ANALISIS PERFORMA KONTROL ADAPTIF DENGAN MODEL REFERENSI UNTUK KENDALI KETINGGIAN PADA *QUADCOPTER*

TESIS

PROGRAM MAGISTER TEKNIK ELEKTRO
SISTEM KONTROL DAN ELEKTRONIS

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Magister Teknik

PANJI PEKSI BRANJANGAN
NIM 146060300111021

UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
M A L A N G
2019

Karya ilmiah ini kutujukan kepada

Ayahanda dan Ibunda tercinta,

Kedua anakku Prabu dan Raka,

Istriku tersayang Roesalina

RIWAYAT HIDUP

Panji Peksi Branjangan. Lahir di Malang, 27 November 1983. Putra dari ayahanda Drs. Sunari dan Ibunda Sri Hardina. Lahir dan tinggal di Kota Malang, Lulus SMA tahun 2002, Lulus Sarjana S1 Teknik Elektro Universitas Brawijaya Malang tahun 2009, Lulus Magister Teknik Elektro Universitas Brawijaya Malang tahun 2019. Sejak 2010 bekerja sebagai tenaga pengajar di Politeknik Kota Malang, Jl Raya Tlogowari no 3 Malang, di Program Studi Mekatronika.

Memiliki keahlian di bidang elektronika dan robotika, dan sedang mengembangkan penelitian di dunia UAV (*Unmanned Aerial Vehicle*). Pernah tergabung di Tim Robot Universitas Brawijaya Malang tahun 2004 dan 2005 hingga memperoleh prestasi juara 3 Nasional KRI 2005. Beberapa kali membimbing mahasiswa dalam partisipasi berbagai lomba robot. Memiliki pengalaman menjadi Juri Lomba Robot antar SMK tahun 2018. Pernah menjadi narasumber dalam seminar *Aeromodelling* di Politeknik Kota Malang.

Malang, 26 Juli 2019

Penulis

UCAPAN TERIMAKASIH

Penulis mengucapkan terimakasih kepada Allah SWT atas rahmatNya penulis dapat menyelesaikan Tesis ini. Tak lupa Penulis juga mengucapkan terimakasih pada kedua Orang Tua yang tak pernah lelah memberikan dukungan dan semangat dalam segala hal. Penulis juga mengucapkan terimakasih pada Istri tercinta dan kedua anak tersayang yang menjadi motivasi dalam segala hal. Akhir kata ucapan terimakasih pada semua pihak yang turut membantu dalam menyelesaikan tesis ini.

Malang, 26 Juli 2019

Penulis

RINGKASAN

Panji Peksi Branjangan, Magister Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya. *Analisis Performa Kontrol Adaptif dengan Referensi Model untuk Kendali Ketinggian pada Quadcopter*. Dosen Pembimbing Dr. Ir. Bambang Siswojo, MT, Muhammad Aziz Muslim, ST., MT., Ph.D.

Untuk mendapatkan model matematis suatu sistem adalah dengan menggunakan proses identifikasi. Dalam hal ini metode yang digunakan adalah menggunakan metode RLS (Recursive Least Square). Sistem kontrol quadcopter digunakan dalam identifikasi sistem RLS tersebut. Sinyal uji untuk identifikasi sistem berasal dari PRBS (Pseudo Random Binary Sequence) yang dibangkitkan dari mikrokontroler ATMega328P. Sedangkan plant yang dikontrol adalah quadcopter yang memiliki board IMU (Inertial Measurement Unit) dengan tipe KK Board 2.1. Hasil Identifikasi selanjutnya diimplementasikan ke dalam kontrol adaptif menggunakan MRAC dengan metode MIT Rule dan Lyapunov. Kedua metode ini menghasilkan parameter kontrol yang akan dibandingkan respon berdasarkan nilai error. Semakin cepat mendekati 0 maka respon sistem semakin bagus.

Hasil identifikasi sistem ini adalah berupa fungsi alih diskrit dengan validasi berupa whiteness test dan uncorrelation test. Parameter fungsi alih diskrit yang didapatkan pada identifikasi sistem ini yaitu denominator A1 = -1,2595, A2 = 0,4422, numerator B1 = 0,0134, B2 = -0,0098, pada time sampling 0,5s. Sedangkan hasil validasi sistem menggunakan whiteness test yaitu RN(0) = 0,1338 , RN(1) = 0,1247, RN(2) = 0,0122, RN(3) = 0,0495 dan $|RN(i)| \leq 0{,}13563$ dimana batas validasi praktikalnya adalah sebesar $|RN(i)| \leq 0{,}15$.

Hasil metode MRAC yaitu untuk metode MIT Rule, parameter kontrol saat $\gamma = 1 \rightarrow \theta_1 = 4{,}33$, $\theta_2 = -0{,}024$. Untuk metode Lyapunov parameter kontrol saat $\gamma = 1 \rightarrow \theta_1 = 4{,}36$, $\theta_2 = -0{,}238$.

Kata Kunci : *Quadcopter*, Identifikasi, MRAC, *Throttle*, Adaptif,

SUMMARY

Panji Peksi Branjangan, Masters in the Department of Electrical Engineering, Faculty of Engineering, University of Brawijaya. *Adaptive Control Performance Analysis with Model Reference for Altitude Control in Quadcopter*. Supervisor Dr. Ir. Bambang Siswojo, MT, Muhammad Aziz Muslim, ST., MT., Ph.D.

To get a mathematical model of a system can be using the identification process. In this case the method used is using the RLS (Recursive Least Square) method. The quadcopter control system is used in the identification of the RLS system. The test signal for system identification comes from the PRBS (Pseudo Random Binary Sequence) which is generated from the ATMega328P microcontroller. The controlled plant is a quadcopter that has an IMU (Inertial Measurement Unit) board with KK Board 2.1 type. The identification results were then implemented into adaptive controls using MRAC with the MIT Rule and Lyapunov methods. Both of these methods produce control parameters and will be compared responses based on error values. The faster it approaches 0 is the better the system response.

