Desain Dan Implementasi Kendali Ketinggian Pada Kendaraan Tanpa Awak Menggunakan Kontrol Pid

Design And Implementation Of Altitude Control On Unmanned Aerial Vehicle Using Pid Control

1st Carlo Anthony Balelang
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
carloanthony@student.telkomuniversity.ac.id

2nd Muhammad Ridho Rosa
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
mridhorosa@telkomuniversity.ac.id

3rd Muhammad Zakiyullah Romdlony
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
zakiyullah@telkomuniversity.ac.id

Abstrak

***Unmanned Aerial Vehicle* (UAV) atau yang dikenal sebagai kendaraan tanpa awak berkembang cukup pesat pada saat ini. UAV sendiri memiliki komponen yang paling penting dimana seluruh sistem kontrol pada UAV diatur dalam satu komponen yaitu *flight controller*, dapat dikatakan bahwa komponen tersebut merupakan pusat pengolahan berbagai data milik UAV, karena semua kontrol berpusat di *flight controller*. Hingga saat ini, kebanyakan pengguna UAV dan pembuat UAV menggunakan barang yang sudah jadi, untuk UAV yang sudah jadi seperti contohnya DJI Spark. Pada penelitian ini dibuat *flight controller* sendiri dengan fungsi yang dapat menggunakan mode *Altitude Hold*. *Flight controller* ini menggunakan mikrokontroler STM32 yang dibantu dengan sensor IMU sebagai penentu sikap dan arah wahana dan juga sensor barometer sebagai komponen pengambilan data ketinggian wahana. *Flight controller* yang dibuat akan menggunakan sistem kontrol PID untuk membantu pengontrolan dari UAV. Hasil dari penelitian ini adalah *Flight Controller* yang dibuat memiliki dua mode terbang yaitu mode *stabilize* yang dikendalikan melalui *Remote Control* dan mode AltHold yang membuat UAV mempertahankan ketinggiannya secara *autonomous*. *Flight Controller* berhasil dibuat dan mempunyai nilai PID untuk AltHold dengan parameter Kp = 9.71, Ki = 5.75, dan Kd = 0.851.**

.

Kata Kunci: *Flight Controller*, UAV, *Altitude Hold*, Barometer, IMU, sistem kontrol PID, *AltHold*.

Abstract

Unmanned Aerial Vehicle is developing quite rapidly at this time. The UAV itself has the most important component where the entire control system on the UAV is arranged in flight controller, it can be said that this component is the center for processing various data belonging to the UAV. Until now, most UAV users and UAV makers use ready-made goods, for ready-made UAVs such as the DJI Spark. In this research, the flight controller made with an Altitude Hold mode. Flight controller uses an STM32 microcontroller which is assisted by an IMU sensor as a determinant of the attitude and direction of the vehicle and also a barometer sensor as a component of collecting data on the height of the vehicle. The flight controller made will use the PID control system to help control the UAV. The result of this research is the Flight Controller which is made to have two flight modes, stabilize mode which is controlled via RC and AltHold mode which makes the UAV maintain its altitude autonomously. The flight controller has been successfully created and has a PID value for AltHold with parameters Kp = 9.71, Ki = 5.75, and Kd = 0.851.

***Keywords*: *Flight Controller, UAV, Altitude Hold, Barometer, IMU, PID Control System, AltHold*.**

I. PENDAHULUAN

Dunia teknologi semakin hari semakin berkembang,mendorong manusia untuk terus dapat beradaptasi dan melakukan berbagai macam inovasi. Salah satu teknologi yang sedang gencar dikembangkan saat ini oleh berbagai pihak adalah *Unmanned Aerial Vehicle* (UAV) atau yang dikenal sebagai kendaraan tanpa awak. Memanfaatkan industri UAV di dunia internasional yang berkembang pesat, UAV telah membantu banyak tugas manusia yang diterapkan di banyak bidang seperti keamanan dan pengawasan, pencarian dan penyelamatan, dan sebagai penganalisa di beberapa cabang olahraga [1]. Contoh lain kegunaan UAV yang biasa ditemukan dalam kehidupan sehari-hari adalah seperti pengambilan gambar, pengambilan video dari udara, hingga pemantauan area yang sulit dijangkau oleh manusia.

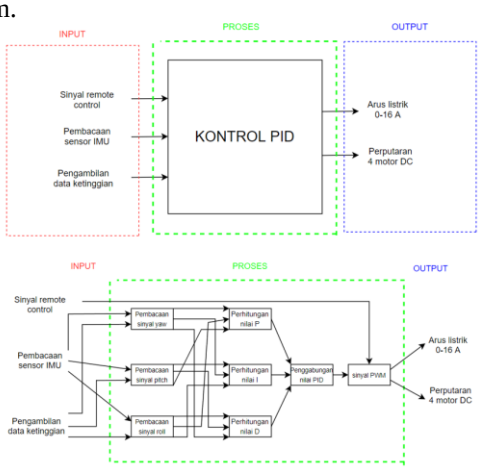
UAV adalah mesin terbang yang dapat bekerja secara *autonomous* maupun dikendalikan secara jarak jauh oleh manusia, menggunakan hukum aerodinamika untuk mengangkat dirinya. Hingga saat ini, perkembangan drone sangat cepat baik dari segi fungsi maupun teknologi [2]. Hal ini dimungkinkan karena UAV terbangun oleh sistem yang sudah terintegrasi dan

