK)
  {
```

```
    _Error_Handler(__FILE__, __LINE__);
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode =
TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2,
&sMasterConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sConfigIC.ICPolarity =
TIM_INPUTCHANNELPOLARITY_RISING;
  sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  sConfigIC.ICFilter = 0;
  if (HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC,
TIM_CHANNEL_1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  if (HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC,
TIM_CHANNEL_2) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
static void MX_TIM3_Init(void)
{

  TIM_ClockConfigTypeDef sClockSourceConfig;
  TIM_MasterConfigTypeDef sMasterConfig;

  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 36000;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
```

```
  htim3.Init.Period = 19;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload =
TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sClockSourceConfig.ClockSource =
TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) !=
HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode =
TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3,
&sMasterConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
static void MX_USART1_UART_Init(void)
{

  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
```

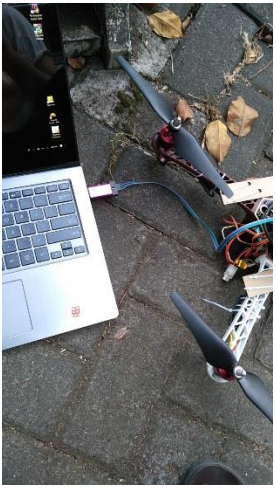
```
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
static void MX_USART3_UART_Init(void)
{

  huart3.Instance = USART3;
  huart3.Init.BaudRate = 9600;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart3) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
static void MX_GPIO_Init(void)
{
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

}
int Cekdata(void){
        if(data_lidar[0] == 'Y'){
          if(data_lidar[1] =='Y'){
                checksum += data_lidar[1];
                if(data_lidar[2]=='Y'){
                        for(int i=2;i<9;i++){
                                checksum += data_lidar[i];
                        }
                        checksum &= 0xff;
                        if(checksum==data_lidar[9]){
```

```
                                        distance = data_lidar[4]<<8;
                                        strength = data_lidar[6]<<8;
                                        distance = distance|data_lidar[3];
                                        strength = strength|data_lidar[5];
                                        return 0;
                                }else{
                                        return 1;
                                }
                        }else{
                                        return 2;
                        }
                }else{
                                return 3;
                }
        }else{
                        return 4;
        }
}

void kontrol_pid(void){

        error_pid = set_point-distance;
        akumulasi_error = error_pid + last_error;
        rate = error_pid - last_error;

        p = kp*error_pid;
        i = ki*akumulasi_error*Ts;
        d = kd*rate/Ts;

        pid = p+(kp*i)+(kp*d);

        last_error = error_pid;

        pwm = 1525+pid;
        if(pwm > 2000) pwm = 2000;
        if(pwm < 1300) pwm = 1300;

}
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim){
```

```
        //Width = 4000;
        if(htim->Instance == TIM2){
                if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_1){

        if(HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_0)==1){
                                        TIM2->CCER|=TIM_CCER_CC1P;
                                        TIM2->CNT=0;
                                }else{
                                        IC1Value = HAL_TIM_ReadCapturedValue(htim,TIM_CHANNEL_1);
                                        TIM2->CCER&=~TIM_CCER_CC1P;
                                        Width1 = IC1Value;
                                }
                        }else if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_2){

        if(HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_1)==1){
                                        TIM2->CCER|=TIM_CCER_CC2P;
                                        TIM2->CNT=0;
                                }else{
                                        IC2Value = HAL_TIM_ReadCapturedValue(htim,TIM_CHANNEL_2);
                                        TIM2->CCER&=~TIM_CCER_CC2P;
                                        Width2 = IC2Value;
                                }
                        }
        }
}
void _Error_Handler(char *file, int line)
{
  while(1)
  {
  }
}