The results of this system identification are in the form of discrete transfer functions with validation in the form of whiteness test and uncorrelation test. The discrete transfer function parameters obtained in this system identification are denominator A1 = -1.2595, A2 = 0.4422, numerator B1 = 0.0134, B2 = -0.0098, at time sampling 0.5s. While the results of system validation use the whiteness test, namely RN (0) = 0.1338, RN (1) = 0.1247, RN (2) = 0.0122, RN (3) = 0.0495 and $| RN (i) | \leq 0.13563$ where the practical validation limit is $| RN (i) | \leq 0.15$.

The results of the MRAC method are for the MIT Rule method, control parameters when $\gamma = 1$, $\theta_1 = 4.33$, $\theta_2 = -0.024$. For the Lyapunov method control parameters when $\gamma = 1$, $\theta_1 = 4.36$, $\theta_2 = -0.238$.

Keywords: Quadcopter, Identification, MRAC, Throttle, Adaptive

PENGANTAR

Alhamdulillah, segala puji syukur penulis panjatkan kehadirat Allah SWT, atas segala karunia dan ridho-NYA, sehingga tesis dengan judul "Analisis Performa Kontrol Adaptif dengan Model Referensi untuk Kendali Ketinggian Quadcopter" ini dapat diselesaikan.

Tesis ini disusun untuk memenuhi salah satu persyaratan memperoleh gelar Magister Teknik (M.T.) dalam bidang keahlian Sistem Kontrol dan Elektronika pada program studi Teknik Elektro Universitas Brawijaya Malang dengan sumber dana mandiri.

Oleh karena itu, pada kesempatan ini penulis menyampaikan rasa hormat dan menghaturkan terima kasih yang sebesar-besarnya, kepada:

1. Dr. Ir. Bambang Siswojo, MT. atas bimbingan, arahan dan waktu yang telah diluangkan kepada penulis untuk berdiskusi selama menjadi dosen pembimbing dan perkuliahan.
2. Muhammad Aziz Muslim, ST., MT., Ph.D yang telah memberikan bimbingan metode, masukan dan saran pada saat perkuliahan, proposal, maupun seminar hasil.
3. Seluruh Dosen program Pascasarja Teknik Teknik Elektro Universits Brawijaya yang telah memberikan arahan dan bimbingan untuk mendalami ilmu Kontrol.
4. Ayahanda Drs. Sunari, Ibunda Sri Hardina, Istri tercinta Roesalina Fenty Effendi S. Ap, dan kedua putra kami atas segala dukungan dan doanya.
5. Rekan-rekan S-2 Teknik Elektro Universitas Brawijaya yang telah membantu dalam menyelesaikan tesis ini.
6. Kepada semua pihak yang tidak dapat penulis sebutkan satu persatu.

Dengan keterbatasan pengalaman, ilmu maupun pustaka yang ditinjau, penulis menyadari bahwa tesis ini masih banyak kekurangan dan pengembangan lanjut agar benar benar bermanfaat. Penulis sangat mengharapkan kritik dan saran agar tesis ini lebih sempurna serta sebagai masukan bagi penulis untuk penelitian dan penulisan karya ilmiah di masa yang akan datang.

Akhir kata, penulis berharap tesis ini memberikan manfaat bagi kita semua terutama untuk pengembangan ilmu pengetahuan yang lebih baik.

Malang, Juli 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang Penelitian

Pada penelitian beberapa tahun terakhir, pesawat tanpa awak (UAV) menjadi tema yang menarik bagi beberapa organisasi penelitian (Gupte & Conrad, 2012). Jenis pesawat tanpa awak yang menjadi tema penelitian lebih banyak menggunakan *multicopter*. *Multicopter* dengan jenis *quadcopter* merupakan jenis pesawat tanpa awak dengan empat rotor. Berbagai parameter sangat mempengaruhi kontrol pada *quadcopter* baik itu parameter dari luar maupun internal sistem kontrol *quadcopter* sendiri. Untuk klasifikasi pesawat tanpa awak, *quadcopter* termasuk wahana yang lebih berat dari udara. Keadaan tersebut menjadikan *quadcopter* memiliki kontrol dengan parameter yang lebih kompleks dibandingkan dengan wahana yang lebih ringan dari udara misalnya balon udara.

Dari latar belakang di atas salah satu yang melatarbelakangi penelitian ini adalah jenis *multicopter* yang merupakan UAV yang lebih berat dari udara. Faktor tersebut menyebabkan kontrol ketinggian menjadi hal yang paling signifikan pada durasi terbang atau pemakaian daya baterai.

Pada penelitian (Ostrowski & Taylor, 2015) quadcopter yang dikendalikan dengan sensor kamera lebih efektif. Namun ketika kurang penerangan, sangat memperngaruhi pembacaan sensor kamera sehingga respon *plant* juga berbeda. Begitu pula yang terjadi pada penelitian (Rondon, Garcia-Carrillo, & Fantoni, 2010). Pada penelitian tersebut *platform* quadcopter membutuhkan *road model* untuk memberikan umpan balik ketinggian *quadcopter* terhadap tanah. Dengan ketinggian tertentu, maka road model sudah tidak dapat ditangkap kamera sehingga menyebabkan quadcopter kehilangan kontrol.

Kontrol ketinggian untuk *quadcopter* berbasis *non-vision* telah dibahas pada penelitian (Fatan, Sefidgari, & Barenji, 2013). Namun dalam penelitian tersebut, masih dilakukan dalam tahap simulasi. Sehingga respon kontrol dari objek yang diteliti belum merepresentasikan respon terhadap gangguan yang aktual baik gangguan internal dari *quadcopter* maupun gangguan eksternal.