dipadukan oleh sensor-sensor yang mendukung UAV untuk dapat melakukan tugasnya secara autonomous. Salah satu komponen penting dalam UAV adalah *flight controller*, komponen ini merupakan pusat dari sistem kontrol, dan diolahnya berbagai macam data untuk mengatur pergerakan dari UAV. Hingga saat ini mayoritas orang menggunakan *flight controller* yang sudah jadi dan membeli dari perusahaan luar negeri seperti Pixhawk, APM, Ardupilot, dll. *Flight controller* yang sudah jadi sangat merugikan, selain membangun perusahaan luar negeri tersebut dengan membeli produknya dampak lainnya adalah ketergantungan dalam menggunakan produk yang sudah jadi dan kurangnya kreatifitas hingga inovasi untuk dapat mengembangkan komponen tersebut. Untuk mengatasi hal ini penulis membuat *flight controller* secara mandiri dengan berbasis pada mikrokontroler jenis STM 32 yang dipadukan dengan sensor barometer untuk dapat mengatur ketinggian dari UAV dan sensor IMU sebagai penentu orientasi UAV. Penulis akan mengembangkan *flight controller* secara mandiri dan *flight controller* yang dibuat dilengkapi dengan mode althold (*altitude hold)*. Mode ini memungkinkan UAV agar dapat mempertahankan letak ketinggian yang sudah diatur, jika dalam kondisi tertentu UAV mengalami terpaan angin maupun penurunan ketinggian secara tidak sengaja maka *flight controller* ini akan mengembalikan posisi UAV kembali ke posisi yang sudah ditentukan.
Pada *flight controller* yang penulis kembangkan, penulis akan menggunakan UAV berjenis *rotary wing*, dengan konfigurasi *quadcopter*. UAV ini berpusat pada gaya angkat yang dikeluarkan oleh *motor brushless* dan *propeller*, UAV ini menggunakan 4 *motor brushless* dan dan 4 *propeller* sebagai daya angkat UAV. Untuk dapat mengatur kecepatan *motor brushless*, UAV ini menggunakan *Electronic Speed Control (ESC)* untuk dapat mengatur arus yang masuk kedalam *motor brushless* guna mengatur kecepatan putaran motor. Sebagai daya, UAV menggunakan baterai berjenis lipo (lithium polymer) dan dilengkapi juga *power module* untuk sebagai pengatur tegangan dan arus yang masuk ke dalam *flight controller*. UAV ini juga dapat dikendalikan menggunakan *transmitter* dengan frekuensi 2.4 GHz, yang akan diterima oleh UAV melalui komponen *receiver*. Pengujian alat yang dilakukan penulis dilakukan pada *outdoor* atau tempat terbuka dengan adanya gangguan faktor angin pada area pengujian.

II. KAJIAN TEORI

A. Desain Konsep Solusi

Pada penelitian ini dirancang sebuah *flight controller* (FC) yang menggunakan dua jenis sensor sebagai masukan sistem dan satu masukan langsung dari pilot melalui sebuah *receiver*, kedua sensor yang digunakan akan mengirimkan tiga data yang dibutuhkan sistem untuk diolah yaitu data *gyroscope*, akselerasi, dan data ketinggian yang dibutuhkan UAV, ketiga data tersebut akan masuk ke dalam mikrokontroler untuk diolah dalam kontrol PID. FC tersebut akan diintegrasikan dengan beberapa komponen penunjang seperti baterai lipo sebagai sumber daya sistem secara keseluruhan, motor brushless DC, ESC, dan juga FTDI untuk memasukan program.



B. *Flight Controller*

Unmanned Aerial Vehicle (UAV) sudah semakin berkembang hingga saat ini karena dapat membantu berbagai macam pekerjaan manusia, mulai dari penyiraman pupuk pada pertanian, pengambilan gambar dan video melalui udara, melakukan pemetaan pada suatu daerah, dll. Untuk dapat melakukan tugasnya tersebut, UAV menggunakan motor DC yang harus tersinkronisasi masing masing RPM motor satu dengan yang lainnya dan untuk melakukan hal tersebut UAV memiliki salah satu komponen terpenting untuk mengatur sistem kontrol dari UAV tersebut yaitu *Flight Controller* (FC).

C. Kontrol PID

Pada kontrol PID, kontrol *proportional* memiliki fungsi untuk menaikan *risetime* menjadi lebih cepat, kontrol *integral* untuk memperkecil *error* yang terjadi, dan kontrol *derivative* berfungsi untuk meredam *overshoot* atau *undershot* pada sistem [3]. Maka didapatkan rumus sebagai berikut:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dT + K_d \frac{de(t)}{dt}$$

D. Ziegler Nicols 1

Dasar pada metode Ziegler Nicols untuk mencari nilai Kp, Ki, dan Kd terdapat pada tabel dibawah:

Tipe Pengontrol	Kp	Ti	Td
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Pada tabel diatas [4] ada beberapa pengontrol dengan beberapa persamaan yang berbeda, karena penulis akan menggunakan pengontrol PID maka niali Kp, Ti, dan Td yang berlaku adalah yang terletak pada kolom PID. Ada beberapa tahapan penting dalam menentukan nilai-nilai dari PID menggunakan Ziegler Nicols 1, yakni:

a. Membuat grafik dari fungsi alih sistem
b. Menarik garis lurus pada saat grafik rise time.
c. Menentukan nilai L dari grafik.

d. Menghitung nilai-nilai yang didapat berdasarkan tabel untuk mendapatkan nilai dari Kp, Ti, dan Td.

Untuk mendapatkan nilai-nilai Kp, Ki, dan Kd. Dapat dicari dengan persamaan PID di bawah:

$$Kp = \frac{1.2\,T}{L}$$
$$Ki = \frac{Kp}{Ti}$$
$$Kd = Kp.Td$$

III. METODE

A. Pemilihan Perangkat Keras *Quadcopter*

Pada penelitian ini, penulis akan membuat *flight controller* yang dapat mempertahankan posisi ketinggiannya pada *quadcopter*, alat ini tersusun dari beberapa komponen, yaitu:

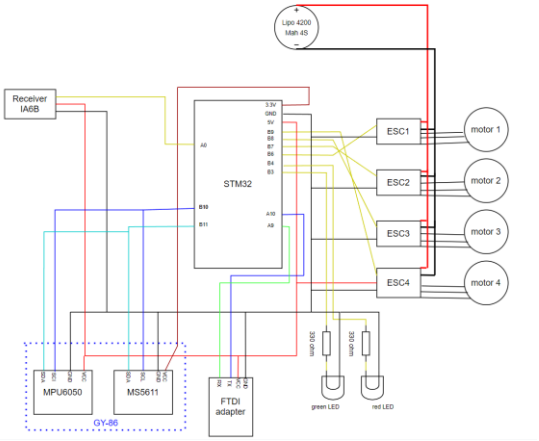
a. DJI F450 *frame*
b. F450 *landing gear*
c. ESC skywalker 30A
d. Motor *brushless* sunnysky 800 kv
e. Lipo baterai 4200Mah 25C
f. Propeller 1045
g. GY-86
h. STM32 F103C8T6
i. Flysky I6
j. Receiver IA6B

a. Kebutuhan *Input* Siste

B. Konstruksi *Hardware*

Dibuat *quadcopter* dengan konfigurasi *X-Copter* yang dapat dilihat pada gambar dibawah:

Pada gambar diatas dapat dilihat, konfigurasi *quadcopter* yang dibuat merupakan jenis *X-copter* dengan diameter antar motornya sejauh 45 cm dan bagian depan dari UAV yang dibuat adalah pada motor 1 dan motor 3. FC diletakan ditengah karena modul GY-86 harus berada pada titik tengah dari UAV, hal ini dikarenakan sensor MPU6050 yang ada pada modul harus berada ditengah-tengah agar dapat membaca pergerakan baik *roll* dan *pitch* dengan baik.

