

**SISTEM KENDALI POSISI VTOL (*VERTICAL TAKE OFF
LANDING*) DENGAN METODE ANFIS MENGGUNAKAN
INTERFACE SIMULINK**

SKRIPSI

Diajukan sebagai salah satu syarat
untuk memperoleh gelar
Sarjana Teknik



Oleh :

VERNANDA SITORINI ZUL HIZMI

NIM. I0716032

**PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS SEBELAS MARET
SURAKARTA
2020**



SURAT TUGAS

Nomor : 040/TA/TE/2020

Kepala Program Studi Teknik Elektro Fakultas Teknik Universitas Sebelas Maret memberikan tugas kepada:

| | |
|-----------------------|---|
| Nama Mahasiswa | : Vernanda Sitorini Zul Hizmi |
| NIM | : I0716032 |
| Bidang peminatan | : Sistem Mekatronika (SM) |
| Pembimbing Utama | : Hari Maghfiroh M.Eng. NIP. 199104132018031001 |
| Pembimbing Pendamping | : Chico Hermanu BA, S.T., M.Eng. NIP. 198804162015041002 |
| Mata kuliah pendukung | : 1. Sistem Otomotif 2. HMI 3. Teknik Kendali Digital |

untuk mengerjakan dan menyelesaikan Tugas Akhir dengan judul :

Sistem Kendali Posisi VTOL (Vertical Take Off Landing) dengan Metode ANFIS Menggunakan Interface Simulink

Surat tugas ini dibuat untuk dilaksanakan dengan sebaik-baiknya.

Surakarta, 27 Januari 2020
Kepala Program Studi

Feri Adriyanto, Ph.D.
NIP. 196801161999031001

Tembusan:

1. Mahasiswa ybs.
2. Dosen Pembimbing TA
3. Koordinator TA
4. Arsip

SURAT PERNYATAAN INTEGRITAS PENULIS

Saya mahasiswa Program Studi Sarjana Teknik Elektro Universitas Sebelas Maret yang bertanda tangan di bawah ini:

Nama : Vernanda Sitorini Zul Hizmi
NIM : I0716032
Judul tugas akhir : Sistem Kendali Posisi VTOL (*Vertical Take Off Landing*) dengan Metode ANFIS Menggunakan *Interface Simulink*

Dengan ini menyatakan bahwa tugas akhir yang saya susun tidak mencontoh atau melakukan plagiat dari karya tulis orang lain. Jika terbukti tugas akhir yang saya susun tersebut merupakan hasil plagiat dari karya orang lain maka tugas akhir yang saya susun tersebut dinyatakan batal dan gelar sarjana yang saya peroleh dengan sendirinya dibatalkan atau dicabut.

Demikian surat pernyataan ini saya buat dengan sebenarnya dan apabila dikemudian hari terbukti melakukan kebohongan maka saya sanggup menanggung segala konsekuensinya.

Surakarta, 5 Juli 2020



Vernanda Sitorini Zul Hizmi

NIM. I0716032

**HALAMAN PENGESAHAN TIM PEMBIMBING DAN TIM PENGUJI
SISTEM KENDALI POSISI VTOL (VERTICAL TAKE OFF LANDING)
DENGAN METODE ANFIS MENGGUNAKAN INTERFACE SIMULINK**

Disusun Oleh

VERNANDA SITORINI ZUL HIZMI

NIM I0716032

Pembimbing 1



Hari Maghfiroh M.Eng.
NIP 199104132018031001

Pembimbing 2



Chico Hermanu BA, S.T., M.Eng.
NIP 198804162015041002

Telah dipertahankan di hadapan Tim Dosen Penguji pada hari Senin tanggal 20 Juli 2020

1. **Hari Maghfiroh M.Eng.**
NIP. 198705062019031009
2. **Chico Hermanu BA, S.T., M.Eng.**
NIP. 198705062019031009
3. **Joko Slamet Saputro, S.Pd., M.T.**
NIP. 198705062019031009
4. **Feri Adriyanto, Ph.D.**
NIP. 198705062019031009



.....



.....



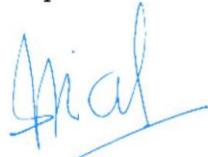
.....



.....

Mengetahui,

Kepala Prodi Teknik Elektro



Feri Adriyanto, Ph.D.
NIP. 196801161999031001

Koordinator Tugas Akhir



Muhammad Hamka I, S.T., M.Eng.
NIP. 198812292019031011

SISTEM KENDALI POSISI VTOL (*VERTICAL TAKE OFF LANDING*) DENGAN METODE ANFIS MENGGUNAKAN *INTERFACE* SIMULINK

Vernanda Sitorini Zul Hizmi

Program Studi Teknik Elektro Fakultas Teknik, Universitas Sebelas Maret

Email : vernandaszh@student.uns.ac.id

Motor listrik telah banyak diaplikasikan dalam berbagai peralatan. Salah satu pengaplikasiannya terdapat pada pesawat tanpa awak/ *Unmanned Aerial Vehicles* (UAV). Sistem pengendalian kecepatan motor listrik yang mampu menyeimbangkan posisi pesawat adalah salah satu fitur wajib yang harus dimiliki pesawat. Kendali penyeimbang posisi juga mendukung sistem *Vertical Take Off Landing* (VTOL). Sistem kendali posisi sederhana VTOL dapat dirancang dengan *hardware* neraca keseimbangan serta komponen sistem: Arduino Uno sebagai *controller*, MPU6050 *sensor module* sebagai sensor posisi, ESC sebagai *driver motor*, BLDC A2212 sebagai *actuator*, dan baterai Li-Po 3S sebagai *supply*. Sistem kendali posisi VTOL pada penelitian ini menggunakan *Hardware-in-the-loop* (HIL) *simulation* dan dapat diuji atau divariasikan dengan *interface* Simulink. Pengujian kinerja sistem kendali posisi dengan metode PID, FLC, dan ANFIS divariasikan dengan posisi awal sudut -15° , 0° , dan 15° serta pembebanan dengan *load* 27,5 gram, 33 gram, dan 38,5 gram. Hasil terbaik pengujian ditunjukkan pada posisi awal 0° dan pembebanan dengan *load* 33 gram menggunakan metode kendali ANFIS.

Keywords: Kendali Posisi VTOL, ANFIS, Simulink.

***POSITION CONTROL SYSTEM VTOL (VERTICAL TAKE OFF LANDING)
USING ANFIS METHOD WITH SIMULINK INTERFACE***

Vernanda Sitorini Zul Hizmi

Program Studi Teknik Elektro Fakultas Teknik, Universitas Sebelas Maret

Email : vernandaszh@student.uns.ac.id

Electric motors have been widely applied in various equipment. One application is found in unmanned aerial vehicles (UAVs). An electric motor speed control system that can balance the position of the aircraft is one of the mandatory features that must be owned by the aircraft. The position balancer control also supports the Vertical Take Off Landing (VTOL) system. The simple VTOL position control system can be designed with a balance hardware and system components: Arduino Uno as a controller, MPU6050 sensor module as a position sensor, ESC as a motor driver, BLDC A2212 as an actuator, and Li-Po 3S battery as a supply. The VTOL position control system in this study uses Hardware-in-the-loop (HIL) simulation and can be tested or varied with the Simulink interface. Position control system performance testing using PID, FLC, and ANFIS methods varied with the initial position angles of -15° , 0° , and 15° and loading with loads of 27.5 grams, 33 grams and 38.5 grams. The best test results are shown at the initial position of 0° and loading with a load of 33 grams using the ANFIS control method.

Keywords: VTOL Position Control, ANFIS, Simulink.

KATA PENGANTAR

Puji syukur kehadiran Allah *Subhanahu Wa Ta'ala* yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul “Sistem Kendali Posisi VTOL (*Vertical Take Off Landing*) dengan Metode ANFIS Menggunakan *Interface Simulink*”. Skripsi ini disusun untuk memenuhi salah satu syarat dalam memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro Universitas Sebelas Maret Surakarta.

Penulis tidak akan sampai pada tahap ini dan skripsi ini tidak akan pernah selesai jika penulis tidak mendapat bantuan dan dukungan dari berbagai pihak. Penulis hendak menyampaikan terima kasih kepada:

1. Bapak Hari Maghfiroh, S.T., M.Eng selaku Dosen Pembimbing I dan Bapak Chico Hermanu Brillianto Apribowo, S.T., M.Eng selaku Dosen Pembimbing II yang selalu memberikan dukungan, ide, arahan, bimbingan, dan motivasi selama perkuliahan di Teknik Elektro sampai pengerjaan skripsi ini.
2. Bapak Joko Slamet Saputro, S.Pd., M.T. selaku Dosen Penguji I dan Bapak Feri Adriyanto, Ph.D selaku Dosen Penguji II serta Kaprodi Teknik Elektro UNS yang telah memberikan ide, arahan, motivasi dalam pengerjaan skripsi ini.
3. Bapak Irwan Iftadi, S.T., M.Eng., selaku Dosen Pembimbing Akademik yang selalu memberikan pemahaman yang baik mengenai perkuliahan.
4. Seluruh Dosen Program Studi Teknik Elektro yang telah memberikan ilmu yang bermanfaat, motivasi, dan inspirasi selama masa perkuliahan penulis.
5. Teman-teman Teknik Elektro angkatan 2016 yang tidak bisa penulis sebutkan satu persatu, yang telah berjuang bersama selama 4 tahun ini. Terhusus untuk “Meka Joss” dan “Assalamu’alaikum Ukhti” yang sering menemani ke-*random*-an penulis.
6. Kakak-kakak angkatan 2014 dan 2015 yang selalu memberi contoh, arahan, dukungan, *tips and trick* selama penulis berkuliah di Teknik Elektro. Serta adik-adik 2017, 2018 dan 2019 yang selalu memberikan dukungan dan bertanya tentang tugas atau soal ujian semester lalu.
7. Keluarga tercinta, bapak Warsito yang selalu memberikan motivasi dan semangat serta pertanyaan “kapan lulus?”, ibu Isrini Suprapti yang diam-diam

selalu punya rencana untuk penulis, serta adik Brilian Rafi Prakoso yang bisa penulis suruh-suruh.

8. dan semua pihak yang telah memberikan bantuan, yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih belum sempurna. Penulis juga memohon maaf apabila terdapat kesalahan dalam penulisan skripsi ini.

Karanganyar, 7 Juli 2020
Vernanda Sitorini Zul Hizmi

DAFTAR ISI

| | |
|--|------|
| HALAMAN JUDUL..... | i |
| HALAMAN SURAT PENUGASAN | ii |
| SURAT PERNYATAAN INTEGRITAS PENULIS..... | iii |
| HALAMAN PENGESAHAN..... | iv |
| ABSTRAK | v |
| <i>ABSTRACT</i> | vi |
| KATA PENGANTAR | vii |
| DAFTAR ISI..... | ix |
| DAFTAR GAMBAR | xi |
| DAFTAR TABEL..... | xiii |
| BAB I PENDAHULUAN | 14 |
| 1.1 Latar Belakang | 14 |
| 1.2 Rumusan Masalah | 2 |
| 1.3 Tujuan Penelitian..... | 3 |
| 1.4 Manfaat Penelitian..... | 3 |
| 1.5 Sistematika Penulisan..... | 3 |
| BAB II TINJAUAN PUSTAKA..... | 4 |
| 2.1 Penelitian Sebelumnya | 4 |
| 2.2 <i>VTOL System</i> | 4 |
| 2.2.1 <i>Controller</i> | 7 |
| 2.2.2 <i>Position Sensor</i> | 10 |
| 2.2.3 <i>Electrictronic Speed Control (ESC)</i> | 13 |
| 2.2.4 <i>BLDC Motor</i> | 14 |
| 2.2.5 <i>Li-Po 3S Battery</i> | 16 |
| 2.3 Sistem Kendali | 17 |
| 2.3.1 <i>Proportional Integral Derivative</i> | 18 |
| 2.3.2 <i>Fuzzy Logic Controller</i> | 19 |
| 2.3.3 <i>Adaptive Network-based Fuzzy Inference System</i> | 20 |
| 2.4 Simulink | 21 |

| | |
|---|----|
| BAB III METODOLOGI PENELITIAN..... | 23 |
| 3.1 Alur Penelitian..... | 23 |
| 3.2 Alat dan Bahan | 24 |
| 3.3 Perancangan dan Pembuatan <i>Hardware</i> | 25 |
| 3.4 Perancangan dan Pembuatan <i>Software</i> | 26 |
| 3.4.1 Perancangan Algoritma Pembacaan Sensor | 27 |
| 3.4.2 Perancangan Algoritma Pengaktifan Motor BLDC | 28 |
| 3.4.3 Perancangan Sistem Kendali | 28 |
| 3.4.4 Perancangan Integrasi Simulink dengan <i>Controller</i> | 39 |
| BAB IV HASIL DAN PEMBAHASAN | 40 |
| 4.1 Hasil Pengujian Pengaruh Posisi Awal | 42 |
| 4.1.1 Posisi awal -15° | 42 |
| 4.1.2 Posisi awal 0° | 44 |
| 4.1.3 Posisi awal 15° | 46 |
| 4.2 Hasil Pengujian Pengaruh Pembebanan | 48 |
| 4.2.1 <i>Load</i> 27,5 gram..... | 49 |
| 4.2.2 <i>Load</i> 33 gram..... | 51 |
| 4.2.3 <i>Load</i> 38,5 gram..... | 53 |
| 4.3 Perbandingan Kinerja <i>Controller</i> | 55 |
| BAB V PENUTUP..... | 56 |
| 5.1 Kesimpulan..... | 56 |
| 5.2 Saran | 56 |
| DAFTAR PUSTAKA | 57 |
| LAMPIRAN..... | 60 |

DAFTAR GAMBAR

| | |
|--|----|
| Gambar 2.1 <i>PinOut</i> Arduino Uno [13] | 8 |
| Gambar 2.2 PWM [16]..... | 9 |
| Gambar 2.3 Orientasi Putaran MPU6050 [17]..... | 10 |
| Gambar 2.4 (a) Kondisi Normal, (b) Kemiringan pada Sumbu Z dan X, (c) Kemiringan pada Sumbu Z dan Y, (d) Kemiringan pada Sumbu X, Y, Z [17] | 11 |
| Gambar 2.5 MPU6050 <i>Sensor Module</i> [17] | 12 |
| Gambar 2.6 Komponen ESC [18] | 13 |
| Gambar 2.7 (a) Motor BLDC A2212/6T 2200KV dan (b) Pin Internal [19]..... | 14 |
| Gambar 2.8 Sirkuit <i>Switching</i> Pengaktifan Motor BLDC [19]..... | 16 |
| Gambar 2.9 Baterai Li-Po 3S [20] | 16 |
| Gambar 2.10 Sistem Kendali Lup Tertutup [21] | 17 |
| Gambar 2.11 Grafik Respons Sistem [22] | 18 |
| Gambar 2.12 Blok Diagram <i>PID Controller</i> [23] | 19 |
| Gambar 2.13 Blok Diagram <i>Fuzzy Logic Control</i> [2] | 20 |
| Gambar 2.14 Arsitektur ANFIS [23] | 21 |
| Gambar 3.1 Diagram Alir Penelitian | 24 |
| Gambar 3.2 <i>Wiring</i> Komponen VTOL | 25 |
| Gambar 3.3 Hasil Pembuatan <i>Hardware</i> | 26 |
| Gambar 3.4 Diagram Alir Algoritma Pembacaan Sensor MPU6050 | 27 |
| Gambar 3.5 Diagram Alir Algoritma Pengaktifan Motor BLDC | 28 |
| Gambar 3.6 <i>System Identification Toolbox</i> | 29 |
| Gambar 3.7 <i>Response Transfer Function</i> dan <i>Response Hardware</i> | 30 |
| Gambar 3.8 Sistem <i>PID Controller</i> | 30 |
| Gambar 3.9 Grafik Hasil Pengendalian Sistem dengan <i>PID Controller</i> | 31 |
| Gambar 3. 10 <i>Fuzzy Logic Designer Toolbox</i> | 32 |
| Gambar 3.11 Fungsi Keanggotaan <i>Input</i> dan <i>Output</i> | 33 |
| Gambar 3.12 <i>Rule Editor</i> | 34 |
| Gambar 3.13 Hasil <i>Training</i> dalam <i>Neuro-Fuzzy Designer Toolbox</i> | 35 |
| Gambar 3.14 <i>ANFIS Model Structure</i> | 36 |
| Gambar 3.15 Fungsi Keanggotaan <i>Input1</i> dan 2 serta <i>Output</i> ANFIS | 37 |

| | |
|---|----|
| Gambar 3.16 <i>Rule Editor</i> ANFIS..... | 38 |
| Gambar 3.17 Diagram Alir Integrasi Simulink dengan Arduino | 39 |
| Gambar 4.1 <i>Interface</i> pada Simulink | 40 |
| Gambar 4.2 Hasil Pembuatan Sistem Kendali Posisi VTOL dengan <i>Interface</i> Simulink..... | 41 |
| Gambar 4.3 Hasil Pengujian Posisi Awal -15° dengan PID <i>Controller</i> | 42 |
| Gambar 4.4 Hasil Pengujian Posisi Awal -15° dengan FLC | 43 |
| Gambar 4.5 Hasil Pengujian Posisi Awal -15° dengan ANFIS | 43 |
| Gambar 4.6 Hasil Pengujian Posisi Awal 0° dengan PID..... | 44 |
| Gambar 4.7 Hasil Pengujian Posisi Awal 0° dengan FLC..... | 45 |
| Gambar 4.8 Hasil Pengujian Posisi Awal 0° dengan ANFIS | 45 |
| Gambar 4.9 Hasil Pengujian Posisi Awal 15° dengan PID <i>Controller</i> | 46 |
| Gambar 4.10 Hasil Pengujian Posisi Awal 15° dengan FLC..... | 47 |
| Gambar 4.11 Hasil Pengujian Posisi Awal 15° dengan ANFIS | 47 |
| Gambar 4.12 Hasil Pengujian <i>Load</i> 27,5 gram dengan PID <i>Controller</i> | 49 |
| Gambar 4.13 Hasil Pengujian <i>Load</i> 27,5 gram dengan FLC | 49 |
| Gambar 4.14 Hasil Pengujian <i>Load</i> 27,5 gram dengan ANFIS..... | 50 |
| Gambar 4.15 Hasil Pengujian <i>Load</i> 33 gram dengan PID <i>Controller</i> | 51 |
| Gambar 4.16 Hasil Pengujian <i>Load</i> 33 gram dengan FLC | 51 |
| Gambar 4.17 Hasil Pengujian <i>Load</i> 33 gram dengan ANFIS | 52 |
| Gambar 4.18 Hasil Pengujian <i>Load</i> 38,5 gram dengan PID <i>Controller</i> | 53 |
| Gambar 4.19 Hasil Pengujian <i>Load</i> 38,5 gram dengan FLC | 53 |
| Gambar 4.20 Hasil Pengujian <i>Load</i> 38,5 gram dengan ANFIS | 54 |

DAFTAR TABEL

| | |
|---|----|
| Tabel 2.1 Penelitian Sebelumnya | 5 |
| Tabel 2.2 Spesifikasi Arduino Uno [13] | 7 |
| Tabel 2.3 Spesifikasi A2212/6T 2200KV [19] | 15 |
| Tabel 2.4 Pengaktifan CW 3-Fase BLDC [19] | 15 |
| Tabel 3.1 Alat dan Bahan | 24 |
| Tabel 3.2 Aturan FLC | 34 |
| Tabel 3.3 Aturan ANFIS | 38 |
| Tabel 4.1 Perbandingan Kinerja <i>Controller</i> dengan Posisi Awal -15° | 44 |
| Tabel 4.2 Perbandingan Kinerja <i>Controller</i> dengan Posisi Awal 0° | 46 |
| Tabel 4.3 Perbandingan Kinerja <i>Controller</i> dengan Posisi Awal 15° | 48 |
| Tabel 4.4 Perbandingan Kinerja <i>Controller</i> dengan <i>Load</i> 27,5 gram | 50 |
| Tabel 4.5 Perbandingan Kinerja <i>Controller</i> dengan <i>Load</i> 33 gram | 52 |
| Tabel 4.6 Perbandingan Kinerja <i>Controller</i> dengan <i>Load</i> 38,5 gram | 54 |
| Tabel 4.7 Perbandingan Kinerja <i>Controller</i> Pengujian Posisi Awal | 55 |
| Tabel 4.8 Perbandingan Kinerja <i>Controller</i> Pengujian Pembebanan | 55 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Penelitian dan pengembangan dalam bidang teknologi akan terus berlangsung kapan pun dan di mana pun. Salah satunya adalah pengembangan pengendalian motor listrik. Motor listrik telah banyak diaplikasikan dalam berbagai peralatan; produksi, medis, industri otomatis, instrumentasi, otomotif sampai kedirgantaraan. Contoh sederhana penggunaan motor listrik dalam bidang kedirgantaraan ada pada pesawat tanpa awak/ *Unmanned Aerial Vehicles* (UAV). Sistem pengendalian kecepatan motor listrik yang mampu menyeimbangkan posisi pesawat adalah salah satu fitur wajib yang harus dimiliki pesawat. Sistem penyeimbang posisi pesawat pada beberapa model pesawat tertentu, sudah harus aktif sejak pesawat akan lepas landas. Seperti model helikopter atau UAV yang sedang populer saat ini, yaitu *drone* yang biasanya berjenis *quadrotor* akan menyeimbangkan seluruh badan pesawat saat lepas landas. Jenis ini berupaya mencapai titik yang stabil dan presisi dengan menyeimbangkan gaya yang dihasilkan oleh keempat rotornya [1].

Kendali penyeimbang posisi juga mendukung sistem *Vertical Take Off Landing* (VTOL) dan telah berkembang dengan beberapa metode. Metode yang saat ini sering dipakai pada VTOL adalah metode *Proportional Integral Derivative* (PID). Untuk meningkatkan kinerja kendali maka beberapa penelitian muncul untuk mencoba menggunakan sistem kendali yang lain, atau bahkan mengombinasikan beberapa sistem kendali tersebut. Salah satu sistem kendali yang juga sering digunakan adalah sistem kendali logika *fuzzy* dengan *Fuzzy Logic Controller* (FLC) atau *Adaptive Network-based Fuzzy Inference System* (ANFIS). Sistem kendali ini dapat diterapkan dan mampu menangani perubahan kondisi sistem walaupun model *plant* belum diketahui [2]. Penelitian dan pengembangan metode untuk VTOL akan terus berkembang seiring dengan berkembangnya jenis atau model pesawat.

Saat ini para peneliti cukup terbantu dengan adanya *software-software* yang mampu menampilkan hasil akhir rancangan proyeknya, tanpa harus mengeluarkan biaya untuk membuat proyeknya dalam bentuk yang nyata.