Pada penelitian (Liu et al., 2014) telah dilakukan eksperimen mengenai *altitude holding quadrotor* menggunakan kalman *filter*. Demikian juga pada penelitian (Keun Uk Lee, Han

Sol Kim, Jin Bae Park, 2012) dengan eksperimen *hovering* quadcopter menggunakan sensor ultrasonik. Penelitian tersebut dilakukan pada ketinggian di bawah dua meter, yaitu menggunakan sensor sonar / ultrasonik. Metode yang digunakan dalam penelitian tersebut adalah PID dan DSC dimana dalam kontrol PID nilai parameter kontrol adalah tetap. Sehingga perubahan lingkungan dan parameter lain yang mempengaruhi dinamika kontrol *quadcopter* tidak dapat diadaptasi oleh *quadcopter* tersebut. Dalam bagian akhir jurnal tersebut ditulis bahwa penelitian lanjutan berikutnya yaitu tentang barometer sebagai sensor umpan balik ketinggian *platform quadrotor* tersebut. Dari penelitian ini maka akan digunakan sensor barometer atau tekanan udara sebagai alternatif sensor ultrasonik untuk mengukur ketinggian *quadcopter* terhadap tanah terutama sebagai pemodelan sistem.

Dari masalah tersebut maka penulis mengusulkan suatu solusi kontrol ketinggian *quadcopter* yang dapat menyesuaikan kondisi perubahan lingkungan sekitar dengan meng-*update* nilai parameter kontrol sesuai dengan model yang diinginkan. Tujuannya yaitu dengan perubahan lingkungan seperti apapun, maka *quadcopter* dapat beradaptasi. Dengan tujuan tersebut maka salah satu solusi dari kontrol tersebut adalah menggunakan metode kontrol adaptif. Diharapkan dengan solusi tersebut maka kontrol ketinggian dari *quadcopter* dapat dilakukan dengan lebih baik dari metode sebelumnya.

1.2 Identifikasi dan Perumusan Masalah

Kecenderungan perubahan kontrol ketinggian *quadcopter* pada ketinggian yang berbeda memunculkan beberapa permasalahan yang harus diselesaikan. Dari berbagai latar belakang tersebut, maka penulis mengambil beberapa perumusan masalah yaitu sebagai berikut:

1. Bagaimana memperoleh data kecepatan vertikal beserta parameter kontrol *quadcopter* dalam domain waktu
2. Bagaimana cara identifikasi plant *quadcopter* dengan input sinyal *throttle* dan output kecepatan vertikal
3. Bagaimana cara validasi sistem plant kontrol ketinggian *quadcopter* yang telah diidentifikasi
4. Bagaimana implementasi adaptif kontrol menggunakan MRAC MIT Rule dan Lyapunov pada sistem plant yang telah diidentifikasi

1.3 Tujuan Penelitian

Tujuan tesis ini adalah:

1. Untuk memilih sensor yang tepat pada pemodelan plant kontrol ketinggian *quadcopter*
2. Untuk menemukan fungsi transfer dari plant kontrol ketinggian *quadcopter*

3. Untuk mengetahui nilai parameter kontrol adaptif menggunakan MRAC MIT Rule dan Lyapunov, serta menentukan perfoma metode yang lebih baik untuk kontrol ketinggian *quadcopter*.

1.4 Kontribusi Penelitian

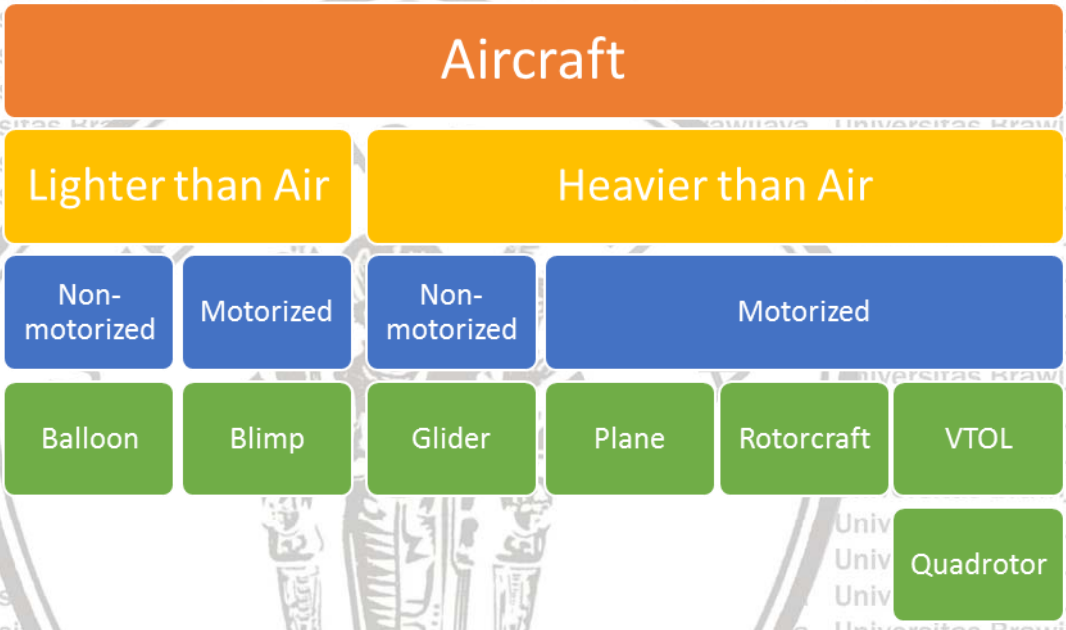
Merujuk pada tujuan penelitian, diharapkan tesis ini memiliki:

1. Manfaat teori, tesis ini bisa menjadi sarana pembelajaran yang lebih praktis pada materi kontrol adaptif
2. Manfaat praktis, dari hasil eksperimen dapat membantu perancang maupun pengendali *quadcopter* dalam kontrol ketinggian pada VTOL sehingga kontrol ketinggian *quadcopter* dapat di-otomatisasi meskipun terdapat gangguan signifikan, serta dapat digunakan untuk implementasi "*vertical crash avoidance*".