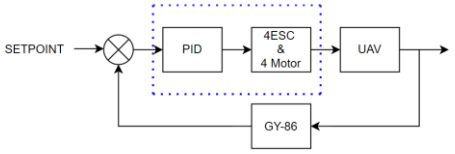


Pada gambar diatas menjelaskan wiring dari sistem keseluruhan yang dibuat, dengan keluaran aktuator yaitu 4 buah ESC dab 4 buah motor DC yang dibagi susun arah putarannya pada tabel dibawah:

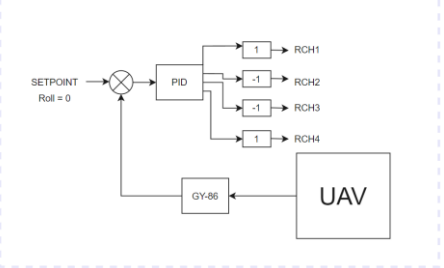
ARAH PERPUTARAN MOTOR	
Motor 1	CCW
Motor 2	CCW
Motor 3	CW
Motor 4	CW
Keterangan : CW = *Clockwise* CCW = *Counter Clockwise*	

C. Diagram Blok

Sistem secara keseluruhan pada pengontrol *quadcopter* yang dibuat oleh penulis dapat dilihat pada diagram blok dibawah:

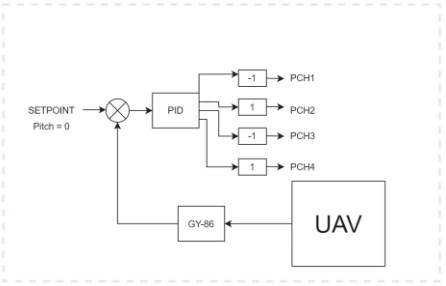


Gambar diatas menunjukan sistem pengontrol secara keseluruhan dimana diawali pada *setpoint* yang telah ditentukan sebelumnya berupa data *roll, yaw* dan *pitch* lalu masuk ke kontrol PID yang akan menerima sinyal *feedback* dari pembacaan modul GY-86 berupa data ketinggian, akselerasi dan juga data kemiringan pada UAV. Hasil pengolahan oleh PID akan diteruskan ke bagian aktuator berupa ESC dan motor yang berupa data PWM yang dapat dibaca oleh ESC yang akan membantu ESC untuk memberikan berapa arus yang dibutuhkan oleh setiap motor agar sistem stabil.

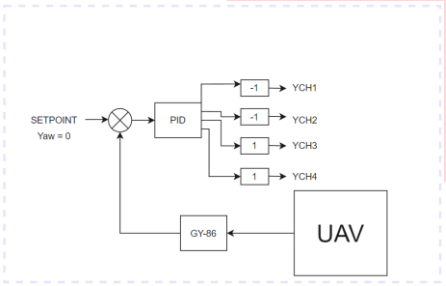


Gambar diatas merupakan diagram blok kontrol PID pada pergerakan *roll*, pada sistem ini PID akan mengolah masukan yang diterima dari sensor MPU6050 dan mengeluarkan empat sinyal yang sudah

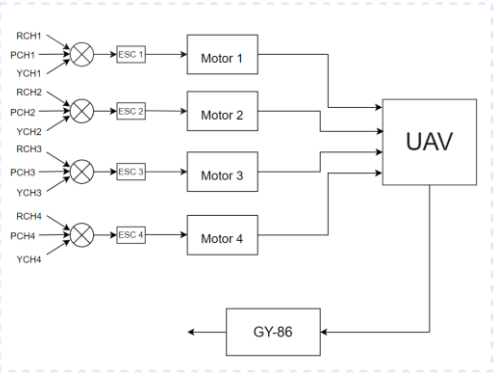
diolah yang akan dikirimkan ke masing-masing ESC, sinyal yang diberikan berbeda beda pada setiap ESC.



Gambar diatas menjelaskan diagram blok dari kontol PID khususnya pada pergerakan *pitch*, kontrol PID akan mengolah data *pitch* pada UAV yang diterima dari MPU6050. Sistem PID akan mengeluarkan empat sinyal yang berbeda untuk diberikan ke masing masing ESC.



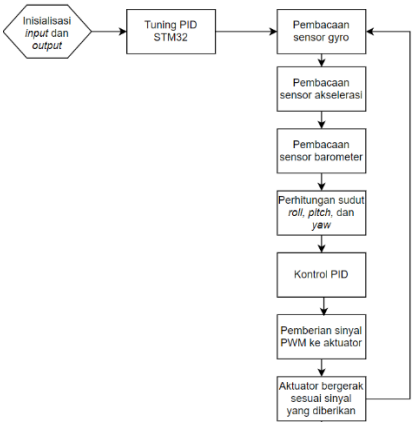
Gambar diatas merupakan diagram blok dari kontrol PID pergerakan *yaw*. Sama seperti pada kontrol PID *roll* dan *pitch*, kontrol PID *yaw* juga memiliki empat keluaran yang berbeda yang akan dikirimkan ke masing masing ESC.



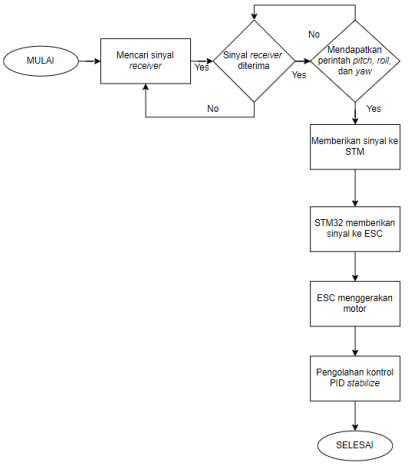
Dapat dilihat pada gambar diatas setelah setiap kontrol PID yaitu *pitch*, *roll*, dan *yaw* menghasilkan keluaran untuk masing masing ESC dan motor maka sinyal keluaran tersebut akan diteruskan ke ESC yang dimana setiap ESC akan menerima tiga sinyal bagian yang terdiri dari sinyal *roll*, *pitch*, dan *yaw* yang akan mengatur motor untuk melakukan hal yang seharusnya dilakukan oleh UAV.

D. Diagram Alir Sistem

Diagram alir sistem dapat dilihat pada gambar berikut:

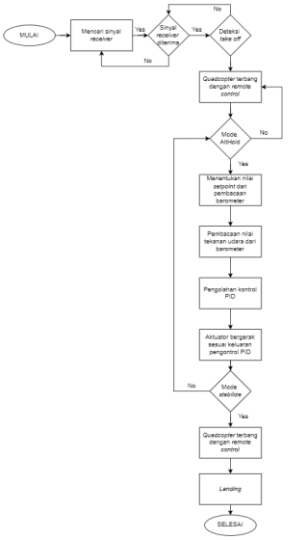


Pada gambar diatas merupakan diagram alir sistem UAV berjenis *quadcopter* dalam menjaga kestabilannya, sistem akan terus melakukan perbaikan pada sistem karena UAV akan terus memberikan *feedback* kepada STM32 yang mengontrol sistem keseluruhan. Bahasa pemrograman yang digunakan merupakan Bahasa C.