Peneliti dapat menyimulasikan perancangan dan melakukan pengujian pada *software*. Kemudian diwujudkan dengan pembuatan prototipe sebelum membuat proyek pada dimensi yang sebenarnya. Salah satu *software* yang digunakan untuk melakukan penelitian adalah Matlab dengan Simulink sebagai salah satu fitur yang dikembangkan oleh MathWorks. Inc. Simulink merupakan bahasa pemrograman grafis yang dapat menampilkan aliran data untuk keperluan modeling, simulasi, dan analisis terhadap suatu sistem yang dinamis sebelum dipindahkan ke *hardware*. Hal tersebut merupakan salah satu keunggulan Simulink untuk programmer pemula karena programmer tidak perlu menuliskan sintaks C, C++ atau HDL *code* tetapi komponen sintaks sudah tersajikan dalam bentuk blok [3] [4]. Selain itu Simulink juga mampu men-*deploy* atau mengintegrasikan sistem pada *hardware* secara *real time*.

Dari referensi yang telah dijelaskan, penelitian kali ini akan dibuat sistem kendali posisi VTOL dengan membandingkan metode PID, FLC dan ANFIS menggunakan *interface* Simulink, yang mampu berintegrasi dengan *embedded system* tanpa perlu memprogram ulang *embedded system* yang digunakan.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka dapat ditarik beberapa rumusan masalah antara lain sebagai berikut:

1. Bagaimana merancang sistem kendali posisi pada VTOL?
2. Bagaimana sistem kendali posisi diuji pada prototipe VTOL dengan *interface* Simulink?
3. Bagaimana perbandingan kinerja sistem kendali posisi dengan metode PID, FLC dan ANFIS bekerja pada prototipe VTOL?

Ruang lingkup permasalahan dalam penelitian ini akan dibatasi dengan:

1. Menggunakan *Hardware-in-the-loop (HIL) Simulation*.
2. Pengendalian posisi hanya pada satu sumbu, berdasarkan pembacaan sudut pada sumbu X.
3. Metode kendali yang dipakai adalah metode PID, FLC dan ANFIS.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang sistem kendali posisi sederhana pada VTOL.
2. Melakukan pengujian sistem kendali posisi prototipe VTOL dengan interface Simulink.
3. Melakukan analisis perbandingan kinerja sistem kendali posisi dengan metode PID, FLC dan ANFIS pada prototipe VTOL.

1.4 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah membuat sistem kendali posisi VTOL dengan metode PID, FLC dan ANFIS menggunakan *interface* Simulink untuk memberikan rekomendasi terkait sistem kendali.

1.5 Sistematika Penulisan

Penulisan penelitian ini terdiri dari lima bab, yaitu:

| | |
|---------|---|
| BAB I | PENDAHULUAN |
| | Penjelasan singkat mengenai latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan. |
| BAB II | TINJAUAN PUSTAKA |
| | Berisi referensi penelitian sebelumnya, menguraikan sistematis dasar teori dan komponen penelitian. |
| BAB III | METODOLOGI PENELITIAN |
| | Uraian metode, tahap-tahap penelitian yang digambarkan melalui diagram alir penelitian dan variabel yang akan diteliti. |
| BAB IV | HASIL DAN PEMBAHASAN |
| | Menyajikan dan menjelaskan hasil temuan data yang dianalisis dari perancangan sistem yang telah dibuat. |
| BAB V | KESIMPULAN DAN SARAN |
| | Menyimpulkan dengan pernyataan singkat yang telah dijabarkan dari hasil penelitian serta merupakan jawaban dari tujuan penelitian dan memberikan saran untuk melanjutkan atau mengembangkan penelitian sejenis. |

BAB II

TINJAUAN PUSTAKA

Bab ini berisi referensi penelitian sebelumnya, menguraikan sistematis dasar teori dan komponen penelitian. Adapun dasar teori yang akan dibahas dari komponen penelitian adalah VTOL *system* dengan penyusun sistemnya, sistem kendali PID, FLC dan ANFIS, serta Simulink.

2.1 Penelitian Sebelumnya

Penelitian atau percobaan yang telah dilakukan sebelumnya yang berkaitan dengan topik VTOL dengan metode PID, FLC dan ANFIS menggunakan Simulink akan dirangkum dalam Tabel 2.1. Perbedaan penelitian ini dengan penelitian sebelumnya adalah pada penelitian [7] dan [8] diambil sebagai referensi dan dikembangkan logika *fuzzy* dari model simulasinya untuk diterapkan pada prototipe VTOL. Pada penelitian [9] diambil referensi HIL dan dikembangkan menggunakan metode ANFIS. Pada percobaan [10], [11], dan [12] diambil sebagai referensi VTOL *system* dan dikembangkan dengan metode FLC serta ANFIS dalam HIL.

2.2 VTOL System

Vertical Take Off Landing (VTOL) mengacu pada pesawat yang dapat lepas landas, melayang, dan mendarat secara vertikal [5]. Tipikal VTOL berbasis seperti desain helikopter atau desain multirotor yang menggabungkan empat atau lebih baling-baling yang menciptakan daya angkat dan dorong untuk pesawat. VTOL multirotor dapat dilengkapi dengan berbagai muatan, termasuk kamera dengan resolusi tinggi, sensor multispektral, dan sensor pemantauan lingkungan seperti CO₂ dan detektor radiasi.

VTOL memiliki beberapa keunggulan dibandingkan pesawat tanpa awak sayap tetap (UAV *fixed wing*), yaitu 1.) membutuhkan lebih sedikit ruang untuk lepas landas dan mendarat, karena tidak menggunakan landasan 2.) cocok untuk inspeksi dan pemantauan di mana pesawat harus mempertahankan posisi untuk jangka waktu tertentu. 3.) bermanuver lebih baik daripada pesawat *fixed wing*,

Tabel 2.1 Penelitian Sebelumnya

| No | Peneliti | Judul | Keterangan |
|----|--|--|---|
| 1. | Mehmet Çunkaş dan Omer Aydoğdu [7] | <i>Realization of Fuzzy Logic Controlled Brushless DC Motor Drives Using Matlab/Simulink</i> | Disajikan sebuah model simulasi yang efisien untuk <i>drive</i> motor DC yang dikendalikan oleh logika <i>fuzzy</i> menggunakan Matlab / Simulink. Motor BLDC dikendalikan secara efisien menggunakan FLC. Algoritma kontrol, logika <i>fuzzy</i> dan PID dibandingkan. Karakteristik dinamis motor BLDC (yaitu kecepatan dan torsi) serta arus dan tegangan komponen inverter diamati dan dianalisis dengan menggunakan model yang dikembangkan. |
| 2. | Mohammed Rabah, Ali Rohan, Yun-Jong Han, dan Sung-Ho Kim [8] | <i>Design of Fuzzy-PID Controller for Quadcopter Trajectory-Tracking</i> | Dalam tulisan ini pengontrol <i>fuzzy</i> -PID dirancang untuk menstabilkan <i>quadcopter</i> pada lintasan tertentu menggunakan kecepatan sebagai input. Pengontrol yang diusulkan dibandingkan dengan pengontrol PD dan pengontrol logika <i>fuzzy</i> (FLC) untuk membedakan mana yang memiliki kinerja yang lebih baik menggunakan MATLAB / Simulink. Hasil simulasi menunjukkan bahwa di lintasan yang berbeda pengontrol <i>fuzzy</i> -PID menunjukkan respons yang lebih baik daripada dua pengontrol lainnya. |
| 3. | Muchamad Malik dan Aan Burhanuddin [9] | Desain Model Fuzzy Control UAV Berbasis Matlab/Simulink Pada Arduino Flight Controller | Dalam penelitian ini peneliti menyajikan FLC untuk <i>quadrocopter</i> . Metode dalam penelitian ini adalah dengan perancangan perangkat keras. Setelah itu desain untuk pengontrol <i>fuzzy</i> . Kemudian kontroler <i>fuzzy</i> yang dirancang diuji dalam <i>Hardware In Loop</i> (HIL). Hasil percobaan dan validasi fungsi |

| | | | |
|----|---------------------------------|--|---|
| | | | penerapan pengontrol dianggap memuaskan dan disimpulkan bahwa mungkin untuk menstabilkan <i>quadrocopter</i> dengan pengontrol logika <i>fuzzy</i> . |
| 4. | Jorge García Tíscar [10] | <i>Arduino + Matlab/Simulink: Controlador PID</i> | Dalam percobaan ini dilakukan pemrograman pengontrol PID di Simulink dengan sistem komunikasi serial. Bertujuan untuk mengendalikan posisi sistem yang sederhana " <i>one degree of freedom helicopter</i> " dan secara bertahap membuat sistem lebih rumit. Percobaan ini membuktikan bagaimana mungkin, dengan biaya yang sangat rendah, untuk membangun sistem yang memungkinkan teori kontrol PID untuk dipraktikkan. |
| 5. | Rodrigo Salazar Zugasti [11] | <i>PVTOL (Plannar Vertical Take Off Landing) - Eje X</i> | Dalam percobaan ini dilakukan penerapan PID dalam kontrol stabilitas sistem PVTOL dengan hasil memuaskan karena sistem tersebut distabilkan dalam referensi yang diberikan dengan memperbaiki kesalahan memrogram variabel speed drive, menggunakan konverter data dalam Simulink khusus UINT8 untuk dapat memproses sinyal, penundaan tertentu dalam sistem dapat diperbaiki dengan memilih nilai PID yang tepat. |
| 6. | Electronoobs Team [12] | <i>PID Control With Arduino</i> | Dalam proyek ini dilakukan percobaan mekanisme pengendalian beberapa motor BLDC untuk mengalibrasi <i>drone</i> . Dengan menempatkan motor pada neraca keseimbangan dan menghitung sudut menggunakan MPU6050. Nilai yang dikendalikan oleh PID adalah sudut kemiringan <i>drone</i> . Kesalahan $e(t)$ akan menjadi perbedaan antara sudut nyata <i>drone</i> dan yang diinginkan |

karena kemampuannya memvariasikan kecepatan setiap rotor untuk menciptakan perubahan dorong dan torsi yang lebih baik. Sedangkan kerugian utama dibandingkan dengan UAV *fixed wing* adalah daya tahan yang lebih rendah. Hal ini disebabkan VTOL menggunakan rotornya untuk menghasilkan gaya angkat dan dorong, sedangkan pesawat *fixed wing* hanya membutuhkan sistem propulsi mereka untuk menghasilkan daya dorong, karena daya angkat dihasilkan oleh sayap [6].

Salah satu pengujian kendali VTOL secara sederhana ditunjukkan pada penelitian atau percobaan yang dilakukan oleh Jorge García Tíscar [10], Rodrigo Salazar Zugasti [11] dan Electronoobs Team [12]. Percobaan sederhana pada salah satu sumbu/ *axis* dengan beberapa komponen yang dibutuhkan antara lain *controller*, sensor, *driver* motor, motor BLDC dan baterai sebagai suplai energi.

2.2.1 Controller

Arduino Uno sebagai *controller* yang mengatur sistem kendali posisi pada prototipe VTOL adalah papan mikrokontroler berbasis ATmega328P. Arduino Uno memiliki 14 pin *input / output* digital (6 pin sebagai *output* PWM), 6 pin *input* analog, koneksi USB, *power jack* DC, header ICSP, dan tombol reset. "Uno" berarti satu dalam bahasa Italia dan dipilih untuk menandai rilis Arduino *Software* (IDE) 1.0. Papan Uno dan versi 1.0. Arduino *Software* (IDE) adalah versi referensi pertama Arduino [13].

Tabel 2.2 Spesifikasi Arduino Uno [13]

| | |
|--------------------------------|---|
| <i>Microcontroller</i> | ATmega328P |
| <i>Operating Voltage</i> | 5V |
| <i>Digital I/O Pins</i> | 14 (<i>of which 6 provide PWM output</i>) |
| <i>Analog Input Pins</i> | 6 |
| <i>DC Current per I/O Pin</i> | 20 mA |
| <i>DC Current for 3.3V Pin</i> | 50 mA |
| <i>Flash Memory</i> | 32KB (ATmega328P) |
| <i>Clock Speed</i> | 16 MHz |
| <i>Length</i> | 68.6 mm |
| <i>Width</i> | 53.4 mm |
| <i>Weight</i> | 25 g |



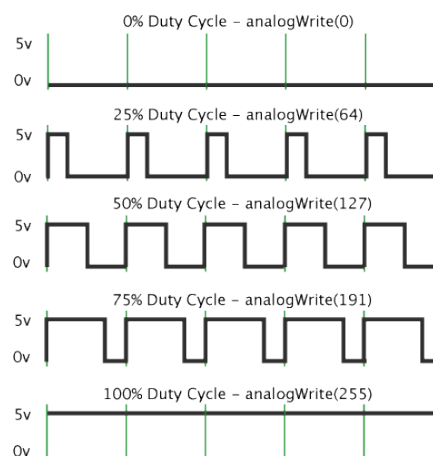
Gambar 2.1 *PinOut* Arduino Uno [13]

Arduino akan diprogram melalui *software* Arduino IDE sebagai inisiasi sistem kendali yang kemudian akan diintegrasikan dengan Simulink menggunakan komunikasi serial. Komunikasi serial digunakan untuk komunikasi antara papan Arduino dan komputer atau perangkat lain. Semua papan Arduino memiliki setidaknya satu port serial (juga dikenal sebagai UART atau USART). Pada Uno, pin 0 dan 1 digunakan untuk komunikasi dengan komputer. Menghubungkan apa saja dengan pin ini dapat mengganggu komunikasi serial, termasuk menyebabkan proses pengunggahan yang gagal [14]. Ada pun serial monitor bawaan Arduino IDE untuk berkomunikasi dengan papan Arduino dapat dilihat melalui tombol serial monitor di *toolbar* dan pilih *baud rate* yang sama dengan yang digunakan dalam *code* untuk membaca data serial.

Selain komunikasi serial, pada pembuatan sistem kendali posisi prototipe VTOL diperlukan komunikasi *Inter-Integrated Circuit* (I2C) untuk saling berbagi informasi dengan *gyroscope sensor*. Misalkan dua papan diprogram untuk berkomunikasi dalam konfigurasi *Master Writer / Slave Receiver* melalui serial I2C. Arduino *Master* diprogram untuk mengirim 6 byte data setiap setengah detik ke *Slave*, setelah pesan diterima, pesan kemudian dapat dilihat di jendela monitor serial papan *Slave*. Protokol I2C

melibatkan penggunaan dua jalur untuk mengirim dan menerima data: pin *Serial Clock* (SCL) yang dikeluarkan papan Arduino Master secara berkala, dan pin *Serial Data* (SDA) tempat data dikirim antara kedua perangkat. Satu bit informasi terbentuk secara berurutan sesuai alamat perangkat dan perintah ditransfer dari papan ke perangkat I2C melalui jalur SDA. Ketika informasi ini dikirim, perangkat yang dipanggil menjalankan permintaan dan mengirimkan kembali data melalui jalur yang sama menggunakan sinyal yang dihasilkan oleh *Master* pada SCL. Delapan bit awal dari *Master* ke *Slave* berisi alamat perangkat yang diinginkan oleh *Master*. Bit-bit setelahnya berisi alamat memori pada *Slave* yang diinginkan oleh *Master*. Setiap perangkat *Slave* harus memiliki alamatnya sendiri dan kedua perangkat *master* dan *slave* perlu bergiliran berkomunikasi melalui jalur data yang sama. Dengan cara ini, papan Arduino dapat berkomunikasi dengan banyak perangkat lain hanya menggunakan dua pin mikrokontroler [15].

Selanjutnya Arduino perlu melakukan pengendalian *Pulse Width Modulation* (PWM) untuk mengendalikan *Electrictronic Speed Control* (ESC). PWM adalah teknik mendapatkan hasil analog dengan cara digital. Kontrol digital digunakan untuk membuat gelombang persegi sinyal yang diaktifkan antara *on* dan *off*. Pola *on-off* ini dapat mensimulasikan tegangan antara *on* penuh (5 Volts) dan *off* (0 Volts) dengan mengubah porsi waktu sinyal *on* dan *off*. Untuk mendapatkan berbagai nilai analog dapat dilakukan dengan mengubah atau memodulasi lebar pulsa.



Gambar 2.2 PWM [16]

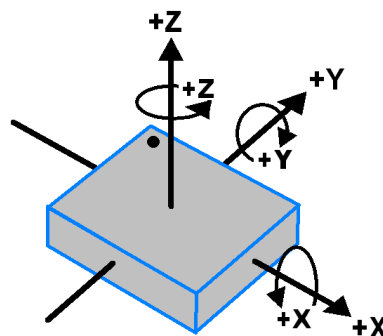
Dalam gambar 2.2 garis hijau mewakili periode waktu. Durasi atau periode ini adalah kebalikan dari frekuensi PWM. Dengan kata lain frekuensi PWM Arduino sekitar 500Hz, garis hijau akan mengukur masing-masing 2 milidetik. Panggilan ke `analogWrite ()` adalah skala 0 - 255, sehingga `analogWrite (255)` meminta *duty cycle* 100% (selalu aktif), dan `analogWrite (127)` adalah *duty cycle* 50% (separuh waktu) [16].

2.2.2 Position Sensor

MPU6050 *sensor module* pada sistem kendali posisi prototipe VTOL ini digunakan sebagai *position sensor*. MPU6050 *sensor module* dilengkapi *Motion Tracking Device 6-axis*. Modul ini menggabungkan 3-axis *gyroscope*, 3-axis *accelerometer* dan *Digital Motion Processor* dalam paket yang kecil. Modul ini juga memiliki fitur tambahan *on-chip temperature sensor*. Modul ini memiliki *interface bus* I2C untuk berkomunikasi dengan mikrokontroler [17].

- 3-Axis Gyroscope

MPU6050 terdiri dari 3-axis *gyroscope* dengan teknologi *Micro Electro Mechanical System* (MEMS), digunakan untuk mendeteksi kecepatan rotasi sepanjang sumbu X, Y, Z seperti yang ditunjukkan pada gambar 2.3.



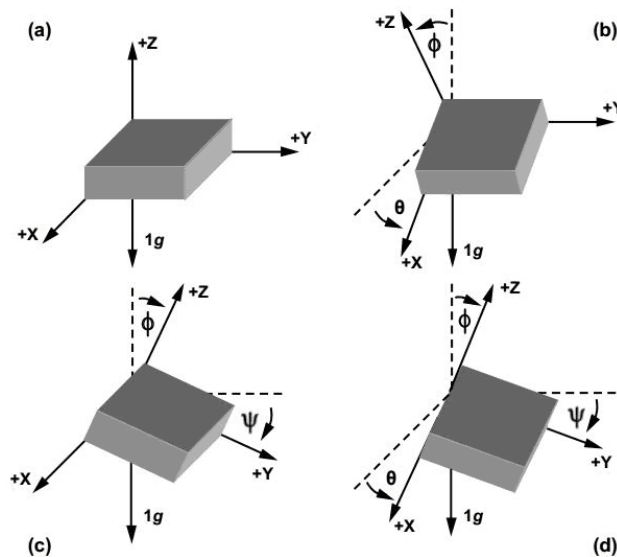
Gambar 2.3 Orientasi Putaran MPU6050 [17]

Ketika giroskop diputar pada salah satu sumbu, *Efek Coriolis* menyebabkan getaran yang terdeteksi oleh MEMS di dalam MPU6050. Sinyal yang dihasilkan diperkuat, didemodulasi, dan disaring untuk menghasilkan tegangan yang sebanding dengan kecepatan sudut. Tegangan ini didigitalkan menggunakan 16-bit ADC untuk setiap sumbu. Rentang *output*-nya adalah

+/- 250, +/- 500, +/- 1000, +/- 2000. Mengukur kecepatan sudut sepanjang setiap sumbu dalam satuan derajat per detik.

- *3-Axis Accelerometer*

MPU6050 terdiri dari *3-axis accelerometer* dengan teknologi MEMS ini digunakan untuk mendeteksi sudut kemiringan atau kemiringan sepanjang sumbu X, Y dan Z seperti yang ditunjukkan pada gambar 2.4.



Gambar 2.4 (a) Kondisi Normal, (b) Kemiringan pada Sumbu Z dan X, (c) Kemiringan pada Sumbu Z dan Y, (d) Kemiringan pada Sumbu X, Y, Z [17]

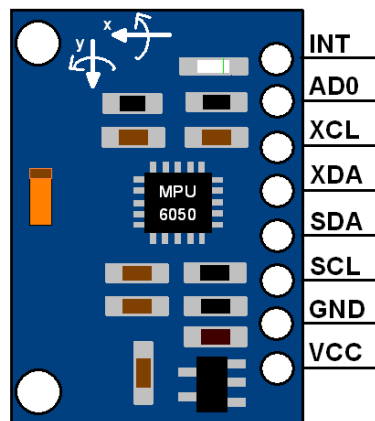
Akselerasi di sepanjang sumbu mengalihkan massa yang bisa bergerak. Perpindahan pelat bergerak ini menghasilkan *output* sensor. 16-bit ADC digunakan untuk mendapatkan *output* digital. Rentang *output*-nya adalah +/- 2g, +/- 4g, +/- 8g, +/- 16g diukur dalam satuan g (gaya gravitasi). Saat perangkat diletakkan pada permukaan datar, ia akan mengukur 0g pada sumbu X dan Y dan + 1g pada sumbu Z.

- *DMP (Digital Motion Processor)*

Embedded Digital Motion Processor (DMP) digunakan untuk menghitung algoritma pemrosesan gerak. Membutuhkan data dari giroskop, akselerometer, dan sensor tambahan

- *On-Chip Temperature Sensor*

Output on-chip temperature sensor didigitalkan menggunakan ADC dengan pembacaan suhu dapat dibaca dari register data sensor.



Gambar 2.5 MPU6050 *Sensor Module* [17]

Data sensor giroskop dan akselerometer modul MPU6050 terdiri dari data mentah 16-bit dalam bentuk *2's complement*. Data sensor suhu pada modul MPU6050 terdiri dari data 16-bit (bukan *2's complement*). Anggaplah

- Akselerometer rentang skala penuh $\pm 2g$ dengan Faktor Skala Sensitivitas 16.384 LSB (Hitungan) / g.
- Giroskop rentang skala penuh $\pm 250^\circ / s$ dengan Faktor Skala Sensitivitas 131 LSB (Hitungan) / $^\circ / s$.

kemudian untuk mendapatkan data mentah sensor, perlu mengubah *2's complement* pada data sensor akselerometer dan giroskop. Setelah mendapatkan data mentah sensor, perhitungan akselerasi dan kecepatan sudut didapat dengan membagi data mentah sensor dengan faktor skala sensitivitasnya sebagai berikut,

- Nilai akselerometer dalam g (g force)

Akselerasi sumbu X = (data mentah akselerometer sumbu X / 16384) g.

Akselerasi sumbu Y = (data mentah akselerometer sumbu Y / 16384) g.

Akselerasi sumbu Z = (data mentah akselerometer sumbu Z / 16384) g.