BAB II
TINJAUAN PUSTAKA

2.1 Klasifikasi *Aircraft*

Aircraft atau kendaraan terbang memiliki beberapa klasifikasi baik secara struktur maupun gejala fisis yang bekerja di dalamnya. Klasifikasi *aircraft* ditampilkan dalam Gambar 2.1.



Gambar 2.1 Klasifikasi *aircraft*
Sumber : (Gupte & Conrad, 2012)

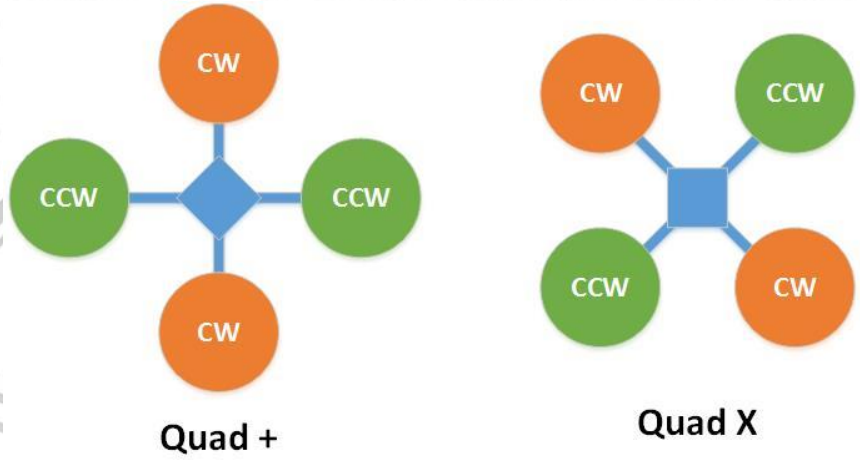
Dalam Gambar 2.1 ditunjukkan bahwa *quadrotor*/ *quadcopter* adalah salah satu jenis *aircraft* yang memiliki kemampuan VTOL yaitu *take-off* dan *landing* secara vertikal. Daya angkat dari *quadcopter* diperoleh dari motor yang digerakkan secara bebas di tiap titik aktuatornya. Struktur fisik *quadcopter* telah jelas terlihat bahwa *quadcopter* termasuk jenis *aircraft* yang lebih berat dari udara dibandingkan dengan balon terbang maupun *blimp*/ *zeppelin*.

2.2 Prinsip Kerja Quadcopter

Quadrotor atau *quadcopter* adalah salah satu jenis unik dari UAV yang memiliki kemampuan VTOL (*Vertical Take-Off and Landing*). Dari sifat dinamika yang dimiliki, *quadcopter* ini memiliki keuntungan bermanuver. Input *quadcopter* ini terdiri dari *elevator*,

aileron, *rudder* dan *throttle*. Sedangkan keluarannya yaitu berupa posisi x, y, z dan orientasi θ, ψ, ω (Gupte & Conrad, 2012).

Desain quadrotor merupakan desain UAV yang memiliki empat rotor dengan posisi rotor memiliki jarak yang sama terhadap titik pusatnya. Konfigurasi tersebut menghasilkan daya dorong vertikal/ tegak lurus. Dengan spesifikasi yang sama antar rotor, maka diharapkan *quadcopter* ini memiliki daya dorong yang sama sehingga dapat bergerak vertikal dengan diberi satu input kontrol saja. Keuntungan dari *quadcopter* adalah tidak perlu menambahkan perangkat tambahan untuk menyeimbangkan torsi inersia yang dihasilkan dari rotornya (M Muhammad, D Swarnaker 2014).

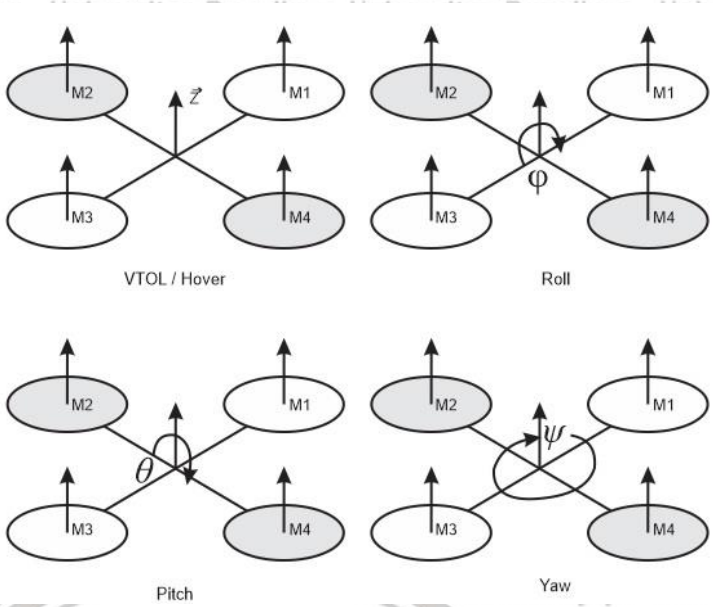


Gambar 2.2 Konfigurasi + dan X pada *Quadcopter*
Sumber: (Gupte & Conrad, 2012)

Dalam Gambar 2.2 tersebut terdapat empat motor yang terbagi menjad dua macam putaran yaitu CW dan CCW. Untuk peletakan motor harus berlawanan arah untuk memberikan resultan gaya yang seimbang antara momen CW dan CCW dari masing-masing motor.