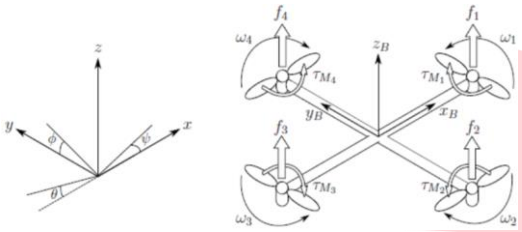
Gambar diatas merupakan diagram alur UAV berjenis *quadcopter* yang digunakan dalam mode manual atau dikendalikan langsung menggunakan *remote control* oleh pilot tanpa ada *feedback* yang diterima oleh sistem.

Adapun mode AltHold yaitu mode disaat *quadcopter* berada dalam posisi menahan ketinggian dengan masukan dari nilai tekanan udara pada pembacaan sensor barometer yg dapay dilihat pada gambar dibawah.



E. Persamaan Gerak Pada *Quadcopter*

Pergerakan dinamika pada *quadcopter* dapat ditampilkan dalam bentuk matematika yang disederhanakan, pergerakan dalam *quadcopter* dapat dimodelkan dengan beberapa variabel, ada 3 sumbu utama pada pergerakan *quadcopter* yaitu sumbu XYZ, dengan sumbu X yang mengarah pada *motor* 1, sumbu Y yang mengarah pada *motor* 4 dan sumbu Z yang mengarah vertikal. Ada pun pemodelan untuk menuntukan kecepatan sudut, torsi motor dan gaya angkat setiap motor yang dapat dilihat pada gambar dibawah:



Persamaan dibawah menunjukan sistem gerak keseluruhan dari *quadcopter* mulai dari pergerakan *roll*, *pitch*, dan *throttle*[5].

$$\dot{x} = \frac{dx}{dt}$$

$$\dot{y} = \frac{dy}{dt}$$

$$\dot{z} = \frac{dz}{dt}$$

$$\ddot{x} = u_1(cos\varphi\ sin\theta\ cos\gamma + sin\varphi\ sin\gamma) - K_1\frac{\dot{x}}{m}$$

$$\ddot{y} = u_1(sin\varphi\ sin\theta\ cos\gamma + cos\varphi\ sin\gamma) - K_2\frac{\dot{y}}{m}$$

$$\ddot{z} = u_1(cos\varphi\ cos\gamma) - g - K_3\frac{\dot{z}}{m}$$

$$\ddot{\theta} = u_2 - IK_4\frac{\dot{\theta}}{I_1}$$

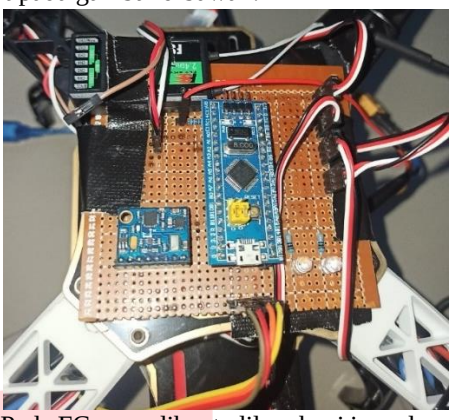
$$\ddot{\gamma} = u_3 - IK_5\frac{\dot{\gamma}}{I_2}$$

$$\ddot{\varphi} = u_4 - IK_6\frac{\dot{\theta}}{I_3}$$

IV. HASIL DAN PEMBAHASAN

A. Bentuk *Flight Controller* Pada *Quadcopter*

Pada *Flight Controller* (FC) secara keseluruhan, penulis menggunakan satu buah STM32 F1 *series* sebagai mikrokontroler yang akan menjadi pusat pengontrol utama dalam quadcopter dan program *looping* pada wahana. Sebagai *frame* utama, penulis menggunakan *frame* DJI F450 yang berdiameter 45 cm antar setiap motornya. FC diletakan di bagian tengah atas yang merupakan titik pusat dari *quadcopter* agar tidak mengganggu keseimbangan dari *quadcopter*. Dalam pembuatan FC, penulis meletakan sensor yang digunakan dalam hal ini yaitu GY-86 pada FC kedalam box tertutup dan hanya dilubangi sedikit dibagian atas, dan penempatan GY-86 lebih tinggi dari posisi *motor*/*propeller*, *Flight Controller* yang dibuat dapat dilihat pada gambar dibawah:



Pada FC yang dibuat, dilengkapi juga dengan dua lampu indikator yang masing masing berwarna hijau dan merah. Lampu indikator iniberfungsi sebagai penanda program yang sedang dijalankan oleh FC untuk memudahkan pengguna dalam menggunakan FC tersebut. Setiap proses yang dilakukan FC dibagi menjadi beberapa *path* LED berkedip, yang dapat dilihat pada tabel dibawah:

LED	Keterangan
● ● ● ● ●	Kalibrasi barometer
●●●●●●●●●	Kalibrasi gyro
●●● ●●●	Receiver mencari sinyal transmitter
●	Mode Stabilize
●●	Mode AltHold
●●●●●●●●●	Kalibrasi level

B. Pengujian *Flight Controller* Pada *Quadcopter*

Pada aplikasi ini dilakukan pengujian *alpha* dan *beta*, pada pengujian *alpha*, metode yang digunakan adalah *blackbox* yang berfokus pada persyaratan fungsional dari perangkat lunak dan *whitebox* yang dilakukan dengan menguji atribut dan *method* yangada di tiap class yang dibangun.

Pada tahap pengujian, penulis melakukan beberapa pengujian terhadap FC dan *quadcopter* yang telah dibuat.Pengujian bertujuan untuk mengecek apakah sistem secara keseluruhan berjalan dengan seharusnya, ada beberapa pengujian yang dilakukan, yaitu:

a. Pengujian sistem keseluruhan FC dengan *quadcopter*.
b. Pengujian sinyal PWM pada *receiver*.
c. Pengujian mode AltHold.
d. Pengujian *throttle* dan ketinggian

a. Pengujian Sistem Secara Keseluruhan Pada *Quadcopter*

Pengujian *alpha whitebox* adalah *testcase* dari atribut dan method pada tiap *class*.Pengujian *alpha blackbox* adalah pengujian yang terfokus pada hasil output dari input yang dimasukkan pada tampilan

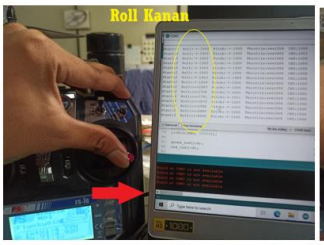
kendali input form tampilan. Ouput harus sesuai dengan input yang ditampilkan.