- Nilai giroskop dalam $^\circ / s$ (derajat per detik)

Kecepatan sudut sumbu X = (data mentah giroskop sumbu X / 131) $^\circ / s$.

Kecepatan sudut sumbu Y = (data mentah giroskop sumbu Y / 131) $^\circ / s$.

Kecepatan sudut sumbu Z = (data mentah giroskop sumbu Z / 131) $^\circ / s$.

- Nilai suhu dalam $^\circ / c$ (derajat per Celcius)

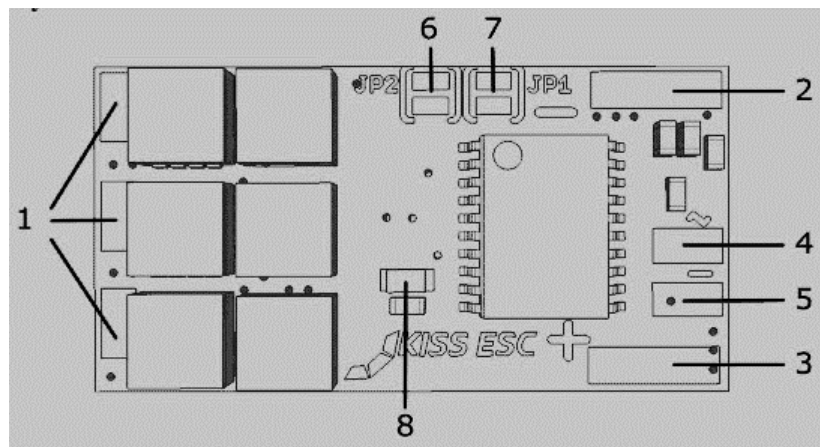
Temperatur dalam derajat C = ((data sensor suhu) / 340 + 36,53) $^\circ / c$.

Sebagai contoh,

Misalkan, setelah *2's complement* mendapatkan akselerometer sumbu X nilai mentah = +15454. Kemudian $A_x = +15454/16384 = 0,94 \text{ g}$.

2.2.3 Electricronic Speed Control (ESC)

Electricronic Speed Control (ESC) sebagai *driver motor* dalam sistem kendali posisi prototipe VTOL. ESC adalah sirkuit elektronik yang digunakan untuk mengubah kecepatan motor listrik. ESC sering digunakan untuk menyediakan tenaga listrik 3-fase yang diproduksi secara elektronik dari tegangan rendah untuk motor BLDC. ESC dapat berupa unit terpisah yang disambungkan dengan saluran kontrol atau disatukan dalam papan sirkuit tunggal sistem [18]. ESC dapat mengontrol kecepatan putaran motor pesawat terbang *radio controller* (RC). Fitur utama dari ESC adalah rangkaian eliminator baterai, cutoff tegangan rendah, dan rem.



Gambar 2.6 Komponen ESC [18]

Keterangan:

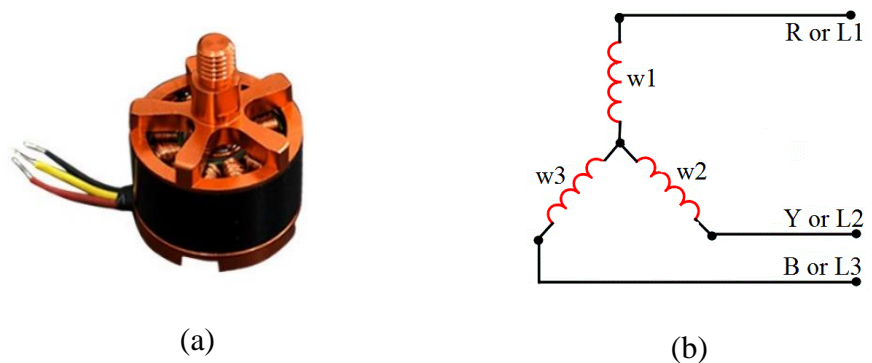
1. *Solder pads for the 3-BLDC motor phases*
2. *Negative (-) Li-Po connections*
3. *Positive (+) Li-Po Connection*
4. *Servo signal or input of the PWM signal*
5. *GND reference of PWM Signal*
6. *Solder jumper, for altering the direction of Rotation (CW/CCW)*
7. *Solder jumper, for varying the type of the PWM input signal*
8. *State LED*

Fungsi dasar ESC adalah untuk mengubah jumlah daya ke motor listrik dari baterai pesawat RC berdasarkan pada *input throttle stick*. Teknik ini bekerja dengan baik pada kecepatan penuh ketika baterai dihubungkan langsung ke motor. ESC membedakan daya ke motor dengan menyalakan dan mematikan daya dengan cepat. Transistor MOSFET digunakan sebagai saklar perangkat mekanis yang aktif sekitar 2000 kali per detik. Daya ke motor dapat diatur dengan mengubah besar waktu *on-off* dalam siklus tertentu. Ketika MOSFET dihidupkan, arus naik karena medan magnet pada belitan motor meningkat. Ketika MOSFET dimatikan, energi magnetik yang disimpan dalam belitan harus diserap oleh ESC.

2.2.4 BLDC Motor

Motor BLDC A2212/6T 2200KV yang digunakan pada sistem kendali posisi prototipe VTOL adalah motor brushless berkecepatan tinggi yang dirancang khusus untuk *quadcopters*, *drone* atau pesawat mainan. Motor adalah jenis *outrunner*, di mana kasing luar berputar (rotor) sementara magnet di dalam tetap (stator) [19]. 22 berarti diameter motor yaitu 22 mm, dengan diameter yang lebih besar tipe motor ini dapat menghasilkan torsi yang lebih besar. 12 berarti ketinggian motor yaitu 12 mm. 6T berarti jumlah *turns per pole*, jumlah yang lebih sedikit berarti memiliki kemampuan kecepatan tinggi tetapi torsi rendah. 2200KV berarti jumlah putaran motor per volt dari *supply*, jika tegangan *supply* 10 volts maka motor akan berputar 22000RPM.

Motor ini memiliki tiga pin dengan koneksi internal di antara ketiganya ditunjukkan gambar 2.7 (b), R (Red) atau L1 dihubungkan pada fase 1, dst.



Gambar 2.7 (a) Motor BLDC A2212/6T 2200KV dan (b) Pin Internal [19]

Tabel 2.3 Spesifikasi A2212/6T 2200KV [19]

| | |
|--------------------------------------|---|
| <i>Operating voltage</i> | <i>7.2V to 12V (2 to 3Li-poly or 6to10 NiCad)</i> |
| <i>No load current</i> | <i>0.5Amp</i> |
| <i>Maximum current</i> | <i>13Amp for 60Sec</i> |
| <i>Maximum Watts</i> | <i>150 Watt</i> |
| <i>Weight of motor</i> | <i>50-60 grams</i> |
| <i>Maximum operating temperature</i> | <i>+ 80°C</i> |

Motor ini digerakkan oleh tenaga listrik tiga fase dan tidak seperti motor DC yang diaktifkan dengan menyambung motor langsung ke daya. Perlu mengikuti pola tertentu untuk menyalakan tiga fase motor agar bisa aktif. Untuk berputar searah jarum jam (*Clock Wise/ CW*) maka di bawah ini:

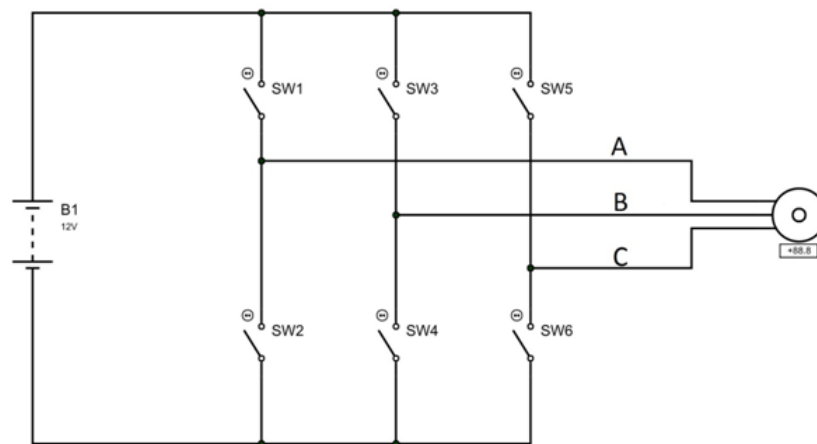
Tabel 2.4 Pengaktifan CW 3-Fase BLDC [19]

| <i>STEP</i> | 1 | 2 | 3 | 4 | 5 | 6 | 7 or 1 |
|-----------------|---|---|---|---|---|---|--------|
| <i>POSITIVE</i> | R | R | Y | Y | B | B | R |
| <i>GROUND</i> | Y | B | B | R | R | Y | Y |
| <i>OPEN</i> | B | Y | R | B | Y | R | B |

Keterangan:

R = *Red* atau A
Y = *Yellow* atau B
B = *Black* atau C

Untuk arah sebaliknya, maka perlu mengubah step pada tabel 2.4 menjadi step 6 - 5 - 4 - 3 - 2 - 1 - 6. Untuk mewujudkan tabel di atas, sirkuit memiliki enam sakelar yang membentuk *h-bridge* yang terhubung ke motor gambar 2.8. Pengaktifan CW maupun CCW 3-fase BLDC menggunakan sirkuit *switching* dapat dilakukan dengan menggunakan ESC yang tersambung dengan *controller* yang tepat.



Gambar 2.8 Sirkuit *Switching* Pengaktifan Motor BLDC [19]

2.2.5 Li-Po 3S Battery

Baterai Li-Po 3S digunakan sebagai *power supply* dalam sistem kendali posisi prototipe VTOL. Baterai Lithium Polymer (Li-Po) adalah jenis baterai yang sekarang digunakan di banyak perangkat elektronik [20]. Beratnya ringan dan dapat dibuat dalam hampir semua ukuran atau bentuk. Memiliki kapasitas yang tinggi dan memungkinkan memiliki daya lebih besar. Tingkat *discharge* lebih tinggi. Tetapi memiliki umur yang jauh lebih pendek; rata-rata Li-Po hanya 150–250 siklus. Terbuat dari bahan kimia yang sensitif dan dapat menyebabkan kebakaran jika baterai tertusuk. Serta memerlukan perawatan khusus untuk pengisian, pemakaian, dan penyimpanan.



Gambar 2.9 Baterai Li-Po 3S [20]

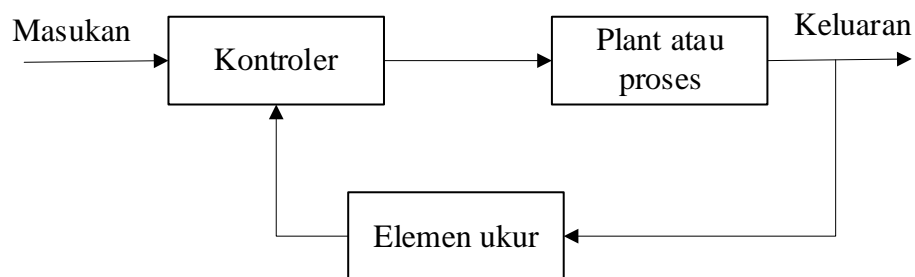
- Tegangan / Jumlah Sel, Sel Li-Po memiliki tegangan nominal 3,7V. Untuk baterai 7.4V berarti ada dua sel (2S) tergabung secara seri dan

tiga sel (3S) adalah 11.1V, dan seterusnya. Tegangan secara langsung mempengaruhi RPM motor listrik.

- Kapasitas, Kapasitas baterai adalah ukuran besar daya baterai dapat bertahan. Satuan ukurannya miliamp jam (mAh) yang berarti lama baterai dikosongkan dalam satu jam.
- *Discharge Rating*, *Discharge Rating* adalah ukuran seberapa cepat baterai dapat habis dengan aman dan tanpa merusak baterai.

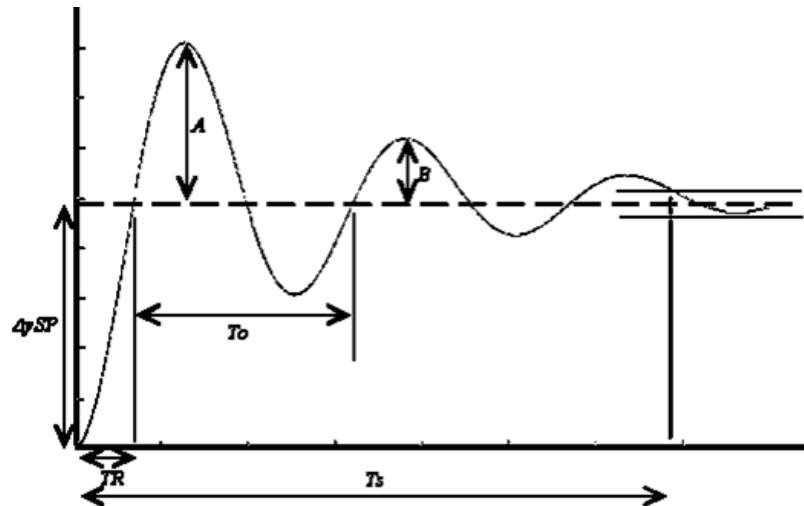
2.3 Sistem Kendali

Sistem kendali yang digunakan pada penelitian ini adalah jenis sistem regulator otomatis (*automatic regulating system*) yaitu sistem kendali berumpan-balik dengan *input* acuan dan *output* yang diinginkan konstan atau berubah terhadap waktu dengan melambat dan tugas utamanya adalah menjaga keluaran yang sebenarnya pada nilai yang diinginkan dengan adanya gangguan. Serta menggunakan sistem kendali lup tertutup (*closed-loop control system*) yaitu sistem kendali yang sinyal *outputnya* mempunyai pengaruh langsung pada aksi pengendalian. Jadi, sistem kendali lup tertutup adalah sistem kendali berumpan-balik. Sinyal kesalahan *actuator* merupakan selisih antara sinyal *input* dan sinyal umpan-balik, diumpankan ke *controller* untuk memperkecil kesalahan dan membuat *output* sistem mendekati nilai yang diinginkan [21].



Gambar 2.10 Sistem Kendali Lup Tertutup [21]

Ada beberapa karakteristik yang dapat digunakan untuk membandingkan kualitas respons sistem.



Gambar 2.11 Grafik Respons Sistem [22]

- *Rise Time*, waktu yang diambil untuk respons proses pertama kali mencapai titik acuan, dalam gambar 2.11 ditandai 'TR'.
- *Settling Time*, waktu yang dibutuhkan sistem untuk menyesuaikan dalam toleransi di sekitar titik acuan, dalam gambar 2.11 ditandai 'Ts'.
- *Overshoot* dan *Undershoot* adalah kuantitas yang relatif untuk respons titik acuan, dalam gambar 2.11 *overshoot* ditandai dengan 'A'. Sedangkan untuk *undershoot* berada di bawah sumbu x.

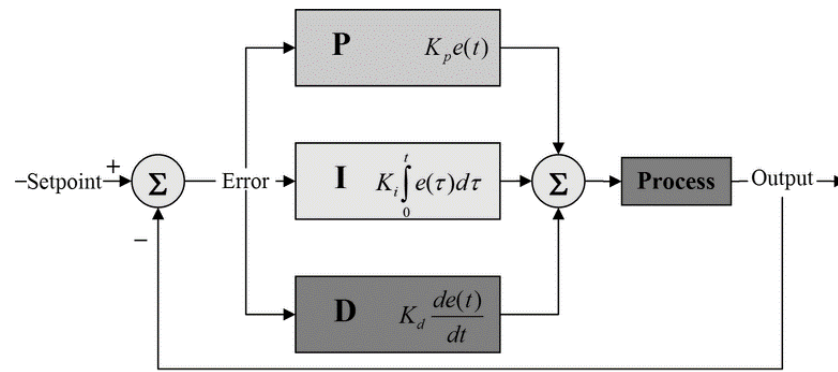
To adalah periode osilasi satu siklus, dan B adalah puncak kedua pada respons.

Selanjutnya kualitas respons sistem dapat dihitung dari *Integral Squared Error* (ISE), dengan mengkuadratkan *error* dari waktu ke waktu. ISE yang minim akan cenderung menghilangkan *error* besar dengan cepat, tetapi akan mentolerir kesalahan kecil yang bertahan untuk jangka waktu yang lama. Seringkali ini mengarah pada respons yang cepat, tetapi dengan osilasi yang besar [22].

Ada pun pengendalian sistem yang digunakan dalam penelitian ini yaitu dengan metode PID, FLC dan ANFIS.

2.3.1 Proportional Integral Derivative

Kontroler PID menghitung nilai "*error*" sebagai perbedaan antara variabel yang diukur dan titik setel yang diinginkan. Pengontrol seperti yang ditunjukkan pada gambar 2.12 mencoba untuk meminimalkan kesalahan dengan menyesuaikan *input* proses kendali [23].

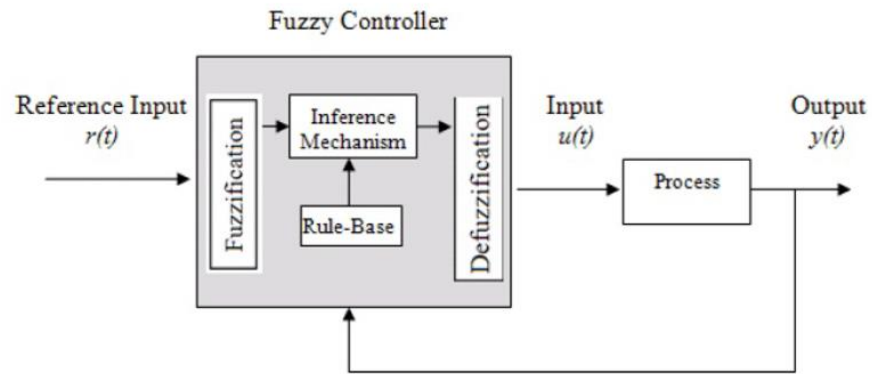


Gambar 2.12 Blok Diagram PID Controller [23]

Pemilihan jenis pengendali dan parameternya (K_p ; K_i ; K_d) tergantung dengan model proses yang akan dikendalikan. Penyesuaian dan pemilihan parameter pengendali untuk mencapai kendali yang memuaskan pada dasarnya ada pada masalah optimisasi. Ziegler dan Nichols (Z-N) mengembangkan metode loop tertutup untuk penyetelan parameter. Dalam penelitian ini, parameter PID akan disetel dengan metode tuning Z-N.

2.3.2 Fuzzy Logic Controller

Logika *fuzzy* adalah salah satu kecerdasan buatan yang mampu membuat mesin lebih cerdas dan memungkinkan untuk bernalar dengan cara yang kabur seperti manusia. FLC membuat pemetaan nonlinear antara *input* dan *output* menggunakan fungsi keanggotaan dan aturan linguistik (biasanya berbentuk *If_Then_*). Sistem *fuzzy* menggunakan dua *input*, variabel *error* (e) dan perubahan *error* (de), serta satu variabel *output* (Y) [23]. Struktur komputasi skema logika *fuzzy* terdiri dari *fuzzifier*, *rule base*, *inference engine*, dan *defuzzifier*, seperti yang ditunjukkan pada gambar 2.13. *Fuzzifier* mengubah *crisp value* (*real value*) menjadi anggota set *fuzzy*. FIS merumuskan pemetaan dari *input* yang diberikan ke *output* menggunakan logika *fuzzy*, sementara *defuzzifier* mengubah *output* yang ditentukan oleh sistem inferensi menjadi *crisp value*. Dalam FLC, fungsi keanggotaan digunakan untuk menemukan keanggotaan elemen dalam set yang diberikan.



Gambar 2.13 Blok Diagram *Fuzzy Logic Control* [2]

R_1 : If X is A_1 and Y is B_1 then Z is C_1

R_2 : If X is A_2 and Y is B_2 then Z is C_3

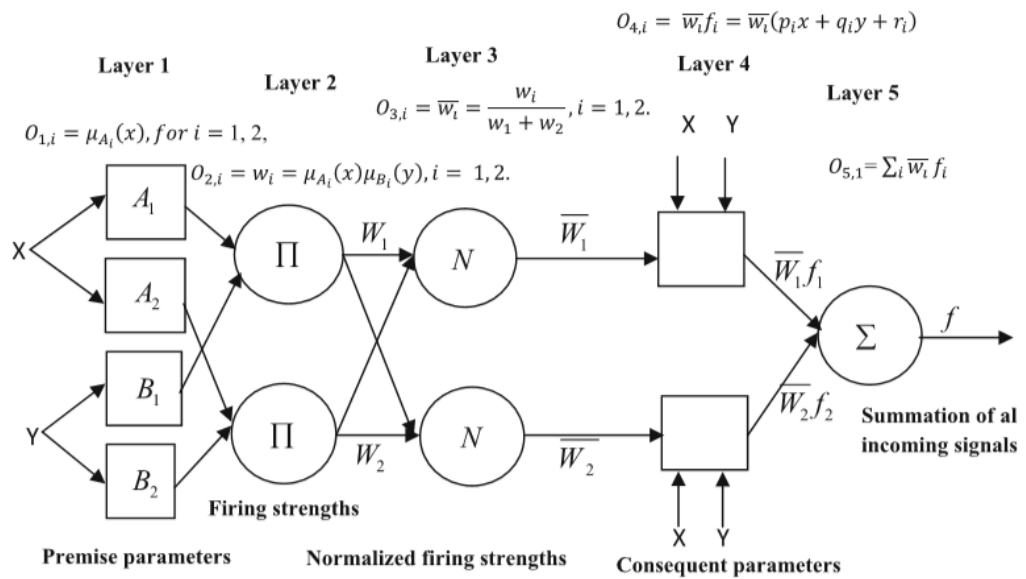
\vdots \vdots \vdots \vdots

R_n : If X is A_n and Y is B_n then Z is C_n

Aturan kontrol fuzzy memiliki bentuk seperti diatas, di mana X , Y , dan Z adalah variabel linguistik yang mewakili dua variabel status proses dan satu variabel kontrol, masing-masing; A_i ; B_i dan C_i adalah nilai linguistik dari variabel X , Y , dan Z dalam semesta pembicaraan U , V , dan W [2], masing-masing, dengan $i = 1, 2, 3 \dots n$.

2.3.3 Adaptive Network-based Fuzzy Inference System

ANFIS adalah kombinasi dari jaringan saraf dan sistem fuzzy sedemikian rupa sehingga jaringan saraf digunakan untuk menentukan parameter sistem fuzzy [23]. Diilustrasikan pada gambar 2.14, sistem *neuro-fuzzy* dengan kemampuan belajar jaringan saraf dapat meningkatkan kinerja secara signifikan dan dapat menyediakan mekanisme untuk memasukkan data pengamatan ke dalam proses klasifikasi. Sistem inferensi fuzzy yang dipertimbangkan memiliki dua *input* x dan y dan satu *output* f .