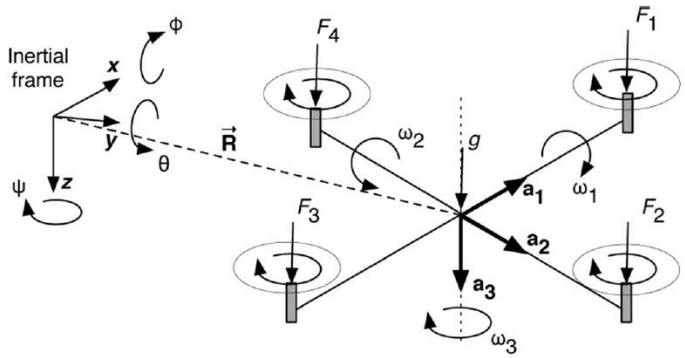
2.3 Gerakan *Quadcopter*

Konfigurasi rotor yang dimiliki *quadcopter*, menjadikan *quadcopter* memiliki respon gerakan yang berbeda dari helikopter konvensional. Hal ini bisa divisualisasikan pada Gambar 2.3.



Gambar 2.3 Gerakan *Quadcopter*
Sumber: (Joyo et al., 2013)

Dalam Gambar tersebut terdapat beberapa istilah yaitu VTOL/ *hover, roll, pitch, yaw*. VTOL adalah suatu cara UAV dalam *take-off* dan *landing* dengan arah vertikal terhadap permukaan bumi. *Roll* adalah gerakan rotasi ke arah samping kanan-kiri terhadap pusat UAV. *Pitch* adalah gerakan rotasi depan-belakang terhadap pusat UAV. Sedangkan *yaw* adalah gerakan orientasi UAV pada sumbu *horizontal*. Dalam gamabr 2.3 pula dijelaskan gaya yang bekerja pada tiap-tiap motor M1 dan M3 yang bergerak berlawanan arah dengan M2 dan M4.



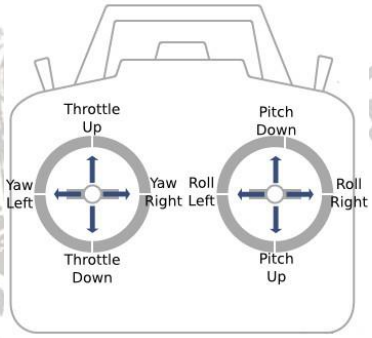
Gambar 2.4 Diagram gerakan *quadcopter*
Sumber: (Sebesta & Boizot, 2014)

Dalam Gambar 2.4 ditunjukkan perpindahan posisi dan orientasi *quadcopter* dari ketiga sumbu X, Y dan Z. untuk a_1 a_2 dan a_3 adalah perpindahan translasional dari *quadcopter* dari ketiga sumbu. Dan untuk ω_1 ω_2 dan ω_3 adalah perpindahan rotasional dari masing-masing sumbu.

Terdapat empat macam input agar *quadcopter* bergerak secara translasional dan rotasional yaitu *Aileron, Elevator, Rudder, Throttle.* Perubahan input *aileron* menyebabkan perubahan respon pada *roll*. Jika *roll* berubah maka *quadcopter* akan berubah posisi pada sumbu x. Begitu pula yang terjadi dengan *elevator* terhadap *pitch* serta *rudder* terhadap *yaw*. Input yang berpengaruh terhadap perubahan gerakan pada sumbu z adalah *throttle*. Input inilah yang selanjutnya akan dikendalikan dengan kontrol adaptif agar dapat menyesuaikan parameter kontrol dengan keadaan lingkungan kerjanya.

2.4 Kontrol Manual *Quadcopter*

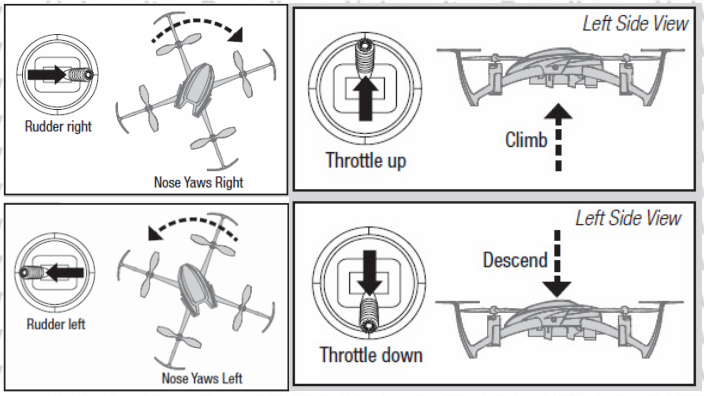
Untuk mengontrol *quadcopter* secra manual atau menggunakan kontrol penuh oleh manusia, maka alat yang biasa digunakan adalah menggunakan modul *radio transmitter*. Minimal jumlah kanal yang digunakan adalah sejumlah empat kanal. Keempat kanal ini terdiri dari kanal *aileron,* kanal *elevator,* kanal *throttle,* dan kanal *rudder*.



Gambar 2.5 Posisi kanal kontrol manual *radio transmitter quadcopter*

Dalam Gambar 2.5 terdapat dua kanal kontrol pada sisi kanan dan dua kanal kontrol pada sisi kiri. Jenis model yang digunakan dalam Gambar 2.5 adalah model *radio transmitter* V2. maksud dari V2 adalah letak kanal *throttle* dan *rudder/yaw control* pada sebelah kiri sedangkan letak *aileron/roll* dan *elevator/pitch* pada sisi kanan. Setiap tuas pengatur kanal terdapat ciri khas yaitu untuk pegas pada tuas kontrol *roll - pitch - yaw* yang dapat mengembalikan posisi tuas ke tengah ketika dilepas. Sedangkan untuk kontrol *throttle* tidak terdapat pegas untuk menahan sinyal *throttle* pada posisi tertentu ketika dilepas.