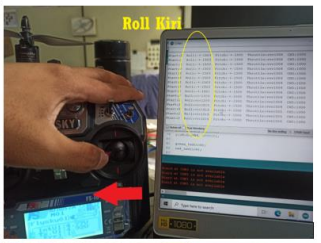
Pengujian kali ini merupakan pengujian sistem secara keseluruhan yang bertujuan untuk mengetahui *quadcopter* dapat melakukan pergerakan sesuai yang seharusnya atau tidak, seperti pergerakan *throttle* atau daya angkat naik turun pada wahana yang dikendalikan melalui *channel* 3 dari *remote control*/*transmitter*, pergerakan *roll* kanan kiri yang dikendalikan melalui *channel 1* pada *transmitter*, dan juga pergerakan *pitch* maju mundur yang dikendalikan melalui *channel 2 transmitter* yang dapat dilihat pada gambar dibawah. Pengujian juga dilakukan apakah *quadcopter* sudah dapat melakukan respon *stabilize* atau penyeimbangan diri saat di udara melalui kontrol PID yang digunakan.

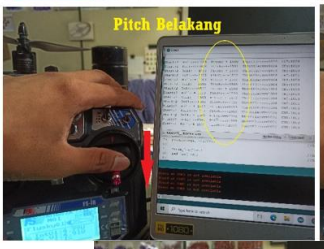
b. Pengujian Sistem Secara Keseluruhan Pada *Quadcopter*

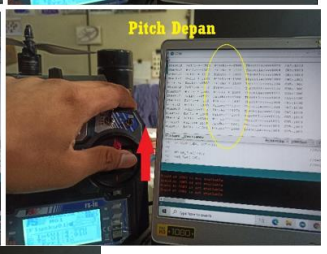
Pengujian sinyal PWM pada receiver dilakukan untuk mengecek apakah *transmitter* berhasil mengirimkan sinyal yang benar pada sistem utama yang diterima melalui *receiver*. Pengujian dilakukan dengan menampilkan data yang diterima oleh *receiver* pada sistem utama dari *transmitter* melalui serial monitor pada Arduino IDE.

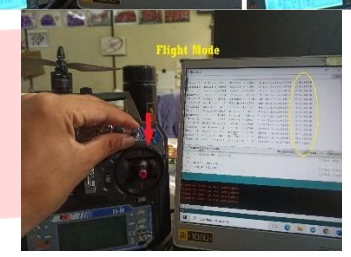
Pada pengujian kali ini dilakukan pengujian terhadap empat sinyal yang digunakan untuk mengendalikan *quadcopter*, yaitu sinyal *throttle, roll, pitch*, dan *yaw* yang dapat dilihat pada gambar dibawah





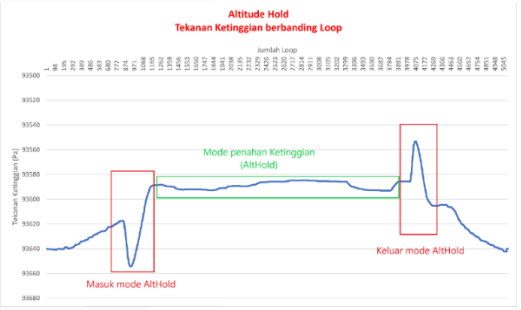






c. Pengujian Mode AltHold

Setelah melakukan pengujian pada sinyal PWM untuk menguji apakah sinyal yang diterima oleh FC sudah sesuai dengan seharusnya, dan telah dilakukan uji terbang sistem secara keseluruhan dengan mode *stabilize* yang dapat dilihat pada *gambar 4.2*, selanjutnya dilakukan pengujian pada mode *AltHold* atau penahan ketinggian dengan jarak dari posisi *take off* ke posisi menahan ketinggian setinggi 5 meter.



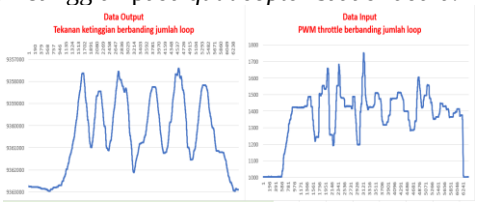
Dapat dilihat pada gambar diatas, Pengujian dilakukan dengan titik mulai atau titik *take off* pada tekanan 93640 Pa dan mengaktifkan mode *AltHold* pada saat tekanan 93586 Pa. Dengan menggunakan rumus untuk merubah tekanan udara kedalam bentuk ketinggian maka didapat nilai seperti pada tabel dibawah:

Proses	Tekanan (Pa)	Ketinggian *Absolute* (m)	Ketinggian *Relative* (m)
Take off	93640	690,10	0
Altitude Hold	93586	695,11	5,01

d. Pengujian *Throttle* Dengan *Ketinggian*

Pengujian *throttle* dan ketinggian dilakukan untuk mendapatkan fungsi alih dari sistem dalam pergerakan naik turun yang dikendalikan oleh *throttle* dan keluaran tekanan udara. Pengujian dilakukan terhadap dua parameter yakni PWM yang dikeluarkan oleh *remote control*/*transmitter* terhadap tekanan

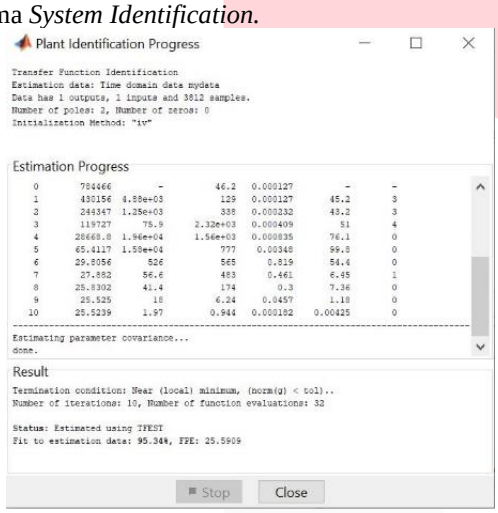
udara ketinggian pada *quadcopter* saat di udara.