Gambar 2.14 Arsitektur ANFIS [23]

Lapisan pertama adalah parameter premis, dihitung dengan fungsi keanggotaan parameter. *Output* dari lapisan kedua adalah produk dari dua sinyal yang masuk, *output* lapisan ketiga menemukan *firing strengths* dari masing-masing node, *output* dari lapisan 4 adalah produk dari *firing weight* yang dinormalisasi dengan fungsi node, dan *output* lapisan 5 adalah keseluruhan dari model ANFIS.

2.4 Simulink

Simulink merupakan bahasa pemrograman grafis yang dapat menampilkan aliran data untuk keperluan modeling, simulasi, dan analisis terhadap suatu sistem yang dinamis sebelum dipindahkan ke *hardware* [3]. Simulink memiliki beberapa keunggulan, antara lain:

- Simulink untuk Pemodelan dan Simulasi Sistem, *Engineer* dan ilmuwan menggunakan Simulink untuk melakukan pemodelan dan simulasi multidomain, karena Simulink dapat menggunakan kembali model untuk disimulasikan bersama pada semua bagian sistem. Simulink juga bisa menggabungkan model ke dalam satu simulasi tingkat sistem bahkan jika sistem itu tidak dibangun di Simulink [24].
- Simulink untuk Verifikasi, Validasi, dan Tes, *Engineering teams* menggunakan Desain Berbasis Model dengan Matlab dan Simulink untuk

merancang sistem tertanam yang rumit dan menghasilkan kode C, C ++, dan HDL. Alat dari MathWorks ini menggunakan pengujian simulasi dan analisis statis untuk melengkapi Desain Berbasis Model dengan kekakuan dan otomatisasi untuk menemukan kesalahan lebih awal dan mencapai kualitas yang lebih tinggi [25].

- Simulink untuk Sistem Tertanam. Dengan menekan satu tombol, Simulink dapat menghasilkan kode dan menjalankannya pada perangkat keras [26]. Berikutnya dari pembuatan prototipe misal pada papan Arduino dapat menggunakan *hardware support package add-ons*.

Pada penelitian ini menggunakan *System Identification Toolbox* untuk membangun model matematika sistem dinamis dari data input-output terukur menggunakan data input-output time-domain untuk mengidentifikasi fungsi transfer. *Fuzzy Logic Toolbox* untuk menganalisis, merancang, dan mensimulasikan sistem berdasarkan logika *fuzzy*. *Neuro-Fuzzy Designer* untuk merancang, melatih, dan menguji sistem ANFIS menggunakan data pelatihan input / output.

BAB III

METODOLOGI PENELITIAN

Penelitian ini terdiri dari beberapa tahapan yang dimulai dari perancangan dan pembuatan alat/ *hardware* berdasarkan referensi. Kemudian melakukan pengecekan masing-masing komponen untuk pengaktifan sederhana sebelum menjadi kendali sistem. Jika sudah sesuai, selanjutnya perancangan dan pembuatan algoritma pemrograman sistem kendali pada *software*. Sistem akan diuji dan dilakukan evaluasi serta analisis dari berbagai kendali yang digunakan.

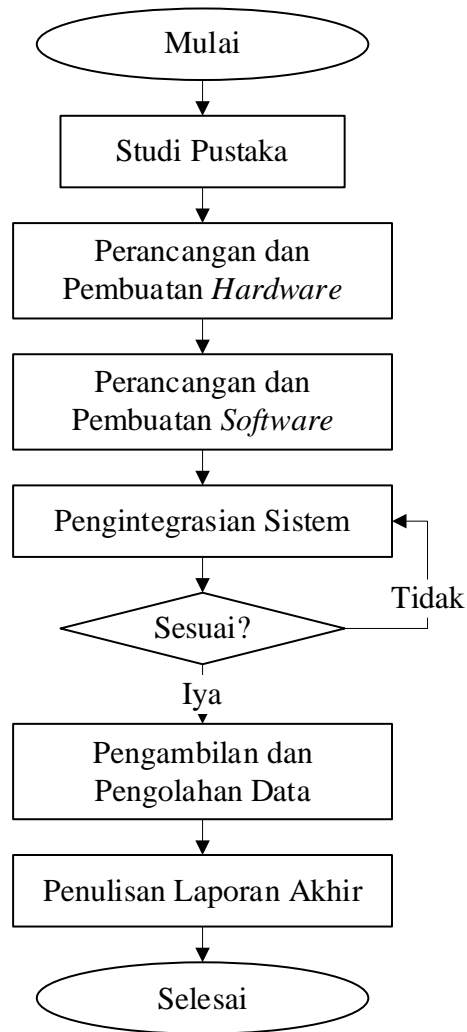
3.1 Alur Penelitian

Penelitian dimulai dengan studi pustaka yang dapat berasal dari *paper*, laporan penelitian, *datasheet*, buku, maupun dokumentasi percobaan/ *trial* dengan topik yang relevan. Setelah pengetahuan dasar dipahami dari referensi yang didapat, penelitian dilanjutkan dengan pembuatan dan penyusunan *hardware* serta pengecekan komponen sistem yang tidak lepas dari *software*.

Pembuatan *hardware* meliputi pembuatan kerangka VTOL hingga penempatan komponen *controller*, sensor, *actuator*, dan *supply* agar sistem dapat bekerja maksimal dan efisien tanpa kendala ketika aktif nantinya. Untuk pembuatan *software*, terlebih dahulu dilakukan pengecekan/ pengaktifan masing-masing sub sistem dengan Arduino IDE sebelum nantinya diintegrasikan dengan sistem kendali yang dibuat pada Simulink.

Setelah semua dapat aktif atau bekerja dengan baik, maka akan dilakukan pengujian menggunakan beberapa sistem kendali dan data yang didapat akan dianalisis untuk mengetahui hasil kinerja sistem. Kemudian penelitian ini diakhiri dengan penulisan laporan sesuai data yang didapat. Diagram alir penelitian dapat dilihat pada gambar 3.1.

Waktu penelitian dimulai pada bulan Januari 2020 – Juni 2020, yang awalnya dilaksanakan di Laboratorium Teknik Elektro Universitas Sebelas Maret kemudian dilanjutkan di rumah penulis karena adanya pandemi Covid-19.



Gambar 3.1 Diagram Alir Penelitian

3.2 Alat dan Bahan

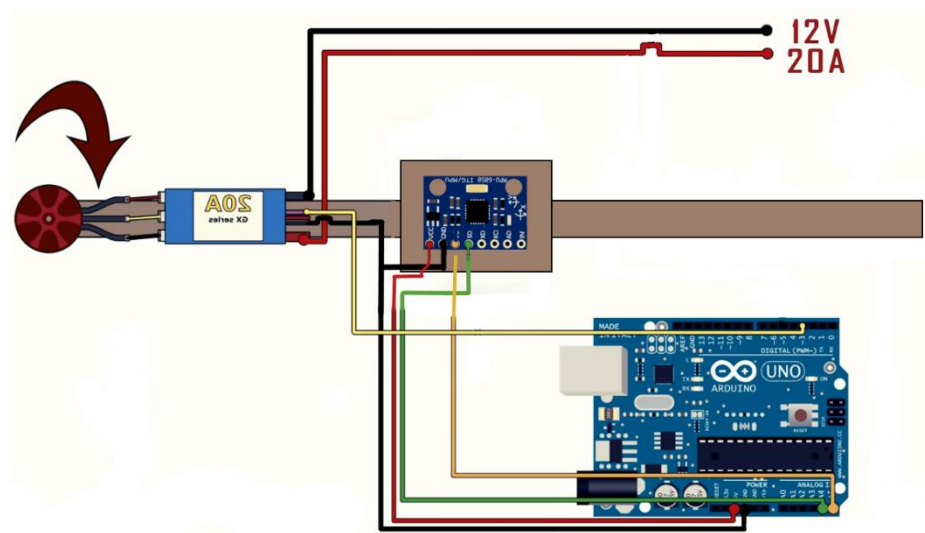
Berikut adalah daftar alat dan bahan yang digunakan sebagai penyusun *hardware* dan *software* untuk melaksanakan penelitian ini,

Tabel 3.1 Alat dan Bahan

| No | Nama Alat / Spesifikasi | Kuantitas | Keterangan |
|----|-------------------------------|-----------|---|
| 1 | Motor BLDC A2212/6T 2200KV | 1 buah | Sebagai penggerak utama sistem dilengkapi dengan propeler |
| 2 | ESC 20A | 1 buah | Sebagai <i>driver</i> motor BLDC |
| 3 | MPU6050 sensor modul | 1 buah | Sebagai sensor posisi |
| 4 | Arduino Uno | 1 buah | Sebagai <i>controller</i> |
| 5 | Baterai Li-Po 3S | 1 buah | Sebagai sumber energi sistem |

| | | | |
|----|---|--------|---|
| 6 | PC | 1 buah | Sudah ter- <i>install</i> Arduino IDE dan Simulink-Matlab 2016b |
| 7 | Kerangka VTOL | 1 set | Sebagai pembangun prototipe yang terbuat dari besi ringan |
| 8 | Mini <i>project board</i> | 1 buah | Sebagai tempat sensor |
| 9 | <i>Switch</i> | 1 buah | <i>Switch on-off</i> pemutus baterai |
| 10 | Kabel Jumper male-male dan male-female serta konektor | 7 buah | Sebagai penghubung antar komponen agar terintegrasi menjadi sebuah sistem |
| 11 | Multimeter | 1 buah | Sebagai pengukur tegangan sisa baterai |
| 12 | Toolset | 1 set | Sebagai peralatan untuk menyusun prototipe |

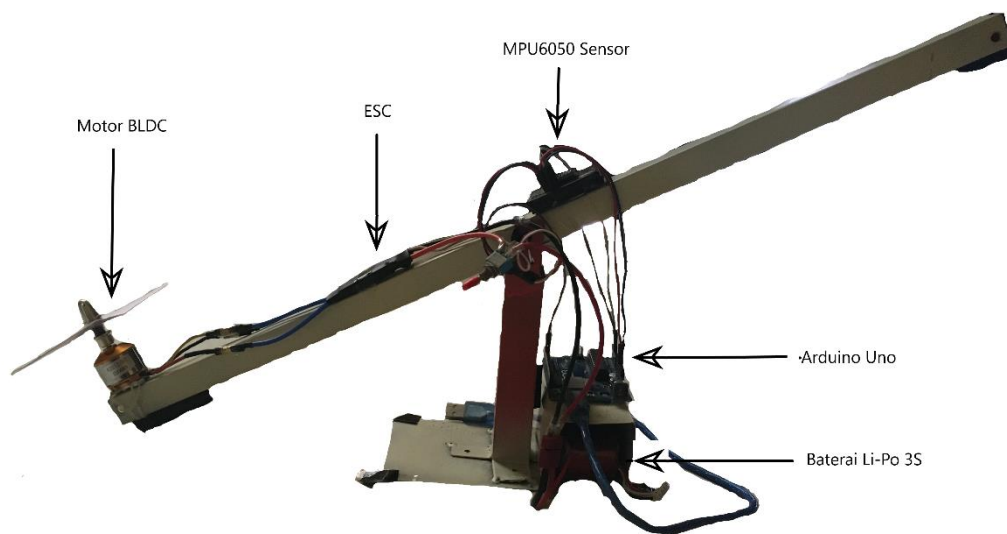
3.3 Perancangan dan Pembuatan *Hardware*



Gambar 3.2 *Wiring* Komponen VTOL

Pada gambar 3.2 merupakan *wiring* dan penempatan komponen pada neraca keseimbangan. Disini sensor MPU6050 ditempatkan seperti pada gambar, sehingga pembacaan sudut yang akan dianalisis adalah sudut sepanjang sumbu X. Sesuai dengan referensi dari percobaan yang dilakukan oleh *Electronoobs Team* [12] kerangka VTOL dibuat dengan besi ringan, tetapi pada penelitian ini poros neraca

keseimbangan tidak menggunakan *bearing*. Besi sepanjang ± 60 cm dengan bagian *center* yang dilubangi digunakan sebagai neraca keseimbangan sistem. Lalu sebagai penyangga, besi sepanjang 15 cm dengan ujung yang dipotong pada sepasang sisi yang berhadapan untuk penempatan neraca keseimbangan. Plat besi dengan panjang 10 cm dan lebar 15 cm dibuat sebagai alas alat. Plat dan penyangga digabungkan dengan pengelasan, sedangkan penyangga dengan neraca keseimbangan digabungkan dengan besi yang dimasukkan di poros besi keseimbangan. Diperlukan pemberat atau perekat pada alas agar alat tidak bergeser saat sistem aktif.

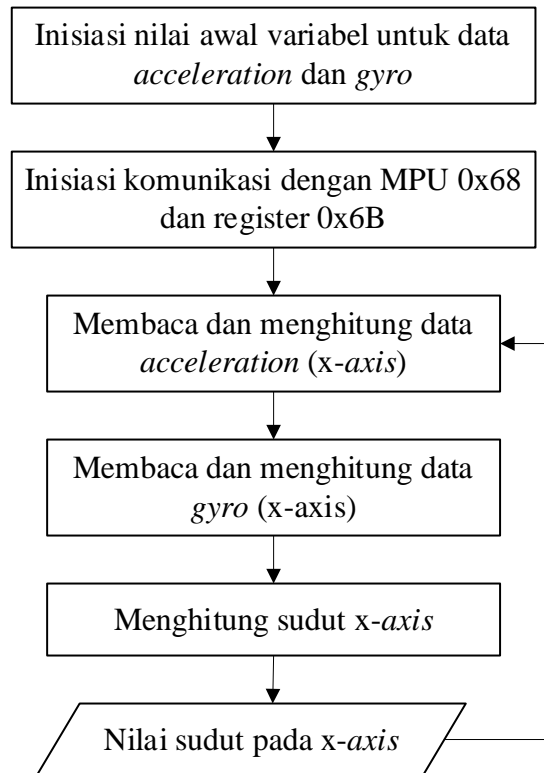


Gambar 3.3 Hasil Pembuatan *Hardware*

3.4 Perancangan dan Pembuatan *Software*

Perancangan dan pembuatan *software* atau program meliputi semua proses yang dilakukan mikrokontroler. Mulai dari menjalankan proses pembacaan sensor dan pengaktifan motor BLDC yang dilakukan pada Arduino IDE, pembuatan sistem kendali pada Simulink serta pengintegrasian kedua sistem.

3.4.1 Perancangan Algoritma Pembacaan Sensor



Gambar 3.4 Diagram Alir Algoritma Pembacaan Sensor MPU6050

Coding dimulai dengan inisiasi nilai awal variabel untuk data *acceleration* dan *gyro* yang akan digunakan untuk pemrograman seterusnya. Selanjutnya adalah inisiasi komunikasi sesuai dengan *addressing* MPU 0x68 dan register 0x6B. Kemudian program akan berulang/ *looping* membaca dan menghitung data yang di dapat dari MPU untuk mendapatkan nilai *acceleration* (x-axis) serta data *gyro* (x-axis) yang telah ter-convert menjadi satuan sudut.

```

Acceleration_angle[0] = atan((Acc_rawY/16384.0)
                             /sqrt(pow((Acc_rawX/16384.0),2)
                             + pow((Acc_rawZ/16384.0),2))) * rad_to_deg;

Gyro_angle[0] = Gyr_rawX/131.0;
  
```

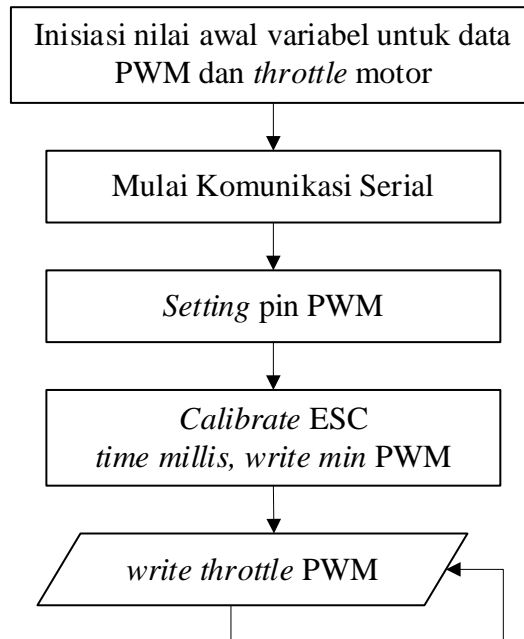
Setelah data didapat maka sudut x-axis dapat dihitung dengan *complementary filter*.

```

Total_angle[0] = 0.98 *(Total_angle[0] + Gyro_angle[0]*elapsedTime)
                + 0.02*Acceleration_angle[0];
  
```

Sehingga akan didapatkan nilai sudut pada x-axis dari pembacaan sensor.

3.4.2 Perancangan Algoritma Pengaktifan Motor BLDC



Gambar 3.5 Diagram Alir Algoritma Pengaktifan Motor BLDC

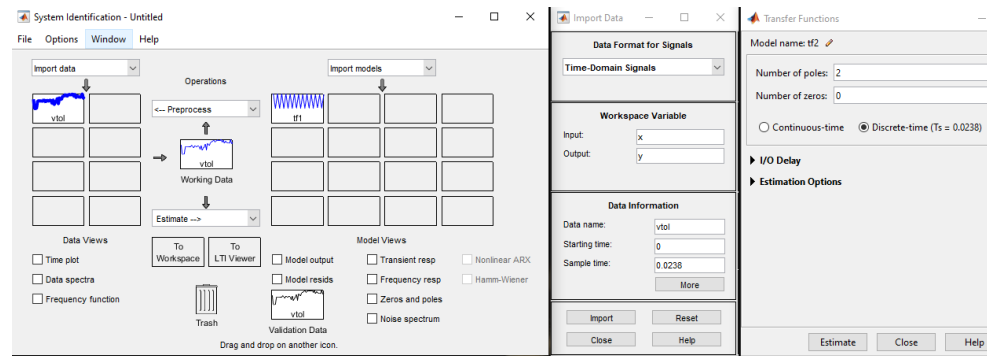
Coding dimulai dengan inisiasi nilai awal variabel untuk data PWM dan *throttle* motor yang akan digunakan untuk pemrograman seterusnya. Selanjutnya adalah memulai komunikasi serial, mengatur pin PWM yaitu pin 3, dan mengkalibrasi ESC dengan mulai menghitung waktu dengan *millis* serta mengirim sinyal minimum PWM. Kemudian program akan berulang/ *looping* membaca nilai PWM *throttle* yang diinginkan.

3.4.3 Perancangan Sistem Kendali

Sistem Kendali diprogram pada Arduino IDE dan Simulink. Dalam Arduino IDE, nilai PWM akan disesuaikan menurut nilai kendali yang diolah pada Simulink. Selanjutnya akan dibahas rancangan dan hasil metode kendali yang dibuat dalam Simulink. Dari percobaan yang dilakukan *Electronoobs Team* [12] sistem kendali posisi keseimbangan VTOL dapat dibuat dengan *full coding* menggunakan Arduino IDE. Data dari hasil percobaan tersebut digunakan pada penelitian ini sebagai referensi sistem yang diinginkan.

- *Proportional Integral Derivative*

Perancangan PID dimulai dengan menggunakan *System Identification Toolbox* untuk mendapatkan model sistem yang akan dikendalikan. *System Identification Toolbox* dapat diakses dengan mengetik “*systemidentification*” pada *command window* Matlab.

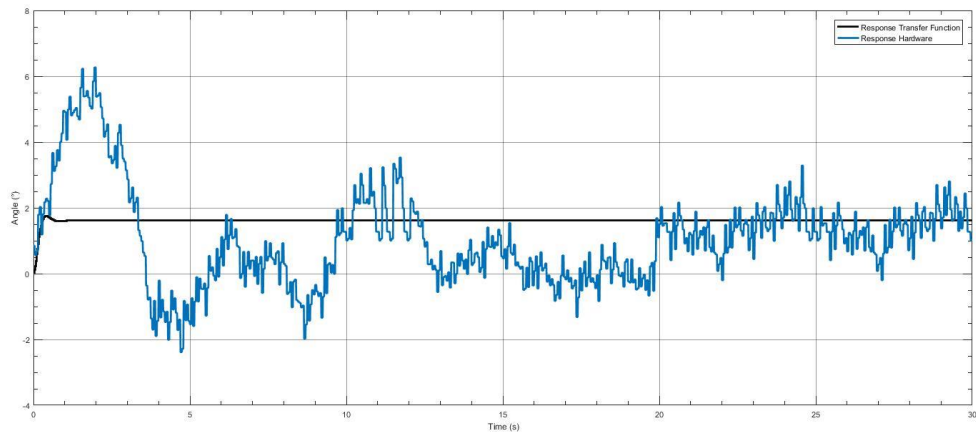


Gambar 3.6 *System Identification Toolbox*

Data referensi yang berbentuk data basis waktu, maka *Time-Domain Signal* dipilih pada pilihan *import data*. Selanjutnya *input* data “x” merupakan *workspace* yang berisi nilai kendali PWM dan *output* data “y” merupakan *workspace* yang berisi nilai sudut dari pembacaan sensor. Terdapat 784 data yang diambil selama ± 18 detik dengan *sample time* 0.0238. Percobaan untuk mencari estimasi fungsi alih menggunakan orde 2, dengan *poles* bernilai 2 dan *zero* bernilai 0. Pemilihan penggunaan orde 2 pada metode estimasi fungsi alih agar hasil persamaan dari fungsi alih yang didapat lebih ringkas dan tidak akan terlalu panjang. Fungsi alih menggunakan *discrete time* yang sama dengan *sample time*. Selanjutnya *estimate transfer function* untuk mendapatkan model sistem yang akan dikendalikan.

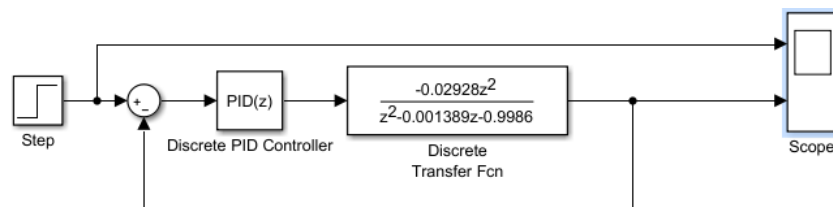
Selanjutnya *transfer function* divalidasi dengan mengamati hasil *response transfer function* di sistem simulink dengan *response hardware* yang diinput nilai konstan 5. Hasil *response transfer function* dan *response hardware* dapat dilihat pada gambar 3.7. Grafik pembacaan dimulai dari sudut 0° selama 30 detik. Terlihat kedua grafik memiliki *rise time* yang kurang dari satu detik, tetapi pada *response hardware* terdapat *overshoot* yang lebih tinggi dibanding *response transfer function* dan memiliki *undershoot* serta *setling time* lebih lama. Nilai *similarity* sebesar 14,16%

karena *response hardware* yang sensitif terhadap perubahan sudut. Serta *delay* komunikasi antara *board microcontroller* dengan PC. Hal tersebut mungkin dapat di minimalisir jika sistem dibuat *stand alone*. Tetapi terlihat *response hardware* semakin lama konvergen dengan *response transfer function* model simulasi, sehingga model yang digunakan sudah bisa mewakili respons sistem *real*.