Untuk mengilustrasikan respon *quadcopter* terhadap gerakan tuas kontrol *radio transmitter* bisa divisualisasikan dalam Gambar 2.6.



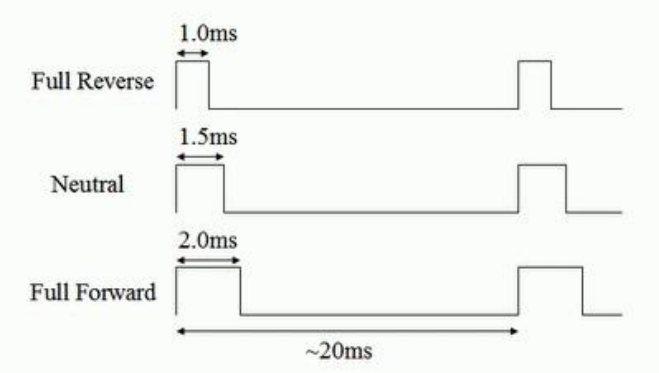
Gambar 2.6 Ilustrasi respon *quadcopter* terhadap kontrol manual *joystick*

Untuk menerima sinyal dari modul *radio transmitter quadcopter*, digunakan *receiver* atau unit penerima sinyal. *Receiver* ini memiliki jumlah kanal output yang dihubungkan ke kontroler utama *quadcopter* sehingga dapat merespon dengan gerakan yang diinginkan.

Gambar 2.7 Contoh receiver Fly Sky 6 kanal output

Dalam Gambar 2.7 jumlah kanal yang dapat digunakan adalah 6 (enam) kanal. Namun untuk mengontrol pergerakan *quadcopter*, terkadang tidak semua kanal dipakai. Sebagai contoh untuk dasar gerakan *quadcopter* terdapat empat kanal utama. Sedangkan sisanya dapat digunakan untuk mengatur mode terbang, *shutter* kamera, *launcher* parasut, dll.

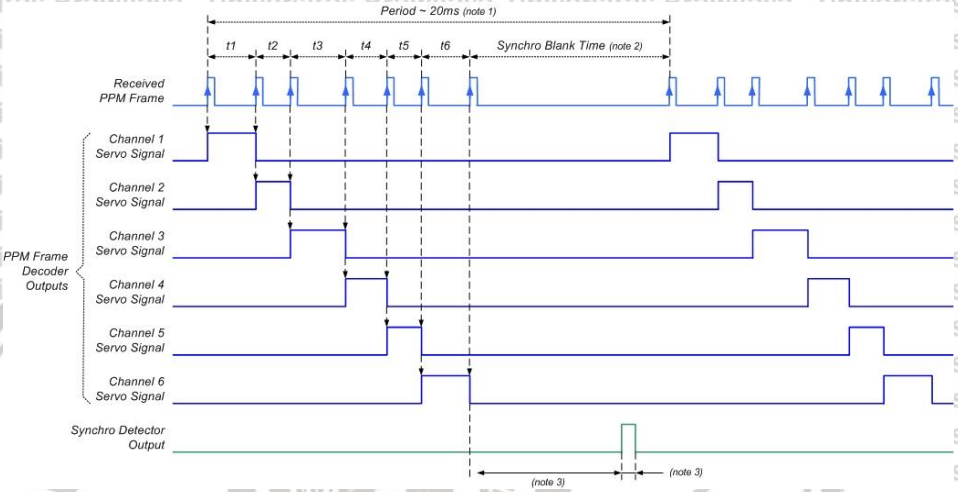
Sinyal output yang dihasilkan oleh unit penerima / *receiver* tersebut berupa sinyal servo. Sinyal servo adalah suatu sinyal PWM dengan parameter; tegangan TTL, periode 20ms, dan % duty 5-10%. Sinyal inilah yang mengatur besarnya *throttle*, seberapa cepat respon *roll dan pitch*, serta seberapa cepat putaran *yaw* dilakukan.



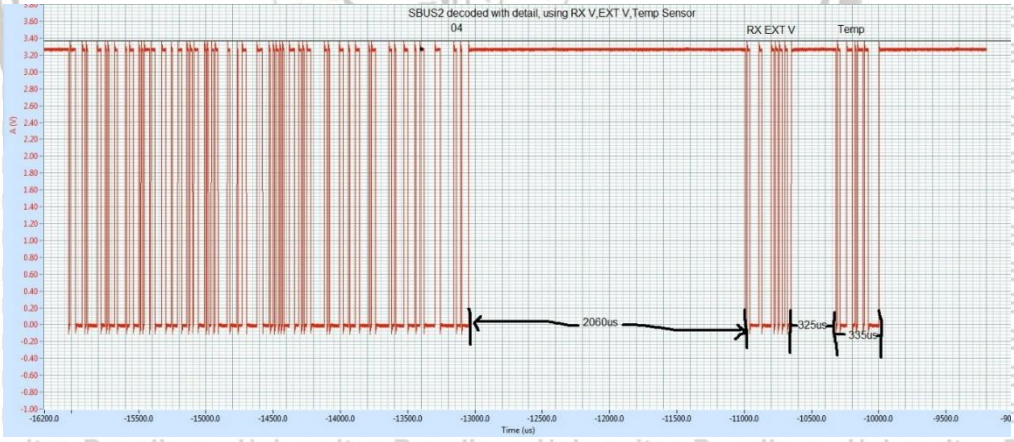
Gambar 2.8 Sinyal PWM *radio receiver*

Dari Gambar 2.8 dapat dijelaskan bahwa sinyal *full reverse* dan *full forward* adalah representasi dari sinyal minimum dan maksimum. *Range*/jangkauan *duty cycle* yang digunakan sinyal tersebut adalah mulai dari 1ms atau 1000µs sampai 2ms atau 2000µs.