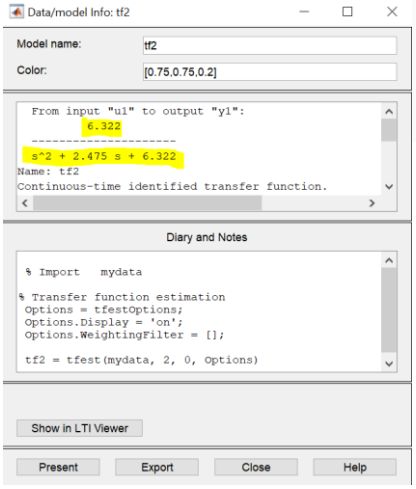
Dapat dilihat pada gambar diatas data masukan dan keluaran tersebut kemudian dipotong di bagian awal *take off* dan bagian *landing* yang kemudian diolah data tersebut untuk mendapatkan fungsi alih yang akan diolah hingga mendapatkan nilai PID dari sistem.

a. Mencari Fungsi Alih Sistem

Data yang telah diambil antara masukan dan keluaran dari sistem kemudian diolah dengan memasukan data masukan dan keluaran tersebut kedalam aplikasi Matlab untuk diolah dan mendapatkan fungsi alih. Untuk mendapatkan fungsi alih maka digunakan *tools* pada aplikasi Matlab yang bernama *System Identification.*



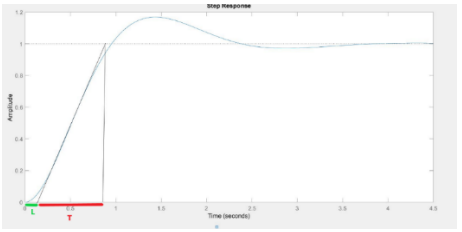
Dapat dilihat pada gambar diatas hasil dari *fitting* antara masukan dan keluaran untuk mendapatkan fungsi alih menunjukan nilai *fit to estimation* sebesar 95.34%. Dengan hasil *fit to estimation* sudah lebih dari 90% maka fungsi alih tersebut dapat digunakan, fungsi alih yang didapatkan dapat dilihat pada gambar dibawah.



Dapat dilihat bahwa fungsi alih sistem tersebut memiliki 2 pole dan 0 zero. Fungsi alih yang sudah didapatkan kemudian akan digunakan dalam mencari nilai PID yang sesuai dengan sistem.

b. Mencari Nilai PID

Dalam mencari nilai Kp, Ki, dan Kd sistem digunakan metode Ziegler nicols 1 atau kurva reaksi, *transfer function* yang didapat di *plot* dalam bentuk grafik untuk dicari nilai L dan dan T nya yang dapat dilihat pada gambar dibawah:



Maka dapat dilihat pada gambar diatas menambahkan dua garis untuk membantu mencari nilai L dan T dari sistem, dan berhasil didapatkan nilai L sebesar 0.296 dan nilai T sebesar 1.418.

Jika nilai T dan L telah didapatkan maka dapat dicari nilai dari K_p, T_i, T_d dengan menggunakan persamaan Ziegler Nicols yang terdapat pada *tabel 2.2.*

$$K_p = 1.2\frac{T}{L} = 5.75$$
$$T_i = 2L = 0.592$$
$$T_d = \frac{L}{2} = 0.148$$

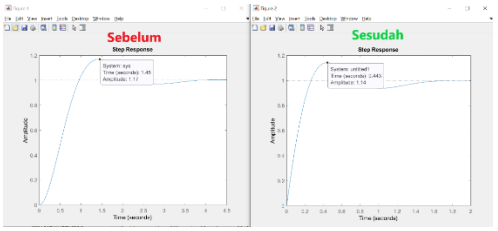
Setelah nilai-nilai dari K_p, T_i, T_d berhasil didapatkan, maka dengan persamaan standar kontrol PID, dapat dicari nilai-nilai dari K_p, K_i, K_d dengan menggunakan persamaan dibawah:

$$K_p + \frac{K_i}{s} + \frac{K_d}{s} = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

Maka,

$$K_p = 5.75$$
$$K_i = \frac{K_p}{T_i} = 9.71$$
$$K_d = K_p T_d = 0.851$$

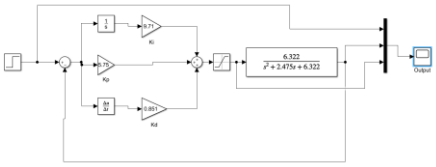
Jika telah mendapatkan nilai-nilai dari K_p, K_i, K_d maka nilai tersebut akan dimasukan kedalam sistem melalui matlab untuk mendapatkan perbedaan grafik perbedaan sebelum memiliki nilai dan *feedback* PID dan juga setelah dimasukan nilai PID ke sistem, grafik perbedaan dapat dilihat pada gambar dibawah:



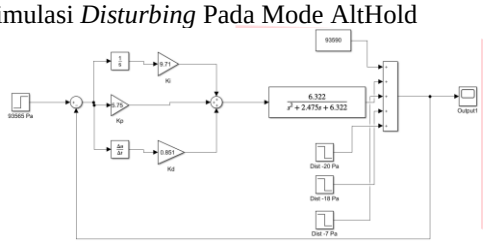
Dapat dilihat pada gambar diatas perbedaan sistem sebelum dan sesudah dimasukan PID hasil *tuning* dari metode Ziegler Nicols 1 memiliki beberapa perbedaan, pertama dalam nilai *overshoot*, sebelum dimasukan PID Ziegler Nicols, sistem memiliki overshoot sebesar 17% dan setelah

dimasukan PID menjadi 14%, dan perbedaan kedua ada pada waktu *settling time* sebelumnya adalah mendekati 4 detik, dan setelah dimasukan PID Ziegler Nicols hanya menjadi 1.8 detik.

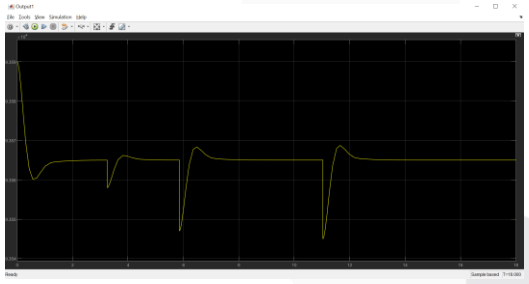
Penulis juga melakukan pengetesan sistem melalui Simulink pada aplikasi matlab, untuk menguji masukan sinyal step yang melalui sistem kontrol PID yang telah dicari dan melewati fungsi alih yang telah dicari seperti pada gambar dibawah.



e. Simulasi *Disturbing* Pada Mode AltHold



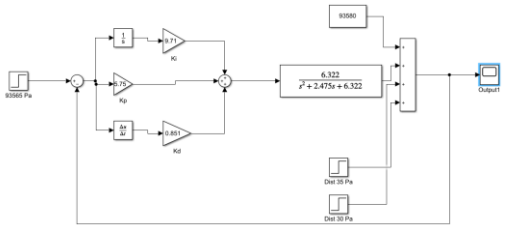
Gambar diatas menunjukan sistem yang di desain oleh penulis menggunakan Simulink pada aplikasi Matlab, dan dapat dilihat penulis melakukan simulasi pertama adalah dimulasi dorongan ke atas dengan menambahkan sinyal *disturbance* pada keluaran, penulis menggunakan stop time sebesar 18 detik, dan ada 3 nilai *disturbance* yang penulis gunakan yakni -7 Pa pada detik ke 3.252, yang kedua -18 Pa dengan mulai *disturbance* pada detik 5.856, dan yang terakhir disturbance sebesar -20 Pa pada detik ke 11.036.