#ifdef  USE_FULL_ASSERT
```

```
void assert_failed(uint8_t* file, uint32_t line)
{
}
#endif
```

LAMPIRAN B



LAMPIRAN C

1 Attentions

1.1 About this Document

- This manual provides all the essential information during the usage of this product.
- Please carefully read this manual and make sure that you fully understand everything herein.

1.2 Usage of Product

- The maintenance work of this product is limited to the professional technician, and the product can only work with the factory spare part for ensuring the performance and safety thereof.
- This product itself has no polarity and over-voltage protection. Please properly connect and supply power as per the manuals herein.
- Operating temperature of this product is between 0℃ and 60℃. Do not use it beyond this temperature range to avoid risk.
- Storage temperature of this product is between -20℃ and 75℃. Do not store it beyond this temperature range to avoid risk.
- For ensuring the product performance, do not open the product shell or do any assembly or maintenance that not listed in this manual.

1.3 Conditions with Potential Product Failure

- The product will be subject to risk of failure if the detecting object has high reflectivity, such as the mirror or the smooth floor tile.
- The product will be subject to risk of failure if there is any transparent object between it and the detecting object, such as glass or water.
- The product will be subject to risk of failure if its transmitting or receiving len is covered by the dust. Please keep the lens clean.
- Please do not directly touch circuit board of the product by hand as it is exposed. Please wear antistatic wrist strap or glove if necessary; Otherwise, the product will be subject to risk of failure, which is shown by failure of normal operation, and even it will be broken.

2 Functions and Key Parameters

2.1 Product Functions

TFmini is a mini LiDAR module. It is mainly capable of the function of real-time and contactless distance measurement and is featured by accurate, stable and high-speed distance measurement.

2.2 Principle of Distance Measurement

TFmini is based on TOF, namely, Time of Flight principle. To be specific, the product transmits modulation wave of near infrared ray on a periodic basis, which wave will reflect after contacting object. The product obtains time of flight by measuring round-trip phase difference and then calculates relative range between the product and the detection object, as shown in Figure1.

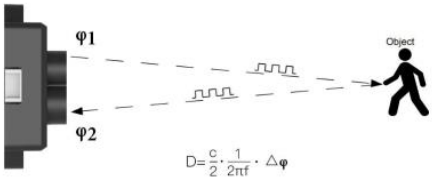


Figure 1 Schematics of TOF Principle

2.3 Key Characteristic Parameters

Table 1 Key Characteristic Parameters of TFmini

Description	Parameter value
Operating Range(Indoor)	0.3m~12m①
Measurement accuracy	±4cm@（0.3-6m）②
	±6cm@（6m-12m）
Default unit of distance	cm
Range resolution	5mm
Receiving half angle	1.15°
Transmitting half angle	1.5°
Frequency	100Hz

① Operating Range reachable under indoor standard white board condition (with reflectivity of 90%);

② A few points may be subject to an error of ±6cm due to switchover of distance measurement position within 0.3-2m.

2.4 Distance Measurement Characteristics

With optimization of light path and algorithm, TFmini has minimized influence from external environment on distance measurement performance. Despite that, the range of distance measurement may still be affected by the environment illumination intensity and the reflectivity of detection object. As shown in Figure 2:

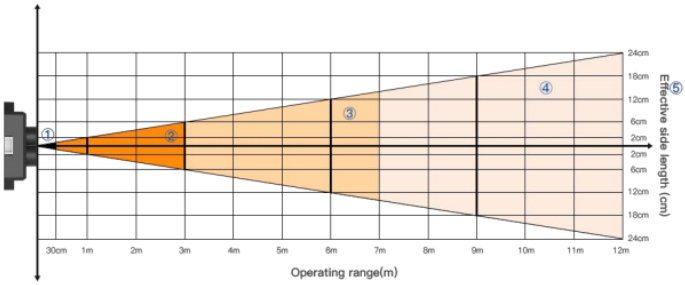


Figure 2 Schematics of Range of distance measurement and Effectiveness of the Product

①: Represents the detection blind area of TFmini, 0-30cm, within which the data is unreliable.

②: Represents the operating range of TFmini under extreme condition, which generally is 0.3-3m. Extreme condition refers to the outdoor glare (of which illumination intensity is around 100klux outdoors at noon in summer) and detection of black target (with reflectivity of 10%).

③: Represents the operating range of TFmini for white target under normal sunshine condition (with illumination intensity of around 70klux), which covers the range of ② and is 0.3-7m.

④: Represents the operating range of TFmini at the indoor environment or considerably weak ambient light environment, which is 0.3-12m.

⑤: Represents the Minimum side length of effective detection for TFmini at the different distances. The data will not be stable and reliable unless "the side length of detection object" is equal to or more than the Minimum side length. The Minimum side length of effective detection depends on the FOV of TFmini (the term of FOV generally refers to the smaller value between the receiving angle and the transmitting angle), which is calculated as follows:

$$d = 2 \cdot D \cdot \tan\beta$$

In the above formula, d is the Minimum side length of effective detection; D is detecting range; β is the half of the value of the receiving angle of TFmini, 1.15°. Correspondence between the Minimum side length of effective detection and detecting range in general is given in Table 2.

Table 2 the Minimum side length of effective detection corresponding to Detecting Range

Range	1m	2m	3m	4m	5m	6m	7m	8m	9m	10m	11m	12m
Minimum side length	4cm	8cm	12cm	16cm	20cm	24cm	28cm	32cm	36cm	40cm	44cm	48cm

(Halaman ini sengaja di kosongkan)

BIODATA PENULIS

Sayyidul Aulia Alamsyah lahir di Sumenep pada tanggal 9 Juli 1998 merupkan anak kedua dari tiga bersaudara. Penulis menyelesaikan pendidikan dasar di SDN Pajagalan 1 Sumenep, dilanjutkan dengan pendidikann tingkat menengah di SMPN 1 Sumenep. Selepas SMP penulis meneruskan pendidikan sekolah menengah di Pondok Pesantren Amanatul Ummah dan bersekolah di MA Amanatul Ummah hingga lulus pada tahun 2015. Penulis meneruskan pendidikan di jenjang perkuliahan di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember Surabaya. Selama masa perkuliahan penulis aktif dalam berbagai kegiatan kepanitiaan dan organisasi mulai dari tingkat jurusan hingga tingkat institut. Selain itu penulis juga menjadi asisten praktikum di bidang studi elektronika.

Email : Sayyidul98@gmail.com

HP/WA : 082330515802

Line : Alam_syah14