Selain sinyal servo terdapat bentuk sinyal lain yang digunakan untuk komunikasi antara *radio receiver* dengan papan kontroler utama *quadcopter*. Yaitu menggunakan protokol PPM dan SBUS. PPM adalah pulse position modulation. Sedangan SBUS adalah pengiriman dengan data digital serial, bukan lebar pulsa. Contoh dari SBUS dapat diamati dalam Gambar 2.9 dan 2.10



Gambar 2.9 Protokol komunikasi PPM unit *receiver* dengan kontroler *quadcopter*



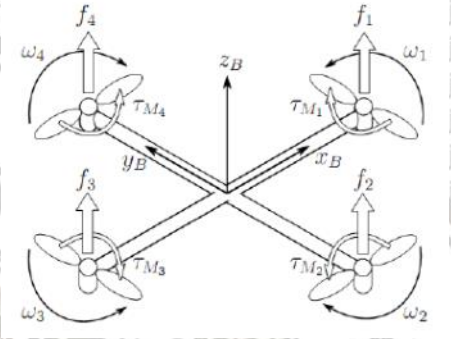
Gambar 2.10 Protokol komunikasi SBUS unit *receiver* dengan kontroler *quadcopter*

Komunikasi PPM adalah pengembangan dari PWM namun dalam satu periode terdapat delapan duty cycle yang dapat digunakan. Sehingga sangat menghemat proses *wiring* / pengkabelan. Hanya dengan satu kabel saja delapan kanal kontrol sudah dapat dikirim. Namun kelemahan dari PPM adalah maksimal mampu mengirim delapan kanal saja karena sinyal kontrol yang maksimal 2ms dikirim dalam periode 20ms dengan sisa 2x sinyal maksimum sebagai notifikasi pergantian periode. Sehingga hanya mampu menggunakan

16ms sinyal komunikasi atau 16/2 kanal. Yaitu 8 kanal. Untuk protokol SBUS, data yang dikirim adalah data digital serial. Sehingga kanal yang dapat dikontrol lebih banyak dari PPM hanya dengan satu kabel saja. Data yang dikirim pun tidak hanya sinyal kontrol tetapi juga dapat mengirim *telemetry* sensor dan lain sebagainya dalam sekali periode waktu pengiriman.

2.5 *Quadcopter* Model

Pemodelan *quadcopter* merupakan salah satu langkah untuk visualisasi gejala fisik yang terjadi dalam *quadcopter* agar dapat dianalisis dengan metode-metode yang ada. Salah satu model *quadcopter* dapat diamati dalam Gambar 2.11.



Gambar 2.11 Struktur *quadcopter* dan gaya yang bekerja di dalamnya
Sumber : (Fatan et al., 2013)

Model *quadrotor* menggunakan persamaan dinamik yang berbasis *Newton-Euler*. Persamaan 2-1 menjelaskan persamaan total yang bekerja dalam *quadcopter*. (Fatan et al., 2013)

$$mV_B+v\times(mV_B)=R^TG+T_B \qquad (2\text{-}1)$$

m = massa quadcopter (g)

V_B = kecepatan liniear (m/s)

v = kecepatan sudut (rad/s)

R = matrix rotasi

T = transpose matriks

G = percepatan gravitasi (m/s^2)

T_B = total gaya (N)

Sistem lepas landas otomatis dibutuhkan untuk UAV lepas landas dari tanah. Seharusnya sistem memastikan ketinggian dari *quadcopter* dalam kondisi yang konsisten. Dalam kasus ini, tiap-tiap motor dan baling-balingnya harus menghasilkan gaya yang melawan gravitasi dan dapat mengangkat *quadcopter*. Performansi yang lebih baik dari lepas landas otomatis menandakan seberapa cepat *quadcopter* menuju kondisi *steady*.

Performansi dari sistem juga tergantung dari tingkat kehalusan sistem dalam mengatur ketinggian *quadcopter* tersebut.

Dalam penelitian (Mustapa, Saat, Husin, & Abas, 2014) dinyatakan bahwa model matematik disajikan dan diuji untuk simulasi perilaku dari *quadcopter*. Hasil dari penelitian tersebut yang digunakan untuk kontrol ketinggian adalah sebagai berikut:

$$\ddot{X}=(\sin\psi\sin\phi+\cos\psi\sin\theta\cos\phi)\frac{U_1}{m} \quad (2\text{-}2\text{-}1)$$

$$\ddot{Y}=(-\cos\psi\sin\phi+\sin\psi\sin\theta\sin\phi)\frac{U_1}{m} \quad (2\text{-}2\text{-}2)$$

$$\ddot{Z}=-g+(\cos\phi+\cos\theta)\frac{U_1}{m} \quad (2\text{-}2\text{-}3)$$

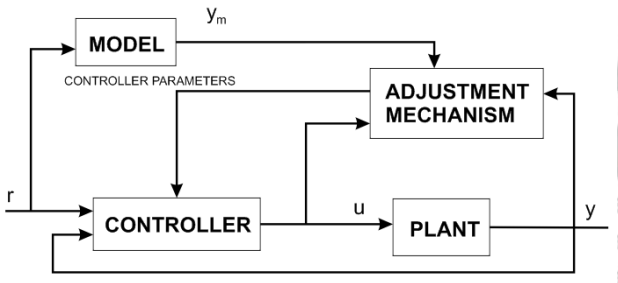
$$\dot{p}=\frac{U_2}{I_x} \quad (2\text{-}2\text{-}4)$$

$$\dot{q}=\frac{U_3}{I_y} \quad (2\text{-}2\text{-}5)$$

$$\dot{r}=\frac{U_3}{I_z} \quad (2\text{-}2\text{-}6)$$

Persamaan 2-2 adalah model matematis dari *quadcopter* secara keseluruhan. 2-2-1 hingga 2-2-3 adalah persamaan vektor linier sedangkan tiga persamaan berikutnya adalah vektor kecepatan sudut dari *quadcopter* itu sendiri.