Gambar 3.7 *Response Transfer Function* dan *Response Hardware*

Transfer function digunakan sebagai model sistem VTOL dalam Simulink sebagai acuan *controller* untuk mendapat nilai PID yang sesuai.

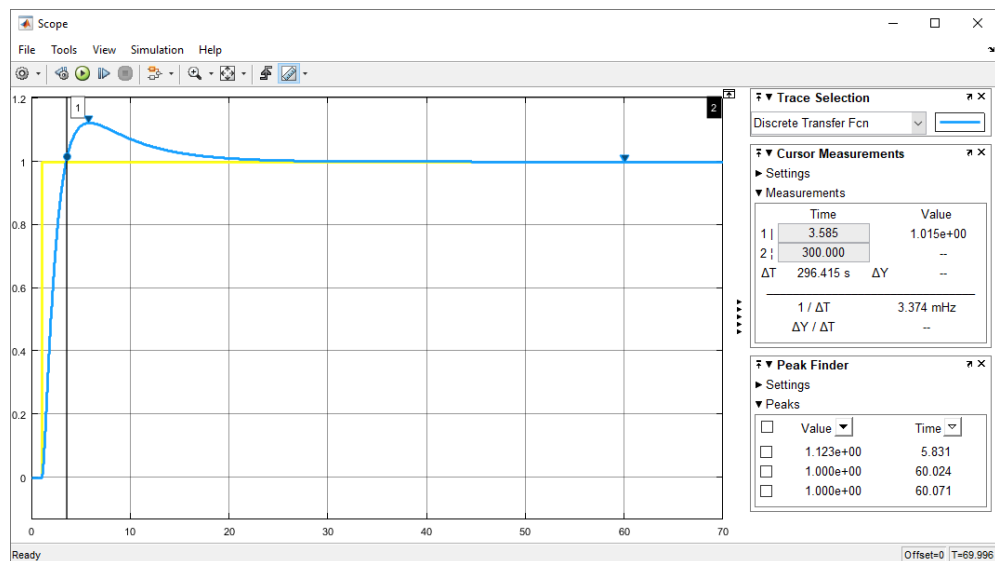


Gambar 3.8 Sistem PID *Controller*

Sistem VTOL direpresentasikan oleh blok *Discrete Transfer Function* dalam gambar 3.8. Sistem PID dengan penguatan proporsional yang tinggi akan mengakibatkan perubahan besar dalam *output*. Jika penguatan proporsional terlalu tinggi, sistem dapat menjadi tidak stabil. Sebaliknya, penguatan kecil menghasilkan respons *output* yang kecil untuk kesalahan *input* yang besar membuat *controller* menjadi kurang responsif. Nilai integratif digunakan untuk mempercepat pergerakan proses menuju *setpoint* dan menghilangkan sisa *error* yang terjadi pada *output* proporsional. Namun,

jika integral merespons akumulasi *error* dari sebelumnya, hal ini menyebabkan nilai sekarang akan melebihi *setpoint*. Tindakan derivatif memprediksi perilaku sistem dan dengan demikian meningkatkan *settling time* dan stabilitas sistem.

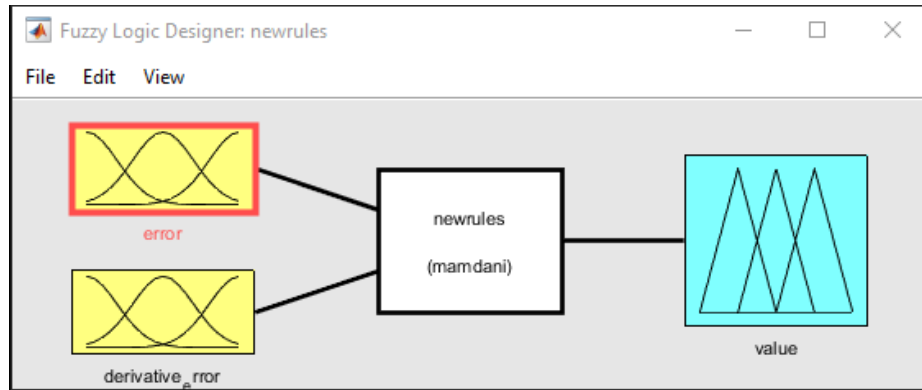
Blok PID pada Simulink menyediakan fitur *tuning* secara otomatis dan akan menampilkan kondisi sistem sebelum dan sesudah dituning beserta parameter tuning dari PID. Jika dirasa *tuning* masih kurang sesuai, nilai PID dapat diubah pada *tab Tuning Tools* agar mendapat hasil yang diinginkan. Proses *tuning* selesai dan mendapat nilai parameter Kp sebesar -1.27398414770311, nilai Ki sebesar -0.19545104336603, dan nilai Kd sebesar 0.287825045565896. Setelah melakukan *update* pada blok PID *controller*, sistem disimulasikan selama 70 detik dan mendapat *output* grafik pada gambar 3.9. Sistem dapat dikendalikan dengan menggunakan parameter PID diatas dengan *rise time* ± 3 detik, *overshoot* 12,3% pada kisaran detik ke-5, dan mencapai *steady state* saat 60 detik. Sistem dapat bekerja sesuai referensi data yang digunakan pada referensi percobaan [12]



Gambar 3.9 Grafik Hasil Pengendalian Sistem dengan PID *Controller*

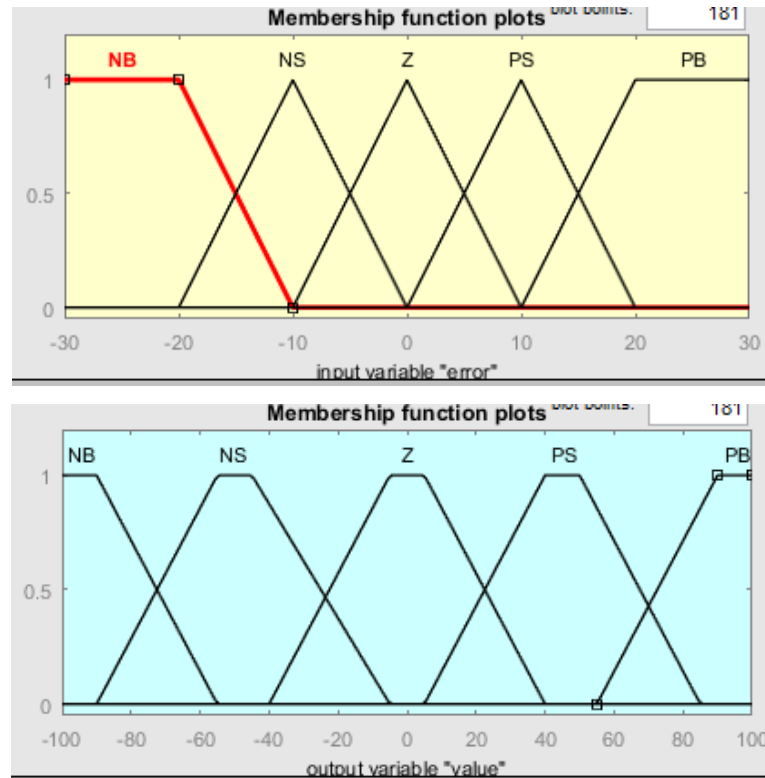
- *Fuzzy Logic Controller*

Perancangan FLC dimulai dengan menggunakan *Fuzzy Logic Designer Toolbox* yang dapat diakses dengan mengetik “*anfisedit*” pada *command window* Matlab. Pada menu File, membuat FIS baru dengan metode Mamdani.



Gambar 3. 10 *Fuzzy Logic Designer Toolbox*

FLC akan menggunakan dua *input*, yaitu *error* dan *derivative error* dan satu *output value*/ nilai kendali (gambar 3.10). Pada masing-masing blok memiliki fungsi keanggotaan/ *membership function* untuk mengklasifikasi nilai-nilai yang akan dikendalikan. Blok *input error* dan *derivative error* ini menggunakan rentang nilai -30 sampai 30, nilai tersebut dipilih karena sudut pembacaan sensor berkisar $\pm 26^\circ$ dan sudut yang diinginkan adalah 0° . Blok ini memiliki lima anggota dengan representasi tiga fungsi berbentuk segitiga dan dua fungsi berbentuk trapesium. Fungsi NB (*Negative Big*) berbentuk trapesium dan memiliki interval [-35 -30 -20 -10] serta fungsi PB (*Positive Big*) berbentuk trapesium dan memiliki interval [10 20 30 35] menggunakan fungsi keanggotaan trapesium karena ingin didapatkan nilai yang tetap/ konstan pada batas bawah dan atas. Selanjutnya fungsi NS (*Negative Small*) memiliki interval [-20 -10 0], fungsi Z (*Zero*) memiliki interval [-10 0 10], dan fungsi PS (*Positive Small*) memiliki interval [0 10 20]. Fungsi keanggotaan *input* dapat dilihat pada gambar 3.11 atas.



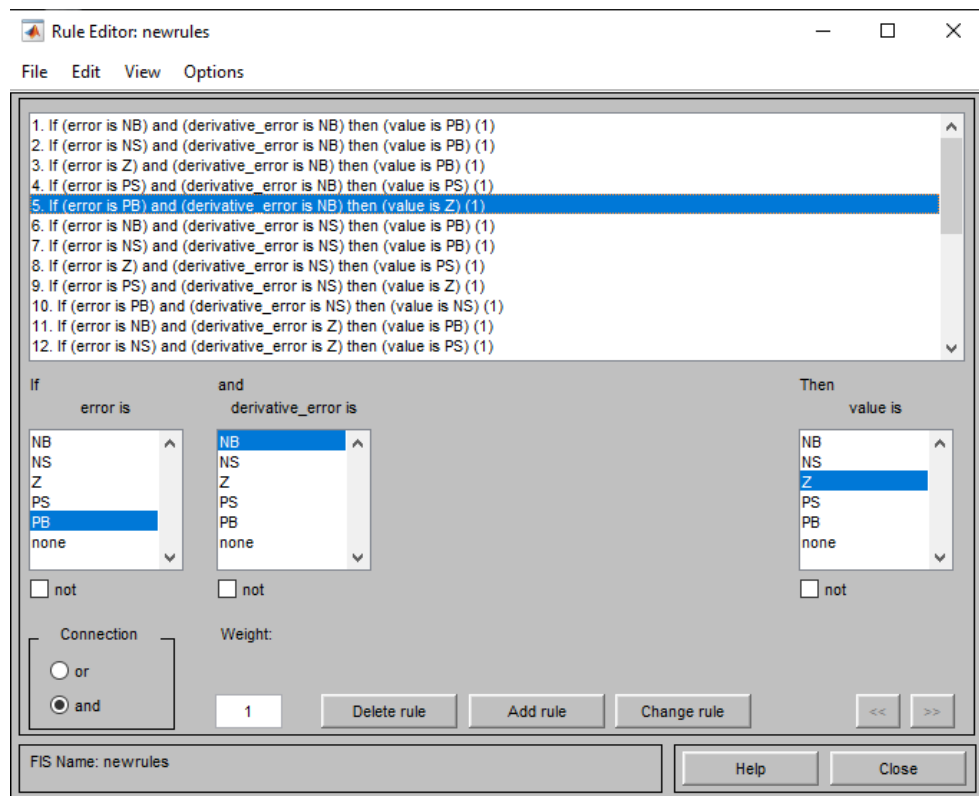
Gambar 3.11 Fungsi Keanggotaan *Input* dan *Output*

Blok *output value* pada gambar 3.11 bawah menggunakan rentang nilai -100 sampai 100, nilai tersebut dipilih karena menyesuaikan pemrograman yang telah direncanakan pada Arduino IDE. Agar nanti tidak perlu memprogram ulang Arduino saat penggantian *controller* pada Simulink. Blok ini memiliki lima anggota dengan representasi fungsi berbentuk trapesium. Fungsi NB memiliki interval [-135 -100 -90 -55], fungsi NS memiliki interval [-90 -55 -45 -5], fungsi Z memiliki interval [-40 -5 5 40], PS memiliki interval [5 40 50 85] serta fungsi PB memiliki interval [55 90 100 135]. Setelah menginisiasi fungsi keanggotaan pada masing-masing blok, selanjutnya perancangan pembuatan aturan/ *rule base* FLC pada *rule editor*.

Tabel 3.2 Aturan FLC

| <i>de/ e</i> | NB | NS | Z | PS | PB |
|--------------|-----------|-----------|----------|-----------|-----------|
| NB | PB | PB | PB | PS | Z |
| NS | PB | PB | PS | Z | NS |
| Z | PB | PS | Z | NS | NB |
| PS | PS | Z | NS | NB | NB |
| PB | Z | NS | NB | NB | NB |

Aturan di-input pada *rule editor* dengan memasukkan nilai *error*, *derivative error* dan *value*. Nilai di-input-kan dari baris pertama menurun, dilanjutkan baris kedua menurun, dst. Urutan penginputan aturan tidak berpengaruh pada apapun. Sebagai contoh pada gambar 3.12, aturan nomor 5 merupakan aturan yang didapat dari tabel 3.2 pada baris pertama kolom ke-5.

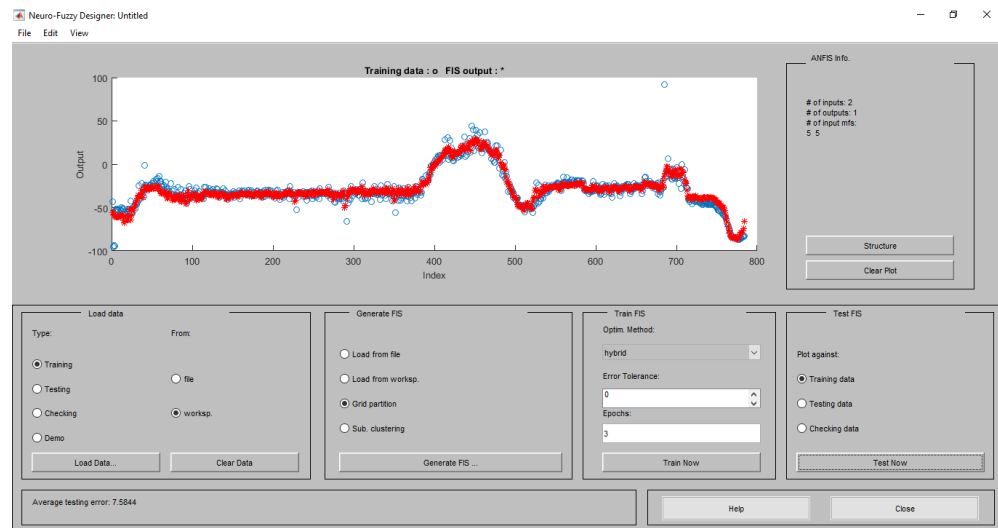


Gambar 3.12 Rule Editor

Selanjutnya FIS akan di-export ke *workspace* Matlab untuk nantinya dipanggil dan digunakan pada blok FLC di Simulink.

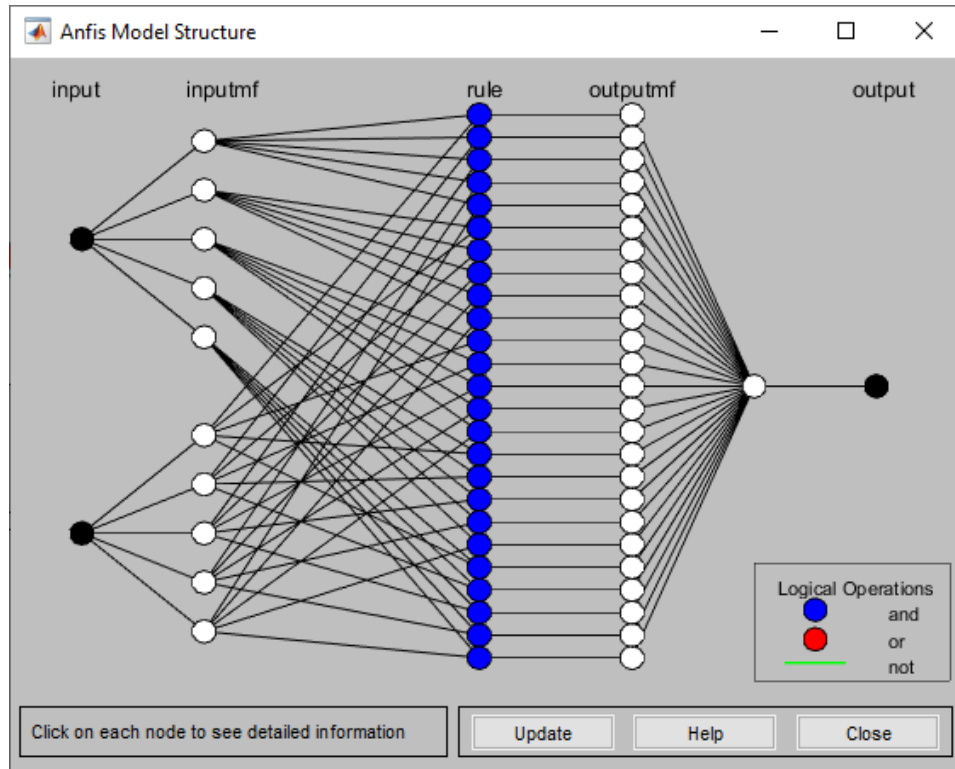
- *Adaptive Network-based Fuzzy Inference System*

Perancangan ANFIS menggunakan *Neuro-Fuzzy Designer Toolbox* yang dapat diakses seperti pembuatan FLC melalui “*anfisedit*” pada *command window*. Pada ANFIS, data yang akan di-training dimuat dalam *workspace* dengan *input1* merupakan data *error*, *input2* data *derivative error*, dan *output* adalah data *value* (nilai kendali yang diinginkan). *Toolbox* akan me-generate FIS serta me-training data agar didapatkan aturan yang sesuai dari *input* dan *output* dari data referensi. Gambar 3.13 menunjukkan hasil *training* dalam *neuro-fuzzy designer toolbox*



Gambar 3.13 Hasil *Training* dalam *Neuro-Fuzzy Designer Toolbox*

Hasil *testing error* pada ANFIS sebesar 7.5844. ANFIS dibuat dengan masing-masing lima fungsi keanggotaan segitiga pada *input* dan *output* konstan. Model struktur ANFIS dapat dilihat pada gambar 3.14, terdiri dari dua *node input*, 10 *node* fungsi keanggotaan *input*, 25 *node* aturan *fuzzy*, 25 *node* fungsi keanggotaan *output*, dan satu *node* nilai *output*.



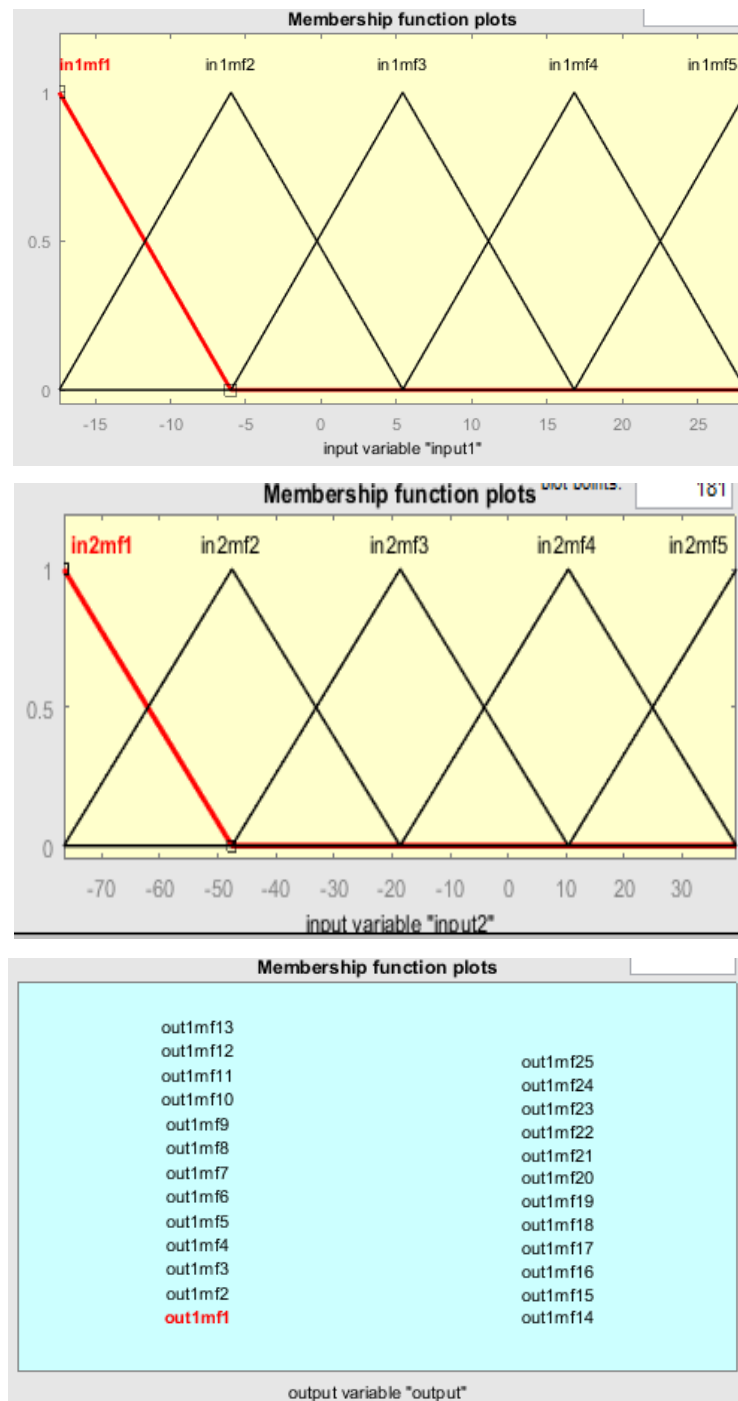
Gambar 3.14 ANFIS Model Structure

Saat *testing* selesai, ANFIS juga membuat fungsi keanggotaan dengan nama dan parameter otomatis dari data referensi. Blok *input1* menggunakan range nilai -17.38 sampai 28.18, memiliki lima anggota dengan in1mf1 memiliki interval [-28.77 -17.38 -5.995], in1mf2 memiliki interval [-17.38 -5.986 5.422], in1mf3 memiliki interval [-5.984 5.403 16.79], in1mf4 memiliki interval [5.393 16.79 28.18], dan in1mf5 memiliki interval [16.79 28.18 39.57].

Blok *input2* menggunakan range nilai -76.52 sampai 39.35, memiliki lima anggota dengan in2mf1 memiliki interval [-105.5 -76.52 -47.55], in2mf2 memiliki interval [-76.52 -47.55 -18.59], in2mf3 memiliki interval [-47.55 -18.59 10.39], in2mf4 memiliki interval [-18.59 10.38 39.35], dan in2mf5 memiliki interval [10.37 39.35 68.32].