Pada gambar diatas menunjukan keluaran sistem saat dilakukan simulasi *disturbance*, penulis mengambil data waktu disaat awal sistem mengalami *disturbance* hingga dapat kembali ke posisi semula yang dapat dilihat pada tabel dibawah.

Nilai *Disturbance* (Pa)	Detik Mulai	Detik Kembali Stabil	*Range* Waktu (detik)
-7	3.252	4.392	1.14
-18	5.856	7.376	1.52
-20	11.036	12.686	1.65

Selain melakukan simulasi dorong ke atas penulis juga melakukan simulasi mode AltHold dengan *disturbance* berupa tarikan ke arah bawah, pada simulasi ini penulis menggunakan *disturbance* sebanyak dua kali ke arah bawah, dengan besaran nilai *disturbance* pertama yakni sebesar 30 Pa dan yang kedua sebesar 35 Pa. Rangkaian sistem simulasi dapat dilihat pada gambar dibawah.



Dapat dilihat pada gambar diatas, *disturbance* dilakukan sebanyak dua kali ke arah bawah dengan nilai setpoint sebesar 93565 Pa, Adapun grafik keluaran dari sistem dapat dilihat pada gambar dibawah.

Pada gambar diatas dapat dilihat keluaran dari sistem saat dilakukan *disturbance* sebanyak dua kali. Dan dari grafik tersebut penulis menggunakan stop time 14 detik dan *disturbance* 30 Pa dimulai pada detik 3.252 dan *disturbance* 35 Pa dimulai saat detik ke 6.1. Penulis juga menghitung rentang waktu yang dibutuhkan sistem untuk kembali ke nilai *setpoint* yang dapat dilihat pada tabel dibawah.

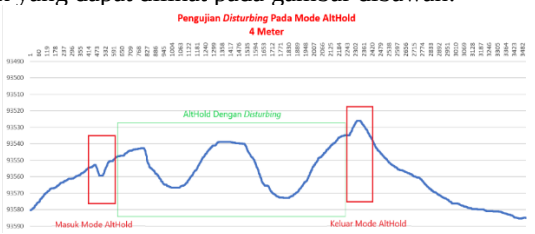
Nilai *Disturbance* (Pa)	Detik Mulai	Detik Kembali Stabil	*Range* Waktu (detik)
30	3.252	4.822	1.57
35	6.1	7.59	1.49

f. Pengujian Disturbing Pada Mode AltHold

Pengujian pertama dilakukan pada sore hari dengan suhu 25 derajat celsius dan dengan tekanan udara saat *take off* 93580 Pa dan tekanan udara saat melakukan mode AltHold adalah 93537 Pa, dimana jika dikonversi dalam meter menggunakan rumus pada *referensi* [6] maka didapat ketinggian seperti pada tabel dibawah:

Proses	Tekanan (Pa)	Ketinggian *Absolute* (m)	Ketinggian *Relative* (m)
Take off	93580	688.7	0
Altitude Hold	93537	692.69	3.99

Pada pengujian pertama, penulis melakukan penarikan ke arah bawah sebanyak 2 kali untuk dapat melihat apakah *quadcopter* dapat kembali ke posisi awal atau tidak, dan pengujian disajikan dalam bentuk grafik yang dapat dilihat pada gambar dibawah:

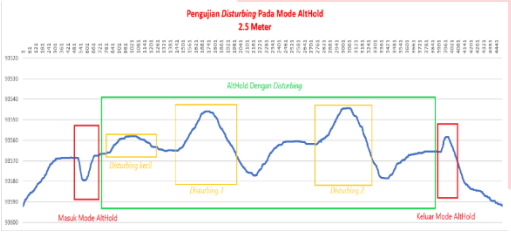


Pada pengujian pertama dapat dilihat bahwa *quadcopter* berhasil kembali kepada posisi semula dengan terkanan udara mendekati tekanan saat AltHold yaitu 93537 Pa.

Untuk pengujian kedua, penulis melakukan pengujian dengan cara mendorong ke atas *quadcopter* saat berada pada mode AltHold, pengujian dilakukan dengan suhu 25 derajat celsius dan dengan tekanan udara saat *take off* adalah 93592 Pa dan tekanan udara saat mode AltHold adalah 93565 Pa, dengan mengunakan rumus pada *referensi* [6] maka didapat seperti pada tabel dibawah:

Proses	Tekanan (Pa)	Ketinggian *Absolute* (m)	Ketinggian *Relative* (m)
Take off	93592	687.63	0
Altitude Hold	93565	690.11	2.48

Pengujian dilakukan pada ketinggian 2.5 meter dengan didorong ke arah atas sebanyak dua kali dan 1 kali dorongan kecil, hal ini bertujuan untuk melihat respon *quadcopter* yang kembali ke posisi awal mode AltHold, hasil grafik pengujian dapat dilihat pada gambar dibawah.

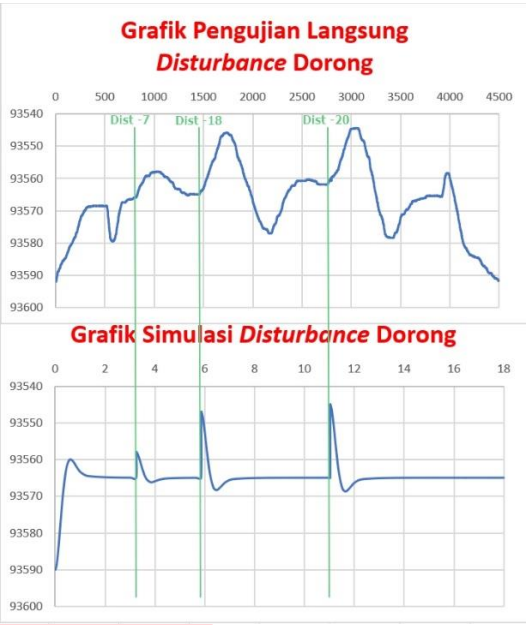


g. Komparasi Simulasi Dengan Pengujian Langsung

Setelah melakukan Pengujian secara nyata dan simulasi melalui Simulink pada aplikasi Matlab, maka didapat dua data dan dua grafik yang menyatakan respon sistem yang telah dibuat. Pertama, penulis melakukan komparasi waktu kembali sistem stabil dalam satuan detik untuk pengujian dorong ke atas.