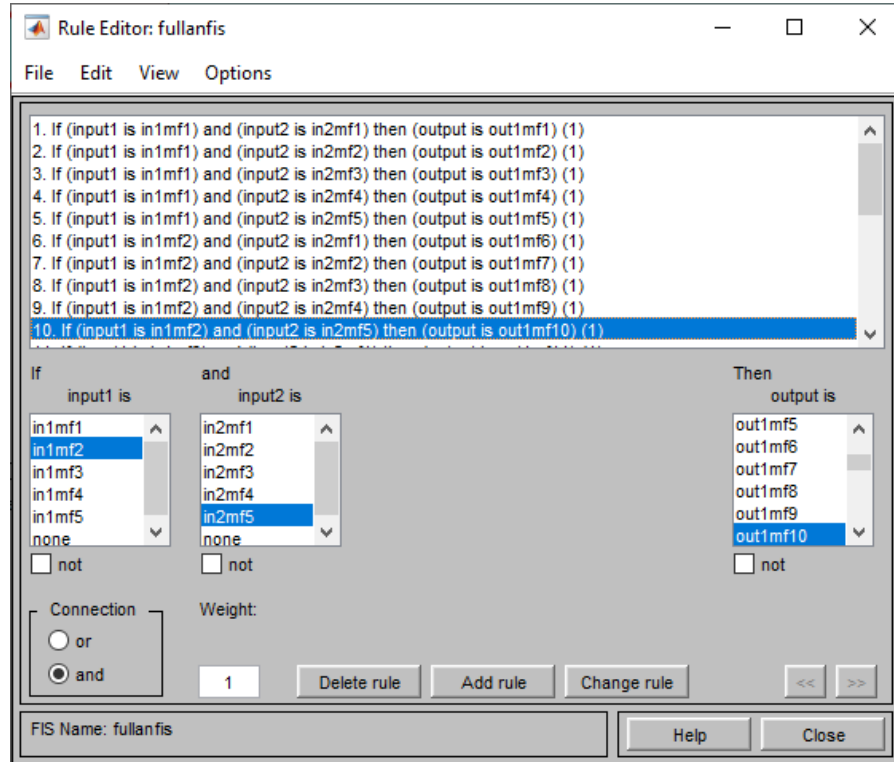
Blok *output* yang merupakan nilai konstan berisi out1mf1 bernilai 0, out1mf2 bernilai -125, out1mf3 bernilai 31.24, out1mf4 bernilai 16.27, out1mf5 bernilai 11.07, out1mf6 bernilai -34.78, out1mf7 bernilai 7.35, out1mf8 bernilai 16.84, out1mf9 bernilai -28.81, out1mf11 bernilai -22.33,

out1mf12 bernilai 20.59, out1mf13 bernilai -19.91, out1mf14 bernilai -30.84, out1mf15 bernilai -54.95, out1mf16 bernilai 0, out1mf17 bernilai 0.7105, out1mf18 bernilai -25.95, out1mf19 bernilai -47.99, out1mf20 bernilai -80.8, out1mf21 bernilai 0, out1mf22 bernilai -164.3, out1mf23 bernilai -36.3, out1mf24 bernilai -78.73, dan out1mf25 bernilai -116.8.



Gambar 3.15 Fungsi Keanggotaan *Input1* dan 2 serta *Output* ANFIS

ANFIS juga membuat aturan dengan nama dan parameter otomatis dari data referensi. Aturan pada *rule editor* disajikan dalam tabel 3.3. Selanjutnya FIS akan di-*export* ke *workspace* Matlab untuk nantinya dipanggil dan digunakan pada blok FLC di Simulink.

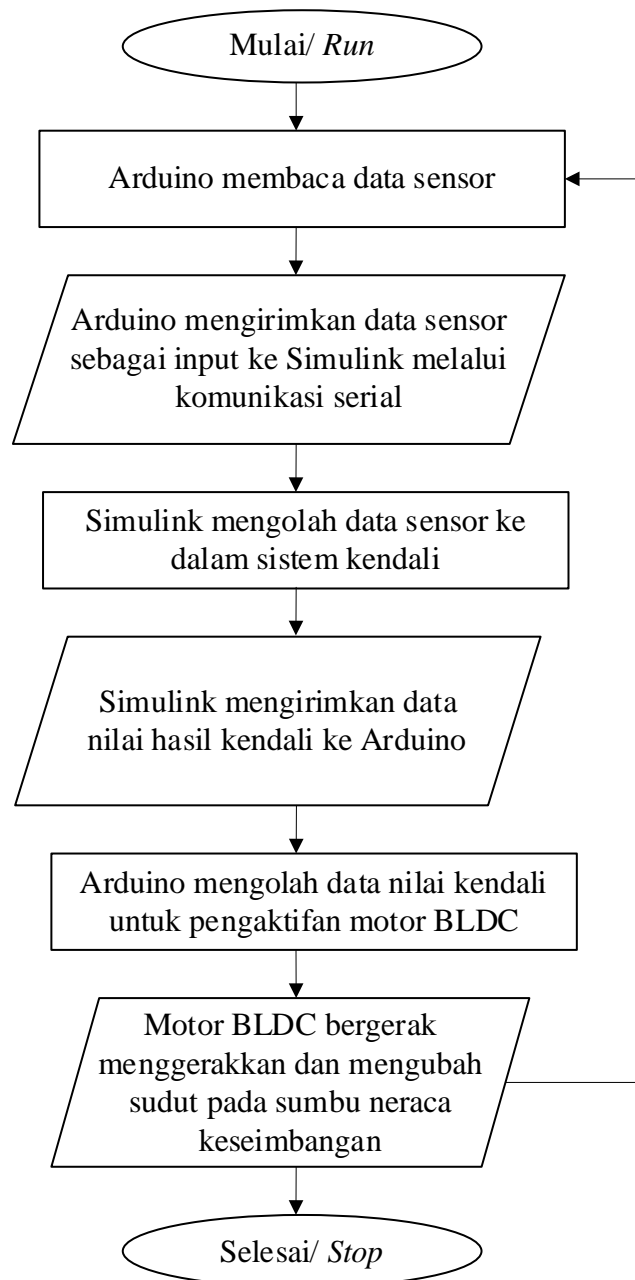


Gambar 3.16 Rule Editor ANFIS

Tabel 3.3 Aturan ANFIS

| <i>de/ e</i> | in1mf1 | in1mf2 | in1mf3 | in1mf4 | in1mf5 |
|---------------|---------------|---------------|---------------|---------------|---------------|
| in2mf1 | out1mf1 | out1mf6 | out1mf11 | out1mf16 | out1mf21 |
| in2mf2 | out1mf2 | out1mf7 | out1mf12 | out1mf17 | out1mf22 |
| in2mf3 | out1mf3 | out1mf8 | out1mf13 | out1mf18 | out1mf23 |
| in2mf4 | out1mf4 | out1mf9 | out1mf14 | out1mf19 | out1mf24 |
| in2mf5 | out1mf5 | out1mf10 | out1mf15 | out1mf20 | out1mf25 |

3.4.4 Perancangan Integrasi Simulink dengan *Controller*

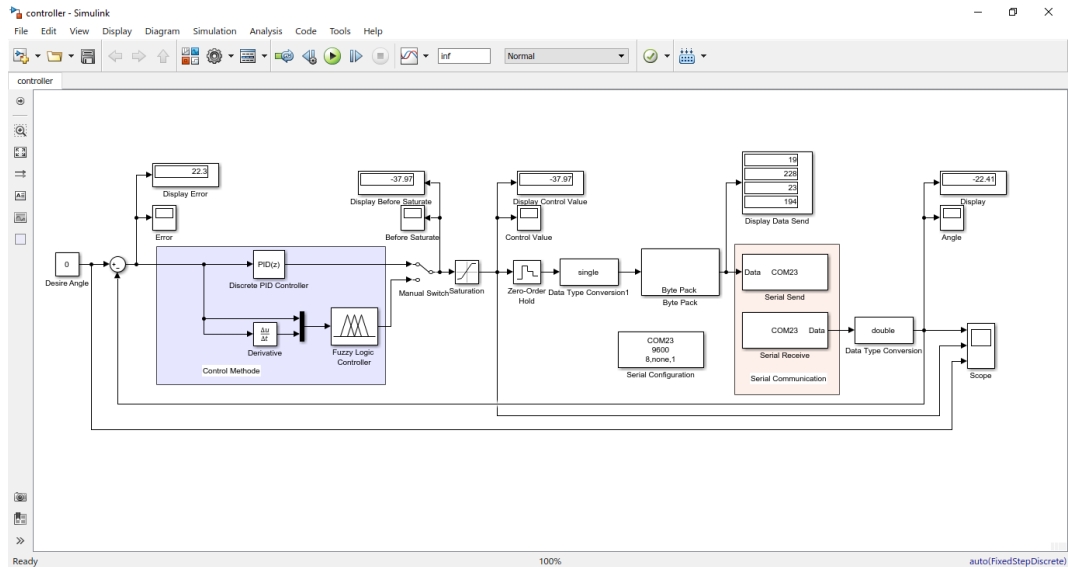


Gambar 3.17 Diagram Alir Integrasi Simulink dengan Arduino

Setelah masing-masing pemrograman pada Arduino IDE dan Simulink selesai, maka perlu ditambahkan *coding* komunikasi serial pada masing-masing pemrograman.

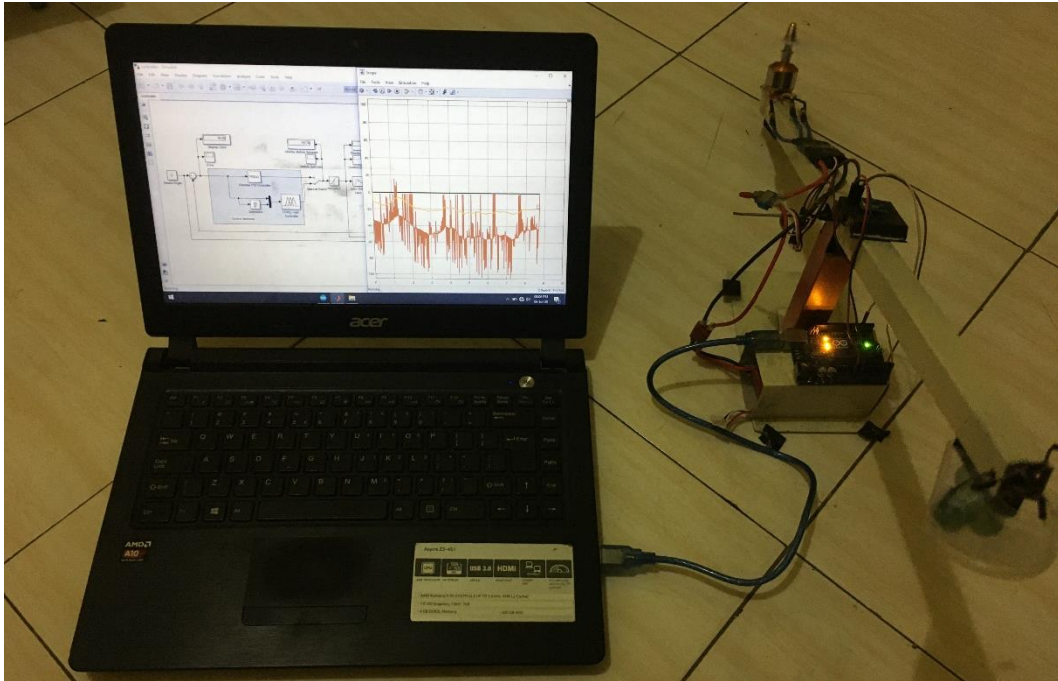
BAB IV

HASIL DAN PEMBAHASAN



Gambar 4.1 Interface pada Simulink

Sistem Simulink dimulai ketika *serial receive* COM23 menerima data pembacaan sensor dari Arduino, kemudian data diubah dengan blok *data type converter* menjadi jenis data *double*. Selisih nilai acuan dengan pembacaan sensor adalah *error* yang akan masuk kedalam sistem kendali. Pada area biru terdapat dua blok kendali, yaitu *PID Controller* dan *Fuzzy Logic Controller*. Sistem ini mengendalikan sinyal digital sehingga pemilihan blok blok harus disesuaikan. Untuk pemrograman *PID Controller* hanya memiliki satu *input* yaitu nilai *error*. Sedangkan untuk *Fuzzy Logic Controller* memiliki dua input, yaitu nilai *error* dan *derivative error*. Terdapat *manual switch* yang digunakan untuk mengubah metode kendali antara metode PID dan FLC. Dalam blok *Fuzzy Logic Controller* file FIS yang telah dibuat sebelumnya akan dipanggil dari *workspace*. Pada blok *Fuzzy Logic Controller*, metode FLC dan ANFIS dapat diubah sesuai nama FIS yang telah dibuat. Nilai kendali dibatasi oleh blok *saturation* ± 150 , hal tersebut dilakukan untuk membatasi nilai pencilan dan menyesuaikan pemrograman pada Arduino. Selanjutnya nilai kendali akan diubah menjadi data *single* dan *byte* untuk dikirim kembali ke Arduino melalui *serial send* COM23. *Display* dan *scope* dapat ditambahkan dimanapun untuk *monitoring* data setiap *step*.



Gambar 4.2 Hasil Pembuatan Sistem Kendali Posisi VTOL dengan *Interface* Simulink

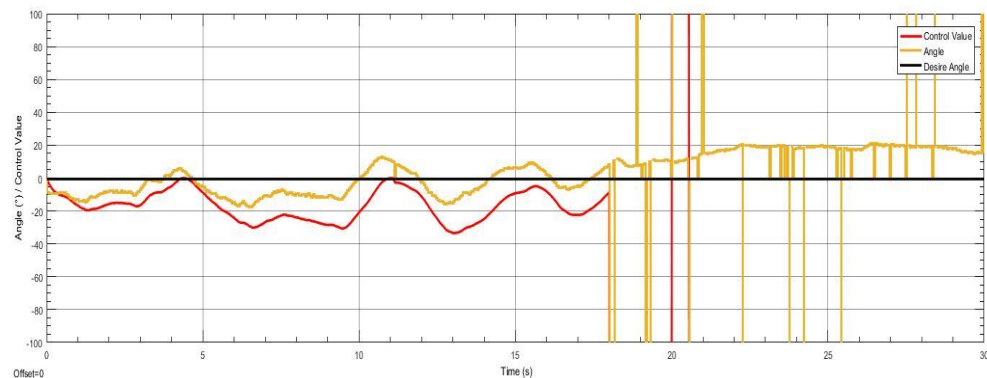
Pemrograman yang telah dilakukan pada masing-masing sistem dapat berjalan dengan baik, pembacaan sensor dan pengiriman data sensor dapat dilihat *display* dan *scope* di Simulink. Sistem metode kendali juga dapat memberikan nilai kendali dan mengirimkan kembali ke Arduino untuk mengendalikan kecepatan motor. Arduino yang telah diprogram pada Arduino IDE dengan program terlampir tidak perlu diprogram ulang ketika mengganti metode kendali pada Simulink. Hal tersebut merupakan salah satu kelebihan dari sistem integrasi ini, karena efisiensi yang dihasilkan lebih baik dibandingkan ketika harus membuat *list coding* panjang untuk satu metode kendali. Berikut hasil dari masing-masing metode kendali dengan beberapa pengujian dengan memvariasikan posisi awal sudut -15° , 0° , dan 15° serta pembebanan dengan memvariasikan *load* 5 kelereng (27,5 gram), 6 kelereng (33 gram), dan 7 kelereng (38,5 gram). dilihat dari *scope* selama 30 detik.

4.1 Hasil Pengujian Pengaruh Posisi Awal

Pengujian pengaruh posisi awal dilakukan dengan memvariasikan posisi awal dengan nilai -15° , 0° , dan 15° . Nilai posisi awal akan diuji pada masing-masing metode kendali dan akan dianalisis pengaruh posisi awal terhadap sistem.

4.1.1 Posisi awal -15°

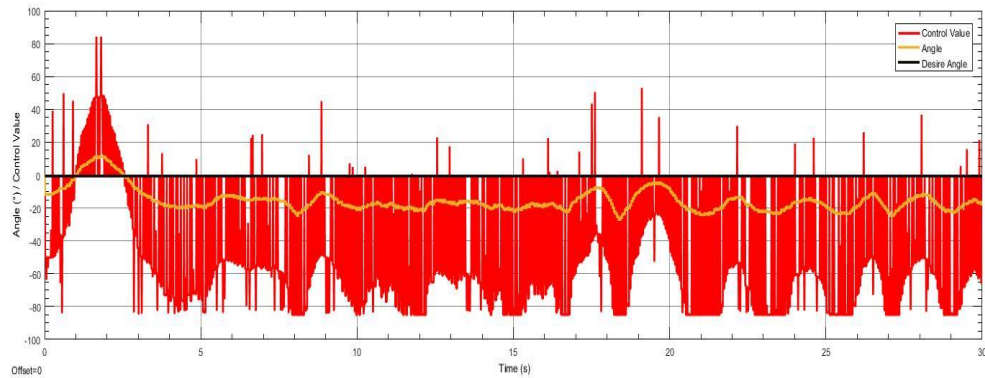
- *PID Controller*



Gambar 4.3 Hasil Pengujian Posisi Awal -15° dengan *PID Controller*

Pada gambar 4.3, grafik pembacaan sudut bergelombang naik turun, hal tersebut menunjukkan sistem berusaha mencapai titik acuan. Sistem menambah nilai 5 kali untuk mendekati titik acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Pada detik ke-18 nilai kendali menyentuh batas bawah saturasi yang menyebabkan nilai kendali akan konstan di angka -100. Pada detik ke 20, nilai kendali naik dan pada detik ke-21 nilai kendali menyentuh batas atas saturasi yang menyebabkan nilai kendali akan konstan di angka 100. Hal tersebut mungkin terjadi karena ada nilai-nilai *error* yang mungkin muncul dari pembacaan sensor.

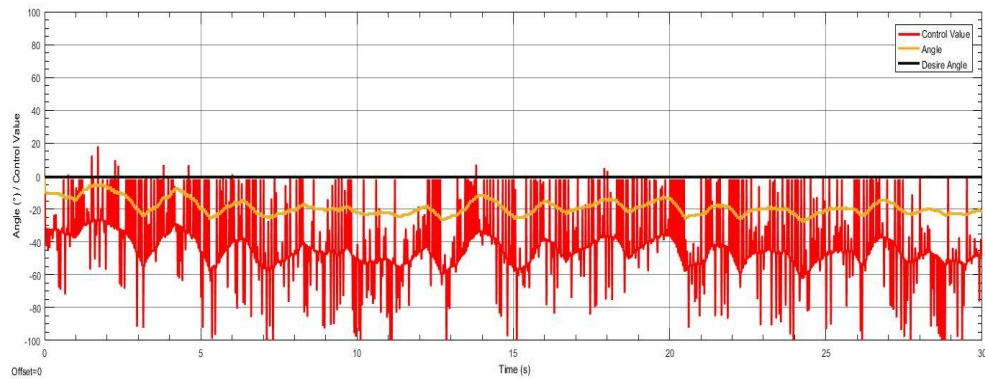
- FLC



Gambar 4.4 Hasil Pengujian Posisi Awal -15° dengan FLC

Pada gambar 4.4, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai 14 kali untuk mendekati titik acuan dan berosilasi sekali melewati titik acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Grafik nilai kendali pada kendali menggunakan *fuzzy* lebih banyak daripada PID karena perbedaan respons yang lebih cepat. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

- ANFIS



Gambar 4.5 Hasil Pengujian Posisi Awal -15° dengan ANFIS

Pada gambar 4.5, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai 16 kali untuk mendekati titik acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

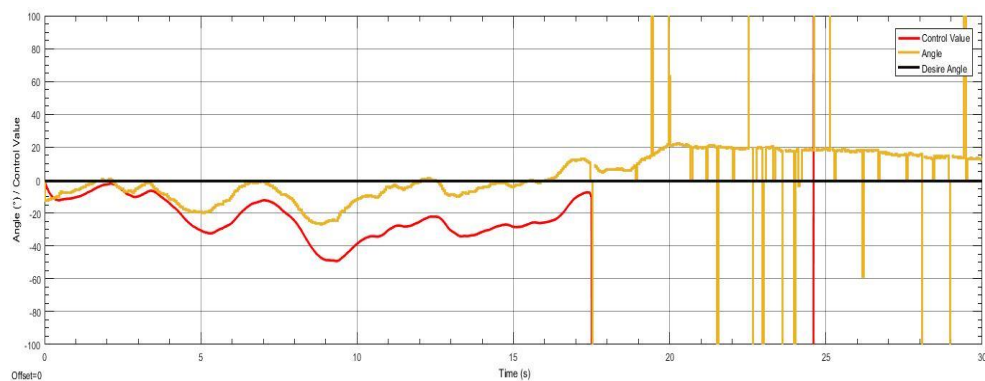
Tabel 4.1 Perbandingan Kinerja *Controller* dengan Posisi Awal -15°

| <i>Controller/Characteristic</i> | PID | FLC | ANFIS |
|----------------------------------|------------|------------|--------------|
| <i>Rise Time (s)</i> | 3,21 | 0,96 | 1,81 |
| <i>Settling Time (s)</i> | 17,31 | 5,84 | 5,56 |
| <i>Overshoot (°)</i> | 21 | 11 | 0 |
| <i>Undershoot (°)</i> | 15 | 25 | 25 |
| <i>Integral Squared Error</i> | 4,965e+03 | 8,367e+03 | 1,210e+04 |

Perbandingan kinerja *controller* dengan posisi awal -15° disajikan dalam tabel 4.1. *Rise time* tercepat diperoleh menggunakan FLC dengan 0,96 detik, *settling time* tercepat diperoleh menggunakan ANFIS dengan 5,56 detik, *overshoot* terkecil diperoleh menggunakan ANFIS (tidak ada *overshoot*), *undershoot* terdekat dengan nilai acuan diperoleh menggunakan PID dengan 15° , dan *integral squared error* terkecil diperoleh menggunakan PID dengan 4,965e+03.

4.1.2 Posisi awal 0°

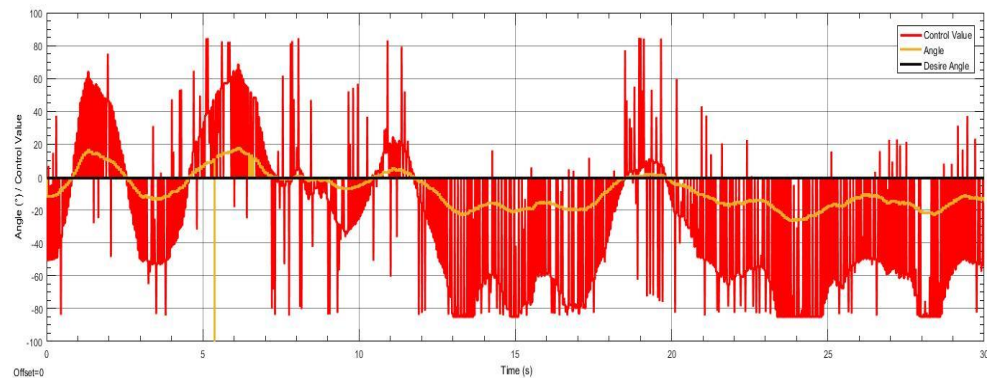
- *PID Controller*

Gambar 4.6 Hasil Pengujian Posisi Awal 0° dengan PID

Pada gambar 4.6, grafik pembacaan sudut bergelombang naik turun, hal tersebut menunjukkan sistem berusaha mencapai titik acuan. Sistem menambah nilai 9 kali untuk mendekati titik acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Pada detik ke-16.5 nilai kendali menyentuh batas bawah saturasi yang menyebabkan nilai kendali

akan konstan di angka -100. Pada detik ke-24.5 nilai kendali menyentuh batas atas saturasi yang menyebabkan nilai kendali akan konstan di angka 100. Hal tersebut mungkin terjadi karena ada nilai-nilai *error* yang mungkin muncul dari pembacaan sensor.

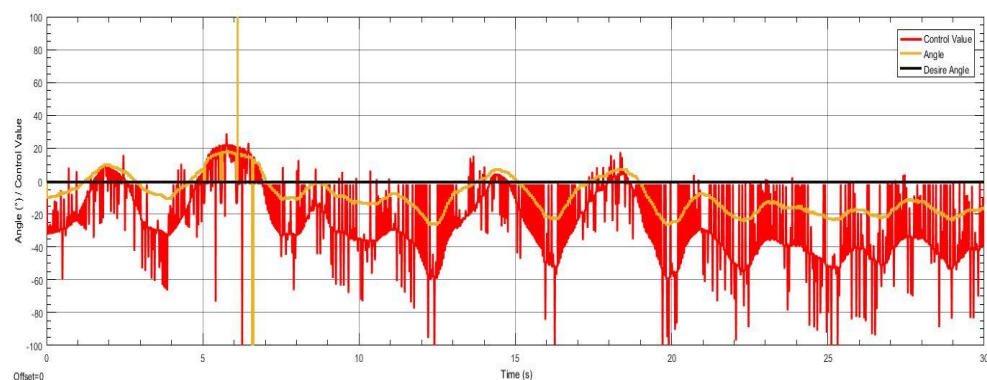
- FLC



Gambar 4.7 Hasil Pengujian Posisi Awal 0° dengan FLC

Pada gambar 4.7, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai 9 kali untuk mendekati titik acuan dan berosilasi empat kali melewati titik acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Grafik nilai kendali pada kendali menggunakan *fuzzy* lebih banyak daripada PID karena perbedaan respons yang lebih cepat. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

- ANFIS



Gambar 4.8 Hasil Pengujian Posisi Awal 0° dengan ANFIS

Pada gambar 4.8, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai 10 kali untuk mendekati titik acuan dan berosilasi empat kali melewati titik acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

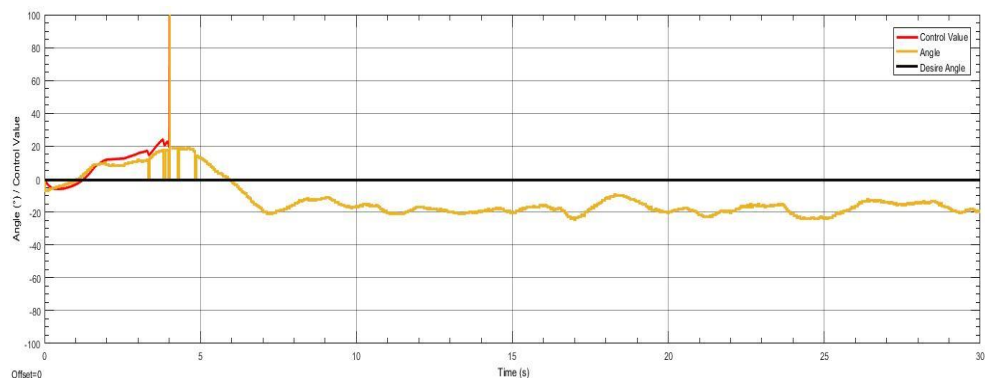
Tabel 4.2 Perbandingan Kinerja *Controller* dengan Posisi Awal 0°

| <i>Controller/Characteristic</i> | PID | FLC | ANFIS |
|---|------------|------------|--------------|
| <i>Rise Time (s)</i> | 1,56 | 0,81 | 1,16 |
| <i>Settling Time (s)</i> | 16,01 | 14,11 | 10,35 |
| <i>Overshoot (°)</i> | 21 | 18 | 18 |
| <i>Undershoot (°)</i> | 25 | 25 | 25 |
| <i>Integral Squared Error</i> | 2,546e+12 | 5,298e+03 | 4,137e+03 |

Perbandingan kinerja *controller* dengan posisi awal 0° disajikan dalam tabel 4.2. *Rise time* tercepat diperoleh menggunakan FLC dengan 0,81 detik, *settling time* tercepat diperoleh menggunakan ANFIS dengan 10,35 detik, *overshoot* terkecil diperoleh menggunakan FLC dan ANFIS dengan 18° , *undershoot* terdekat dengan nilai acuan sama pada ketiga kendali dan *integral squared error* terkecil diperoleh menggunakan FLC dengan $5,298e+03$.

4.1.3 Posisi awal 15°

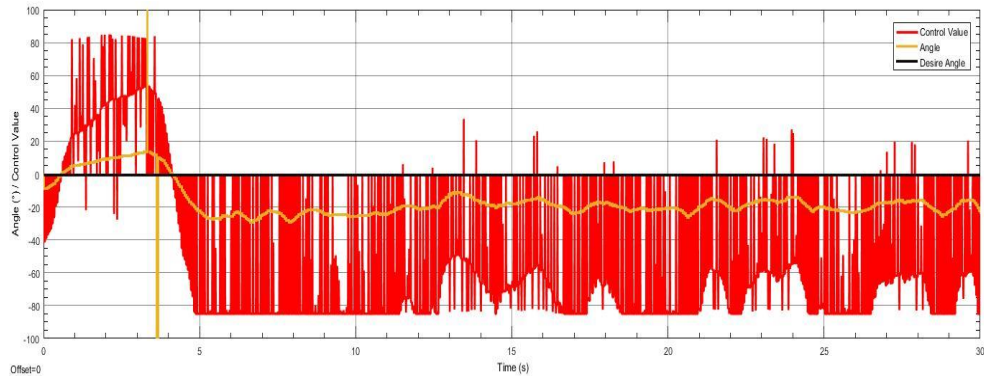
- *PID Controller*



Gambar 4.9 Hasil Pengujian Posisi Awal 15° dengan *PID Controller*

Pada gambar 4.9, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai kendali 3 kali tetapi menjauhi titik acuan. Nilai kendali pada detik ke-4 menyentuh batas atas saturasi yang menyebabkan nilai kendali akan konstan di angka 100 dan tidak menurun sampai detik ke-30 selesai.

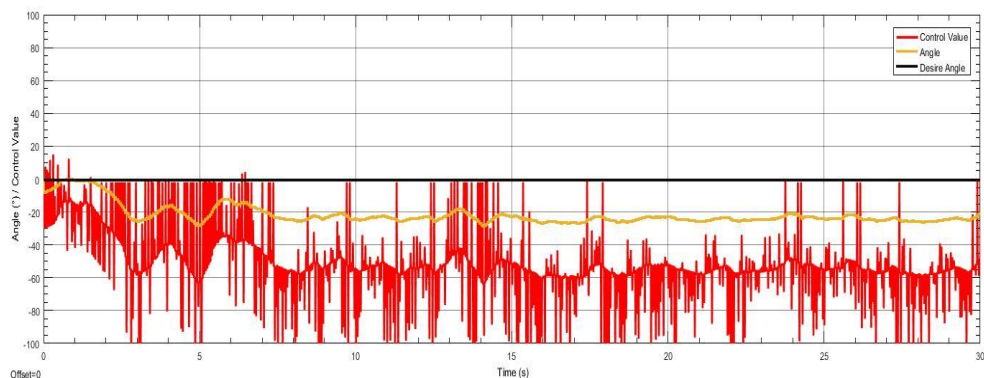
- FLC



Gambar 4.10 Hasil Pengujian Posisi Awal 15° dengan FLC

Pada gambar 4.10, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai 13 kali untuk mendekati titik acuan dan beresilasi satu kali melewati titik acuan. Nilai kendali mengikuti naik-turun grafik pembacaan sensor. Grafik nilai kendali pada kendali menggunakan *fuzzy* lebih banyak daripada PID karena perbedaan respons yang lebih cepat. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

- ANFIS



Gambar 4.11 Hasil Pengujian Posisi Awal 15° dengan ANFIS

Pada gambar 4.11, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai 14 kali untuk mendekati titik acuan, tetapi nilai kendali tidak bertambah secara besar. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

Tabel 4.3 Perbandingan Kinerja *Controller* dengan Posisi Awal 15°

| <i>Controller/Characteristic</i> | PID | FLC | ANFIS |
|---|------------|------------|--------------|
| <i>Rise Time (s)</i> | 0,96 | 0,51 | 0,56 |
| <i>Settling Time (s)</i> | 6,96 | 11,31 | 7,26 |
| <i>Overshoot (°)</i> | 19 | 15 | 0 |
| <i>Undershoot (°)</i> | 24 | 30 | 28 |
| <i>Integral Squared Error</i> | 1,008e+47 | 1,100e+04 | 1,944e+04 |

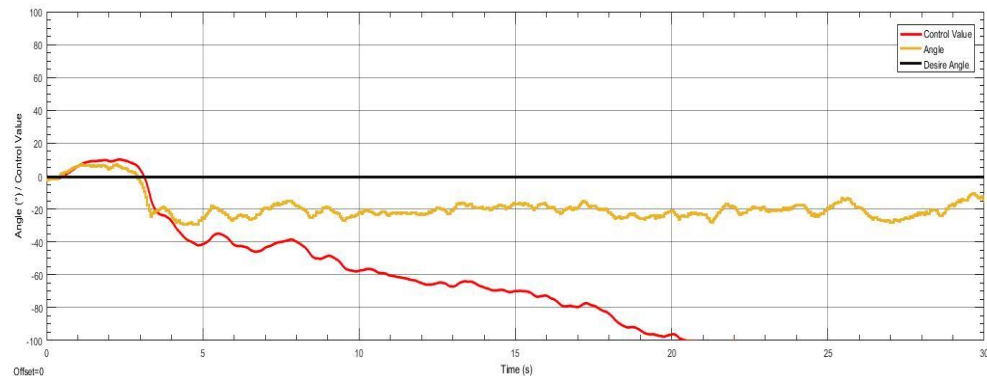
Perbandingan kinerja *controller* dengan posisi awal 15° disajikan dalam tabel 4.3. *Rise time* tercepat diperoleh menggunakan FLC dengan 0,51 detik, *settling time* tercepat diperoleh menggunakan PID dengan 6,96 detik, *overshoot* terkecil diperoleh menggunakan ANFIS (tidak ada *overshoot*), *undershoot* terdekat dengan nilai acuan diperoleh menggunakan PID 24°, dan *integral squared error* terkecil diperoleh menggunakan FLC dengan 1,100e+04.

4.2 Hasil Pengujian Pengaruh Pembebanan

Pengujian pengaruh pembebanan dilakukan dengan memvariasikan muatan (*load*) dalam sistem. *Load* yang dipakai adalah kelereng yang digantung dalam wadah di ujung lain dari motor BLDC. Satu kelereng memiliki berat rerata 5,5 gram. Percobaan dilakukan dengan *load* 5 kelereng (27,5 gram), 6 kelereng (33 gram), dan 7 kelereng (38,5 gram).

4.2.1 Load 27,5 gram

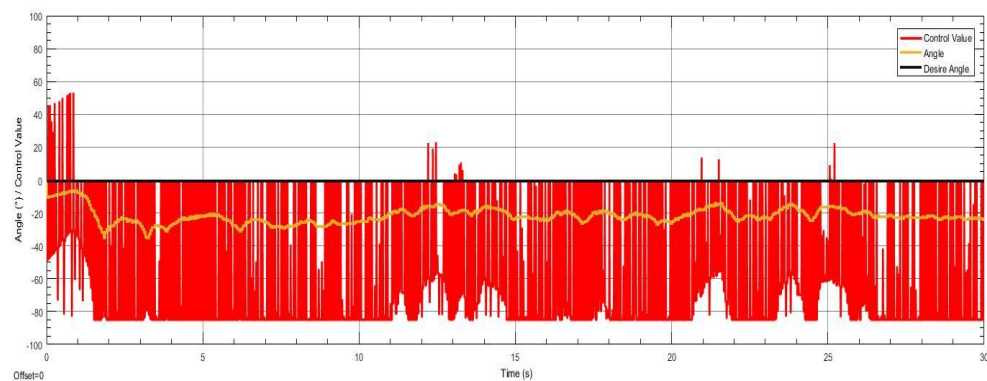
- PID Controller



Gambar 4.12 Hasil Pengujian *Load* 27,5 gram dengan PID Controller

Pada gambar 4.12, grafik pembacaan sudut bergelombang naik turun, hal tersebut menunjukkan sistem berusaha mencapai titik acuan. Sistem menambah nilai kendali 8 kali mendekati titik acuan. Pada percobaan ini sistem tidak menyentuh nilai saturasi secara konstan, berbeda dengan tiga percobaan sebelumnya. Namun nilai kendali pada sistem ini semakin lama semakin menurun mendekati nilai saturasi.

- FLC

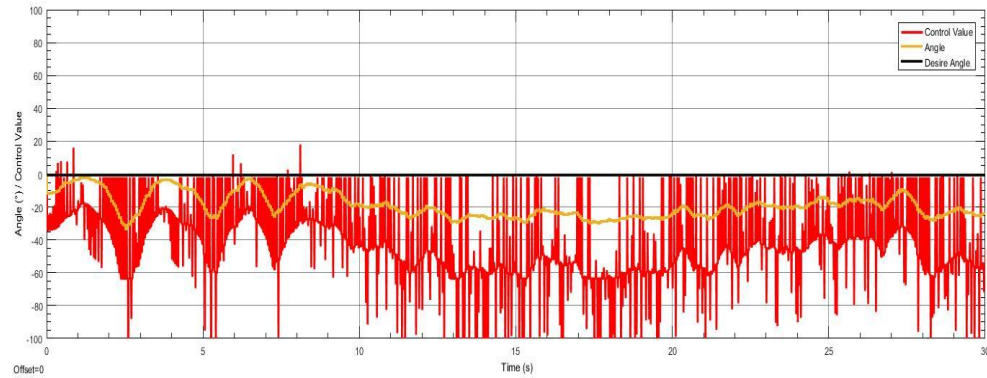


Gambar 4.13 Hasil Pengujian *Load* 27,5 gram dengan FLC

Pada gambar 4.13, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai lebih dari 18 kali untuk mendekati titik acuan. Nilai kendali mengikuti naik-turun grafik pembacaan sensor. Grafik nilai kendali pada kendali menggunakan *fuzzy* lebih banyak daripada PID karena

perbedaan respons yang lebih cepat. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

- ANFIS



Gambar 4.14 Hasil Pengujian *Load* 27,5 gram dengan ANFIS

Pada gambar 4.14, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai 16 kali untuk mendekati titik acuan. Empat kali beresilasi hampir menyentuh nilai acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

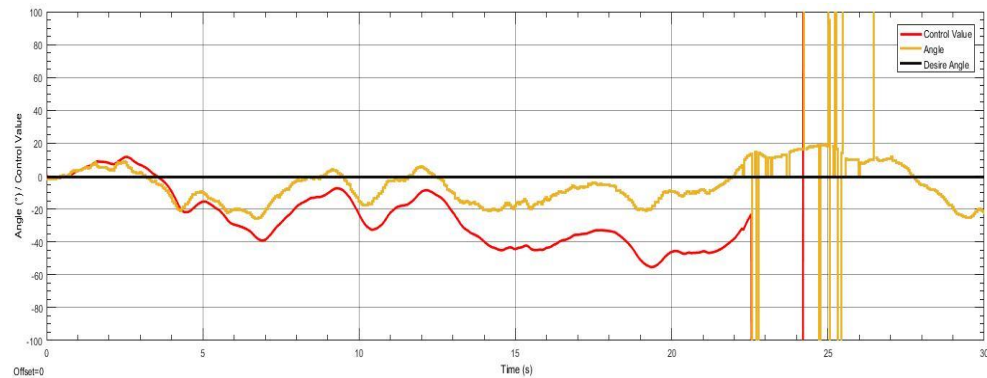
Tabel 4.4 Perbandingan Kinerja *Controller* dengan *Load* 27,5 gram

| <i>Controller/Characteristic</i> | PID | FLC | ANFIS |
|---|------------|------------|--------------|
| <i>Rise Time (s)</i> | 0,36 | 0,56 | 1,16 |
| <i>Settling Time (s)</i> | 3,26 | 10,91 | 10,86 |
| <i>Overshoot (°)</i> | 7 | 0 | 0 |
| <i>Undershoot (°)</i> | 30 | 31 | 31 |
| <i>Integral Squared Error</i> | 1,226e+04 | 1,452e+04 | 1,229e+04 |

Perbandingan kinerja *controller* dengan *load* 27,5 gram disajikan dalam tabel 4.4. *Rise time* tercepat diperoleh PID dengan 0,36 detik, *settling time* tercepat menggunakan PID dengan 3,26 detik, *overshoot* terkecil diperoleh menggunakan FLC dan ANFIS (tidak ada *overshoot*), serta *undershoot* terdekat dengan nilai acuan diperoleh menggunakan PID 30°, *integral squared error* terkecil diperoleh menggunakan PID dengan 1,226e+04.

4.2.2 Load 33 gram

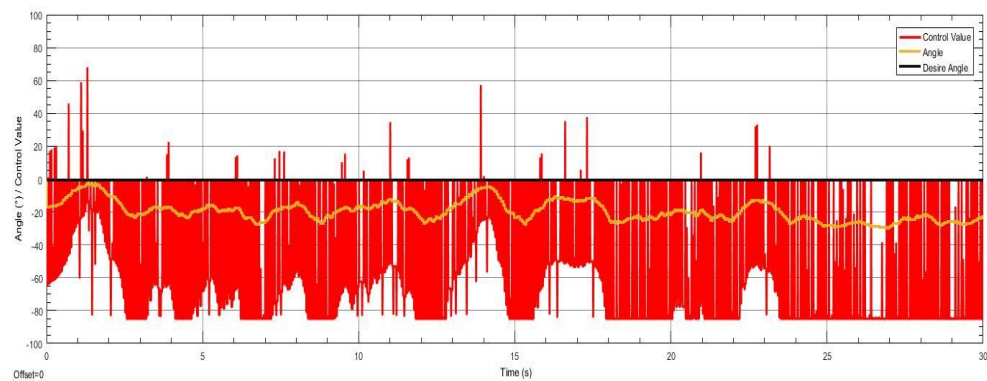
- *PID Controller*



Gambar 4.15 Hasil Pengujian *Load* 33 gram dengan *PID Controller*

Pada gambar 4.15, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai kendali 6 kali mendekati titik acuan dan 4 kali melewati sudut acuan serta sekali hampir menyentuh sudut acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Pada detik ke-22.5 nilai kendali menyentuh batas bawah saturasi dan pada detik ke-24 nilai kendali menyentuh batas atas saturasi. Hal tersebut mungkin terjadi karena ada nilai-nilai *error* yang mungkin muncul dari pembacaan sensor.

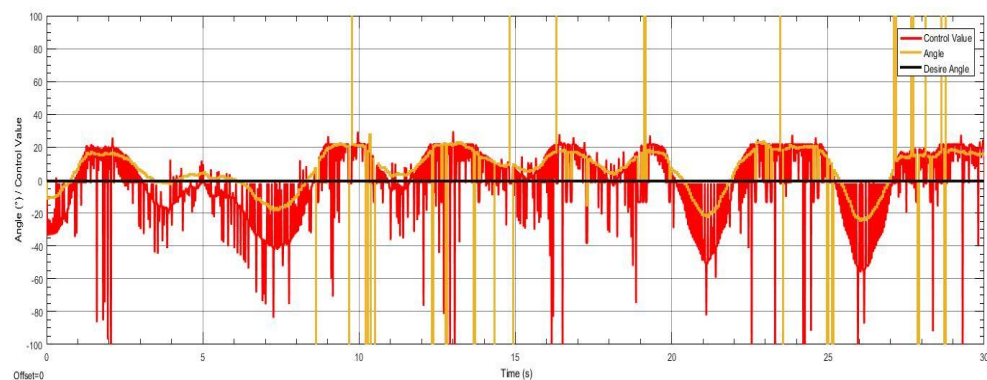
- *FLC*



Gambar 4.16 Hasil Pengujian *Load* 33 gram dengan *FLC*

Pada gambar 4.16, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai lebih dari 15 kali untuk mendekati titik acuan dan tidak pernah melewatinya. Nilai kendali mengikuti naik-turun grafik pembacaan sensor. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

- ANFIS



Gambar 4.17 Hasil Pengujian *Load* 33 gram dengan ANFIS

Pada gambar 4.17, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai 8 kali dan mengurangi 7 kali untuk mendekati titik acuan. Sistem berosilasi pada sudut acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi.

Tabel 4.5 Perbandingan Kinerja *Controller* dengan *Load* 33 gram

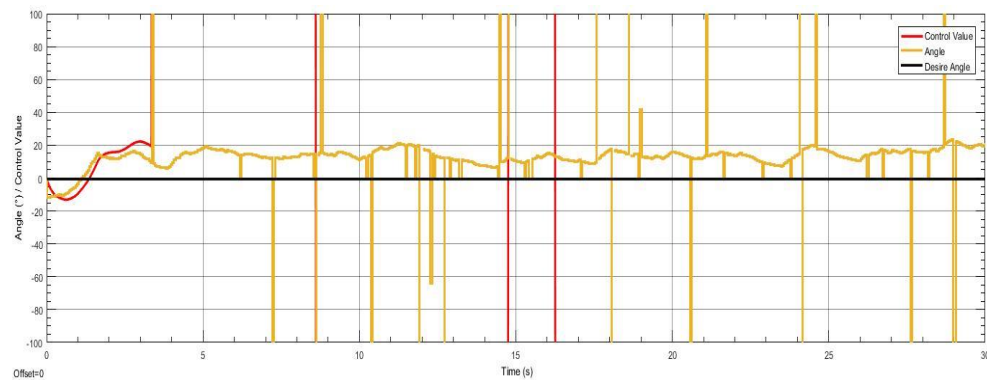
| <i>Controller/Characteristic</i> | PID | FLC | ANFIS |
|---|------------|------------|--------------|
| <i>Rise Time (s)</i> | 0,41 | 1,26 | 0,71 |
| <i>Settling Time (s)</i> | 7,46 | 3,01 | 8,56 |
| <i>Overshoot (°)</i> | 20 | 0 | 22 |
| <i>Undershoot (°)</i> | 25 | 25 | 15 |
| <i>Integral Squared Error</i> | 7,090e+16 | 1,230e+04 | 4,539e+03 |

Perbandingan kinerja *controller* dengan *load* 33 gram disajikan dalam tabel 4.5. *Rise time* tercepat diperoleh menggunakan PID dengan 0,41 detik, *settling time* tercepat diperoleh menggunakan FLC dengan 3,01 detik,

overshoot terkecil diperoleh menggunakan FLC (tidak ada *overshoot*), *undershoot* terdekat dengan nilai acuan diperoleh menggunakan ANFIS 15°, dan *integral squared error* terkecil diperoleh menggunakan FLC dengan 1,230e+04.