Nilai *Disturbance* (Pa)	SIMULASI			PENGUJIAN			
	Detik Awal	Detik Kembali	*Range* Waktu (detik)	*Loop* Awal	*Loop* Kembali	*Range Loop*	*Range* Waktu (detik)
-7	3.252	4.392	1.14	817	1403	586	2.344
-18	5.856	7.376	1.52	1457	2434	977	3.908
-20	11.036	12.686	1.65	2759	3754	995	3.98

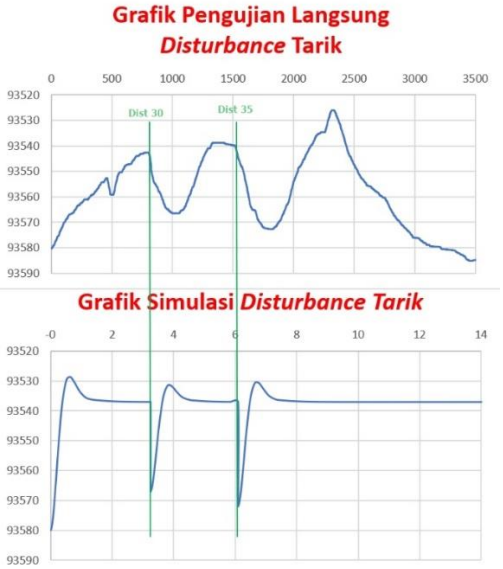
Pada tabel diatas menunjukan selisih waktu yang dibutuhkan sistem untuk kembali ke posisi *setpoint* dalam pengujian *disturbing* didorong ke arah atas. Pada pengujian *disturbing* -7 Pa, respon pengujian langsung lebih lambat untuk mencapai posisi kembali ke nilai *setpoint* dengan perbedaan 1.204 detik dibandingkan dengan simulasi. Pada *disturbance* sebesar -18 Pa dapat dilihat pengujian langsung respon lebih lambat sebesar 2.388 detik dibandingkan simulasi, dan pada *disturbance* -20 Pa respon sistem saat pengujian langsung lebih lambat 2.33 detik disbanding simulasi. Penulis juga menyajikan komparasi antara simulasi dengan pengujian diatas dalam bentuk grafik yang dapat dilihat pada gambar dibawah.



Penulis juga melakukan komparasi untuk pengujian *quadcopter* ditarik ke bawah.

Nilai *Disturbance* (Pa)	SIMULASI			PENGUJIAN			
	Detik Awal	Detik Kembali	*Range* Waktu (detik)	*Loop* Awal	*Loop* Kembali	*Range Loop*	*Range* Waktu (detik)
30	3.252	4.822	1.57	790	1346	556	2.224
35	6.1	7.59	1.49	1522	2246	724	2.896

Pada tabel diatas menunjukan komparasi antara simulasi dengan pengujian langsung pada kondisi *disturbance* berupa tarikan ke arah bawah. Pada *disturbance* pertama sebesar 30 Pa antara simulasi dengan pengujian nyata memiliki selisih 0.654 detik, dan pada *disturbance* 35 Pa memiliki perbedaan waktu kembali stabil sebesar 1.406 detik. Adapun komparasi simulasi dengan pengujian dalam grafik dapat dilihat pada gambar dibawah.



V. KESIMPULAN

A. Kesimpulan

Mengacu pada hasil dan pengujian-pengujian yang dilakukan oleh penulis terhadap *flight controller* (FC) yang telah dibuat oleh penulis menggunakan mikrokontroller STM32 dan dengan modul GY-86 yang

terdapat sensor MPU6050 dan barometer MS5611, penulis menyimpulkan:

a. *Quadcopter* yang telah dibuat dapat melakukan pergerakan *roll* atau kanan kiri, *pitch* atau maju mundur, dan *throttle* atau naik dan turun dengan benar dengan pengendali berupa *Remote Control* (RC).
b. *Flight controller* (FC) yang dibuat oleh penulis dapat melakukan mode *Altitude Hold* atau menahan ketinggian pada ketinggian 5 meter diatas tanah.
c. *Flight Controller* (FC) yang dibuat oleh penulis memiliki kecepatan *looping* sebesar 4mS atau 4000μS. Dengan pengambilan data barometer setiap 3 kali *looping* program utama, karena pengambilan MS5611 membutuhkan waktu 9ms atau 9000μS.
d. Nilai PID terbang AltHold yang digunakan untuk pergerakan menahan ketinggian adalah K_p = 5.75, K_i = 9.71, dan K_d = 0.851 dengan nilai *overshoot* 14% dan *settling time* sebesar 1.8 detik.
e. *Flight Controller* (FC) memiliki indikator berupa LED hijau dan LED merah yang menunjukan program yang berjalan pada FC.

B. Saran

Adapun saran-saran yang diajukan oleh penulis sebagai bentuk pengembangan atas *Flight Controller* (FC) yang telah dibuat oleh penulis adalah sebagai berikut:

a. Mengembangkan mode tambahan pada FC seperti *GPS lock* dan *heading lock*.
b. Pembuatan GUI (*Graphical User Interface*) untuk memantau data penting saat *quadcopter* berada di udara berbasis telemetri.

REFERENSI

[1] D. Du *et al.*, “The unmanned aerial vehicle benchmark: Object detection and tracking,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11214 LNCS, pp. 375–391, 2018, doi: 10.1007/978-3-030-01249-6_23.
[2] R. F. Rahman, “IMPLEMENTASI AUGMENTED REALITY PADA PENGENALAN UNMANNED AERIAL VEHICLE DRONE BERBASIS ANDROID,” 2016.
[3] N. Yuliarmas, S. Aisyah, and H. Toar, “Implementasi Kontrol PID pada Mesin Pengembang Roti,” *J. Rekayasa Elektr.*, vol. 11, no. 3, 2015, doi: 10.17529/jre.v11i3.2304
[4] N. Allu and S. Salu, “Aplikasi Penalaan Dengan Metode Ziegker Nichols di Perancangan Motor DC,” *Pros. Semin. Nas. 2018*, vol. 1, no. April, pp. 71–77, 2018.
[5] U.A. Fiaz and A. Mukarram, “Altitude Control of a Quadcopter Altitude Control of a Quadcopter By Department of Electrical Engineering Pakistan Institute of Engineering & Applied Sciences,” no. June 2015, 2018
[6] B. P. Sensor, “MS5611-01BA03,” vol. m, 2015.