4.2.3 Load 38,5 gram

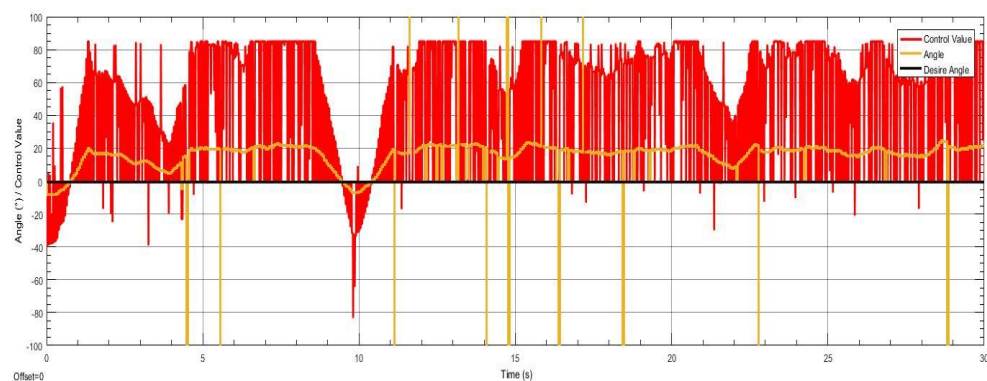
- PID Controller



Gambar 4.18 Hasil Pengujian *Load* 38,5 gram dengan PID Controller

Pada gambar 4.18, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai kendali satu kali dan melebihi titik acuan. Pada detik ke-3,5 nilai kendali menyentuh batas atas saturasi. Hal tersebut mungkin terjadi karena ada nilai-nilai *error* yang mungkin muncul dari pembacaan sensor serta *load* yang terlalu berat. Sehingga nilai minimum PWM tidak mampu membuat sistem seimbang kembali.

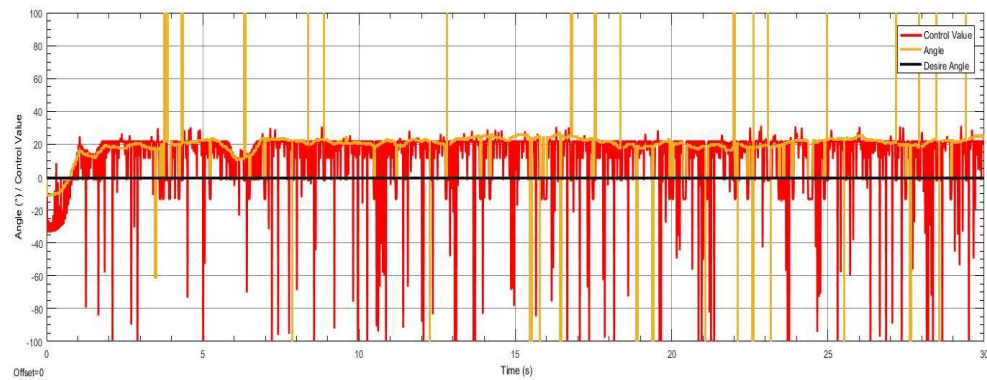
- FLC



Gambar 4.19 Hasil Pengujian *Load* 38,5 gram dengan FLC

Pada gambar 4.19, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai lebih dari satu kali dan melebihi nilai acuan. Serta turun 7 kali untuk mendekati titik acuan Nilai kendali mengikuti naik-turun grafik pembacaan sensor. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi. Tetapi karena *load* yang terlalu berat sehingga nilai minimum PWM tidak mampu membuat sistem seimbang kembali.

- ANFIS



Gambar 4.20 Hasil Pengujian *Load* 38,5 gram dengan ANFIS

Pada gambar 4.20, grafik pembacaan sudut bergelombang naik turun. Sistem menambah nilai satu kali dan mengurangi tiga kali untuk mendekati titik acuan. Nilai kendali selalu mengikuti naik-turun grafik pembacaan sensor. Pada percobaan kendali ini sistem tidak pernah menyentuh nilai saturasi. Tetapi karena *load* yang terlalu berat. Sehingga nilai minimum PWM tidak mampu membuat sistem seimbang kembali.

Tabel 4.6 Perbandingan Kinerja *Controller* dengan *Load* 38,5 gram

| <i>Controller/Characteristic</i> | PID | FLC | ANFIS |
|---|------------|------------|--------------|
| <i>Rise Time (s)</i> | 1,11 | 0,71 | 0,71 |
| <i>Settling Time (s)</i> | 2,95 | 4,71 | 2,7 |
| <i>Overshoot (°)</i> | 20 | 21 | 23 |
| <i>Undershoot (°)</i> | 0 | 5 | 0 |
| <i>Integral Squared Error</i> | 1,351e+20 | 1,404e+19 | 6,168e+18 |

Perbandingan kinerja *controller* dengan *load* 38,5 gram disajikan dalam tabel 4.6. *Rise time* tercepat diperoleh menggunakan FLC dan ANFIS dengan 0,71 detik, *settling time* tercepat diperoleh menggunakan ANFIS dengan 2,7 detik, *overshoot* terkecil diperoleh menggunakan PID dengan 20°, *undershoot* terdekat dengan nilai acuan diperoleh menggunakan PID dan ANFIS (tidak ada *undershoot*), dan *integral square error* terkecil diperoleh menggunakan FLC dengan 1,404e+19.

4.3 Perbandingan Kinerja Controller

Perbandingan kinerja *controller* pada pengujian yang telah dilakukan dibandingkan dengan menghitung rata-rata pada masing-masing karakteristik. Pada tabel 4.7 adalah nilai rata-rata dari tabel perbandingan kinerja *controller* dengan posisi awal -15°, 0°, 15° (tabel 4.1, 4.2, dan 4.3). Pada tabel 4.8 adalah nilai rata-rata dari tabel perbandingan kinerja *controller* dengan pembebanan 27,5 gram, 33 gram, dan 38,5 gram (tabel 4.4, 4.5, dan 4.6).

Tabel 4.7 Perbandingan Kinerja Controller Pengujian Posisi Awal

| Controller/Characteristic | PID | FLC | ANFIS |
|--------------------------------------|------------|------------|--------------|
| <i>Rise Time (s)</i> | 1,91 | 0,76 | 1,17 |
| <i>Settling Time (s)</i> | 13,42 | 10,42 | 7,72 |
| <i>Overshoot (°)</i> | 20,33 | 14,67 | 6 |
| <i>Undershoot (°)</i> | 21,3 | 26,67 | 26 |
| <i>Integral Squared Error</i> | 3,36e+46 | 8,221e+4 | 1,1892e+4 |

Tabel 4.8 Perbandingan Kinerja Controller Pengujian Pembebanan

| Controller/Characteristic | PID | FLC | ANFIS |
|--------------------------------------|------------|------------|--------------|
| <i>Rise Time (s)</i> | 0,62 | 0,84 | 0,86 |
| <i>Settling Time (s)</i> | 4,55 | 6,21 | 7,37 |
| <i>Overshoot (°)</i> | 15,67 | 7 | 15 |
| <i>Undershoot (°)</i> | 18,33 | 20,33 | 15,33 |
| <i>Integral Squared Error</i> | 4,51e+19 | 4,68e+18 | 2,06e+18 |

BAB V

PENUTUP

5.1 Kesimpulan

Dari penelitian yang telah dilakukan, diperoleh beberapa kesimpulan sebagai berikut:

1. Sistem kendali posisi sederhana pada VTOL dapat dirancang dengan *hardware* neraca keseimbangan serta komponen sistem: Arduino Uno sebagai *controller*, MPU6050 *sensor module* sebagai sensor posisi, ESC sebagai *driver motor*, BLDC A2212 sebagai *actuator*, dan baterai Li-Po 3S sebagai *supply*.
2. Hasil pengujian variasi posisi awal pada penelitian ini memperoleh hasil terbaik pada percobaan dengan posisi awal 0° metode ANFIS dengan ISE $4,137\text{e}+03$. Serta hasil pengujian variasi *load* memperoleh hasil terbaik pada pembebanan dengan *load* enam kelereng (33 gram) metode ANFIS dengan ISE $4,539\text{e}+03$.
3. Hasil perbandingan kinerja *controller* pada pengujian yang telah dilakukan adalah metode ANFIS memiliki ISE terkecil pengujian variasi posisi awal $1,1892\text{e}+4$ dibandingkan metode PID $3,36\text{e}+46$ dan metode FLC $8,221\text{e}+4$ serta $2,06\text{e}+18$ untuk pengujian variasi *load* dibandingkan metode PID $4,51\text{e}+19$ dan metode FLC $4,68\text{e}+18$.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, ada beberapa saran untuk pelaksanaan penelitian selanjutnya agar mendapatkan hasil yang lebih baik, diantaranya:

1. Menambahkan motor BLDC pada sisi yang lainnya dan memperbaiki desain *hardware* agar lebih baik.
2. Menggunakan *supply* yang konstan, agar setiap metode dan pengujian yang diterapkan pada sistem dapat bekerja dengan nilai *supply* yang sama.

DAFTAR PUSTAKA

- [1] P. Castillo, R. Lozano dan A. E. Dzul, *Modelling and Control of Mini-Flying Machines*, Compiegne, France: Springer, 2005.
- [2] H. Maghfiroh, "Optimasi Sistem Kendali PID dengan Double Tuning Dalam Implementasi Pengendalian Kecepatan Motor DC Berbasis PLC," Skripsi, Universitas Gajah Mada, Yogyakarta, 2013.
- [3] K. Wardana, "Tutor Keren - Mengapa Menggunakan Simulink untuk Memprogram Mikrokontroler ?," CV. Narayana Instrument, 21 November 2015. [Online]. Available: <https://tutorkeren.com/artikel/mengapa-menggunakan-simulink-untuk-memprogram-mikrokontroler.htm>. [Diakses 27 Juni 2020].
- [4] "MathWorks - Simulink," The MathWorks Inc., 2020. [Online]. Available: <https://in.mathworks.com/products/simulink.html>. [Diakses 27 Juni 2020].
- [5] L. Plummer, "WIRED - What is VTOL? A beginner's guide to vertical take-off and landing technology," Condé Nast Britain, 28 April 2017. [Online]. Available: <https://www.wired.co.uk/article/vtol-vertical-take-off-landing-explained>. [Diakses 29 Juni 2020].
- [6] "Unmanned Systems Technology - VTOL Drones & Multirotor UAVs," EchoBlue Ltd, 2019. [Online]. Available: <https://www.unmannedsystemstechnology.com/category/supplier-directory/platforms/vtoluav/>. [Diakses 2 Juli 2020].
- [7] Aydoğdu, M. Çunkaş dan Omer, "Realization of Fuzzy Logic Controlled Brushless DC Motor Drives Using Matlab/Simulink," *Mathematical and Computational Applications*, vol. 15, no. 2, pp. 218-229, 2010.
- [8] M. Rabah, A. Rohan, Y.-J. Han dan S.-H. and Kim, "Design of Fuzzy-PID Controller for Quadcopter Trajectory-Tracking," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 18, no. 3, pp. 204-213, 2018.
- [9] Burhanuddin, M. Malik dan Aan, "Desain Model Fuzzy Control UAV Berbasis Matlab/Simulink pada Arduino Flight Controller," *JURNAL ENGINE*, vol. 2, no. 2, 2018.

- [10] J. G. Tíscar, “WE CHOOSE THE MOON - Arduino + Matlab / Simulink: PID controller,” procrastination™, 21 July 2011. [Online]. Available: <https://wechoosethemoon.es/2011/07/21/arduino-matlab-simulink-controlador-pid/>. [Diakses 29 Juni 2020].
- [11] R. S. Zugasti, “Mechatronics Engineering,” [Online]. Available: <http://mechatronics-rsz.blogspot.com/p/pvtol.html>. [Diakses 29 Juni 2020].
- [12] “ELECTRONOBS - PID Control With Arduino,” [Online]. Available: http://www.electrionoobs.com/eng_robotica_tut6.php. [Diakses 29 Juni 2020].
- [13] “Arduino - ARDUINO UNO REV3,” Arduino, 2020. [Online]. Available: <https://store.arduino.cc/usa/arduino-uno-rev3>. [Diakses 2 Juli 2020].
- [14] “Arduino - Serial,” Arduino, 15 June 2020. [Online]. Available: <https://www.arduino.cc/reference/en/language/functions/communication/serial/>. [Diakses 2 Juli 2020].
- [15] “Arduino - Master Writer/Slave Receiver,” Arduino, 17 May 2018. [Online]. Available: <https://www.arduino.cc/en/Tutorial/MasterWriter>. [Diakses 3 Juli 2020].
- [16] T. Hirzel, “Arduino - PWM,” Arduino, 2020. [Online]. Available: <https://www.arduino.cc/en/tutorial/PWM>. [Diakses 3 Juli 2020].
- [17] “ElectronicWings - MPU6050 (Gyroscope + Accelerometer + Temperature) Sensor Module,” ElectronicWings, 2020. [Online]. Available: <https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module>. [Diakses 2 Juli 2020].
- [18] “Electronics Project Focus - Introduction To Electronic Speed Control (ESC) Working and Applications,” Electronics Project Focus, 2013. [Online]. Available: <https://www.elprocus.com/electronic-speed-control-esc-working-applications/>. [Diakses 3 Juli 2020].
- [19] “Components 101 - A2212 Brushless Motor,” Components 101, 21 April 2018. [Online]. Available: <https://components101.com/motors/2212-brushless-motor>. [Diakses 3 Juli 2020].

- [20] B. Schneider, "Roger's Hobby Center - A Guide to Understanding LiPo Batteries," Roger's Hobby Cente, 2020. [Online]. Available: <https://rogershobbycenter.com/lipoguide>. [Diakses 3 Juli 2020].
- [21] K. Ogata, "Pengantar Sistem Kontrol," dalam *Teknik Kontrol Automatik*, Jakarta, Erlangga, 1995, pp. 3-4.
- [22] "Measures of Controlled System Performance," [Online]. Available: http://www.online-courses.vissim.us/Strathclyde/measures_of_controlled_system_pe.htm. [Diakses 9 Juli 2020].
- [23] O. P. Verma, S. Kumar dan G. Manik, "Analysis of Hybrid Temperature Control for Nonlinear Continuous Stirred Tank Reactor," dalam *Proceedings of Fourth International Conference on Soft Computing for Problem Solving*, India, 2015 .
- [24] "MathWorks - System Modeling and Simulation," The MathWorks Inc., 2020. [Online]. Available: <https://in.mathworks.com/solutions/system-design-simulation.html>. [Diakses 4 Juli 2020].
- [25] "MathWorks - Verification, Validation, and Test," The MathWorks Inc, 2020. [Online]. Available: <https://in.mathworks.com/solutions/verification-validation.html>. [Diakses 4 Juli 2020].
- [26] "MathWorks - Embedded Systems," The MathWorks Inc, 2020. [Online]. Available: <https://in.mathworks.com/solutions/embedded-systems.html>. [Diakses 4 Juli 2020].

LAMPIRAN

Kutipan *Listing Program* di Arduino IDE.

```
#include <Wire.h>
#include <Servo.h>

Servo left_prop;

//Initiate
int16_t Acc_rawX, Acc_rawY, Acc_rawZ, Gyr_rawX, Gyr_rawY, Gyr_rawZ;
float Acceleration_angle[2], Gyro_angle[2], Total_angle[2], angle;
float rad_to_deg = 180/3.141592654;
float elapsedTime, time, timePrev;
int i;
float value, pwmleft, error, previous_error;
double throttle=1200; //initial value of throttle to the motors
1200
float desired_angle = 0; //This is the angle in which want the
//balance to stay steady

// Create a union to easily convert float to byte
typedef union{
    float number;
    uint8_t bytes[4];
} FLOATUNION_t;

// Create the variables to receive
FLOATUNION_t myValue1;
FLOATUNION_t myValue2;

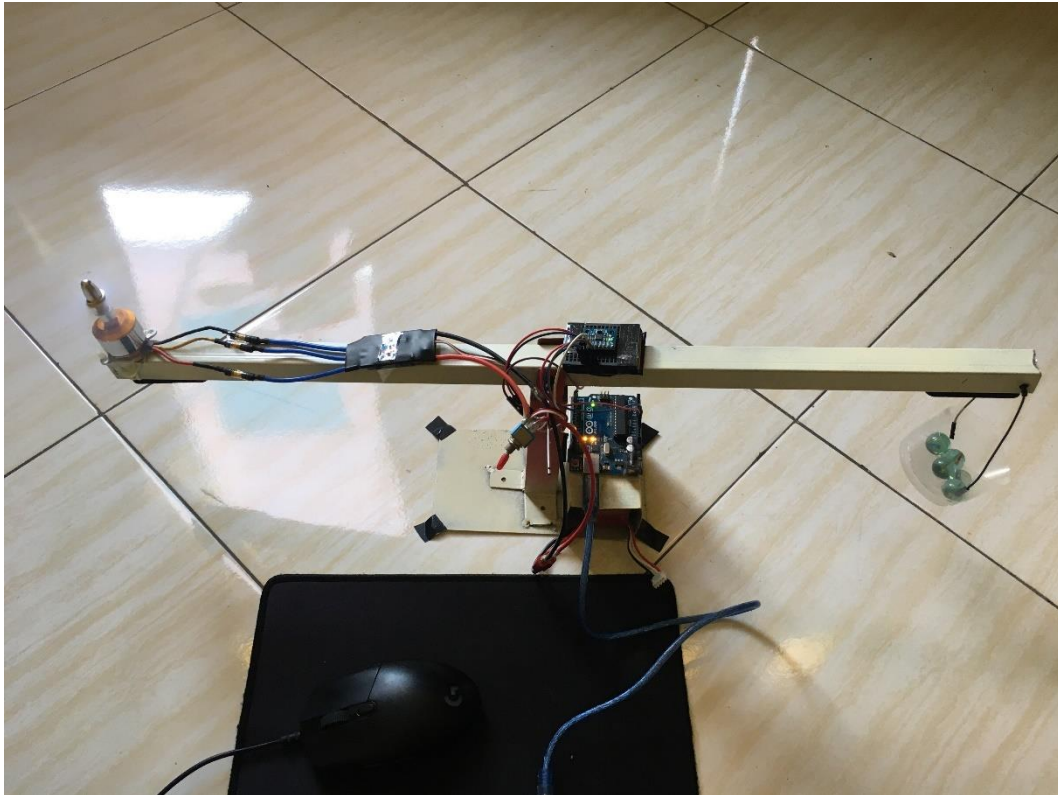
// Create the variables to send
FLOATUNION_t send1;
FLOATUNION_t send2;
```

•
•
•

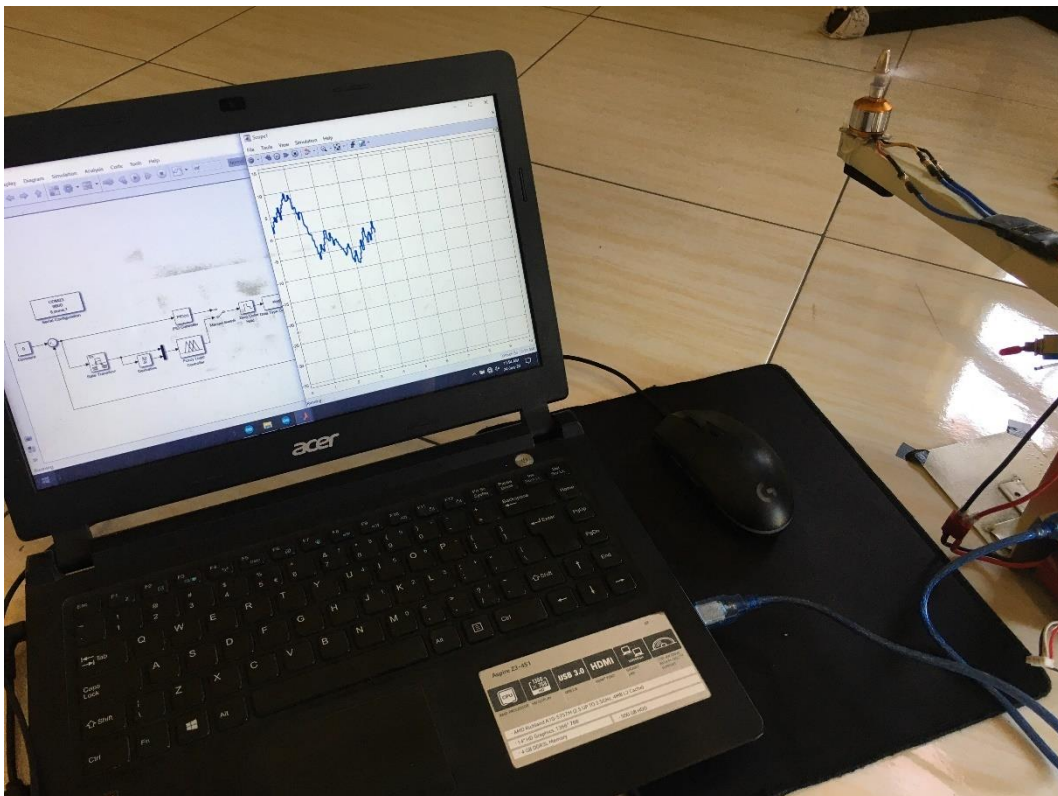
```
if(pwmleft > 1250)
{
    pwmleft=1250;
}
```

```
/*using the servo function to create the PWM pulses with the
calculated
width for each pulse*/
left_prop.writeMicroseconds(pwmleft);
previous_error = error; //Remember to store the previous error.
//delay(50);
} //end of loop void
```

```
float getFloat(){
    int cont = 0;
    FLOATUNION_t f;
    while (cont < 4 ){
        f.bytes[cont] = Serial.read() ;
        cont = cont +1;
    }
    return f.number;
}
```



Dokumentasi Pengujian Pembebanan



Dokumentasi Pembacaan Sistem *Real Time* di Simulink