2.6 MRAC (*Model Reference Adaptive Control*)



Gambar 2.12 Model Reference Adaptive Control
Sumber: (Pawar, Parvat, & D, 2015)

Model reference adaptive kontrol adalah suatu sistem kontrol dimana spesifikasi yang diinginkan berdasarkan model yang diberikan. Pada dasarnya MRAC terdiri dari dua loop, pertama adalah loop umpan balik normal, loop kedua adalah loop untuk penyesuaian parameter. Model referensi menunjukkan bagaimana proses keluaran seharusnya memberikan respon pada sinyal perintah. Referensi model keluaran dan plant dibandingkan, galat di antara keduanya diumpankan pada umpan balik loop pengatur parameter. Parameter dari kontroler di-update untuk meminimalisasi galat hingga bernilai nol. Terdapat dua metode pendekatan untuk implementasi MRAC yaitu MIT rule dan teori *Lyapunov*. Kedua

metode tersebut dapat pula digunakan sebagai perbandingan dalam melakukan perurunan formulasi sebelum digunakan dalam kontrol eksperimental.

2.6.1 Metode MIT Rule

MIT Rule adalah salah satu pendekatan dalam *Model Reference Adaptive Control (*MRAC*).* Nama MIT Rule muncul saaat pertama kali dikembangkan di Laoratorium Instrumentasi MIT.

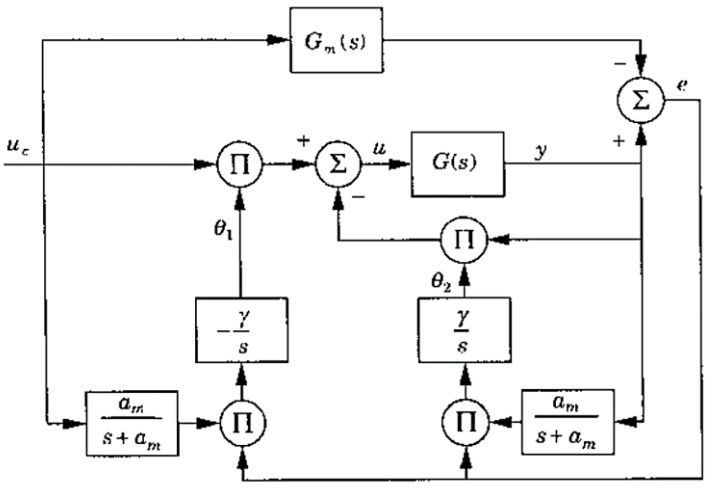
Untuk menjelaskan MIT Rule, hal yang harus diperhatikan adalah pada sistem loop tertutup terdapat satu *adjustable* parameter kontrol yaitu θ. Respon loop tertutup yang diinginkan ditentukan oleh model dengan output y_m. Biarkan e sebagai error antara output y dan output y_m dari model. Satu kemungkinan untuk menyesuaikan parameter sesuai dengan

$$J(\theta)=\frac{1}{2}e^2 \quad (2\text{-}3)$$

diminimalisasi, fungsi tersebut dapat diminimalisasi jika mengubah parameter dengan gradien negatif dari J, dikenal dengan pendekatan *gradient descent*, yang diikuti dengan pola,

$$\frac{d\theta}{dt}=-\gamma\frac{\partial J}{\partial \theta}=-\gamma e\frac{\partial e}{\partial \theta} \quad (2\text{-}4)$$

Persamaan 2-4 dikenal dengan MIT rule.



Gambar 2. 13 Blok Diagram MIT Rule

Dari Gambar 2.13 terlihat bahwa terdapat terdapat parameter theta (θ) 1 dan 2 yang akan terus di*update* hingga memperoleh nilai error (e) yang mendekati 0. Besarnya error diperoleh dari selisih antara y dan y_m.

2.6.2 Metode Lyapunov

Kontribusi dasar untuk teori stabilitas dari sistem tak linier dibuat oleh ahli matematika Rusia yaitu *Lyapunov* pada akhir abad 19. *Lyapunov* menyatakan bahwa persamaan diferensial tak linier,

$\frac{dx}{dt}$=f(x) dimana f(0)=0 (2-5)

f(0)=0 adalah persamaan yang memiliki solusi $x(t) = 0$. Untuk menjamin solusi tersebut, penting dalam membuat asumsi bahwa *f(x)* adalah *Lipschitz* lokal, yang mana

$\|f(x) - f(y)\| \leq L \|x\| - y$ dimana L>0 (2-6)

adalah nilai yang dekat dengan pusat. Jika terdapat fungsi $V : R^n \rightarrow R$ adalah positif dengan nilai terhingga,yang mana menghasilkan fungsi turunan,

$\frac{dV}{dt}=\frac{\partial V^T}{\partial x}\frac{dx}{dt}=\frac{\partial V^T}{\partial x}$f(x)=-W(x) (2-7)

adalah bernilai negatif *semidefinite*, maka solusi x(t)=0 untuk persamaan 2-5 memiliki sifat stabil. Jika $\frac{dV}{dt}$ bernilai negatif terhingga, maka memiliki solusi stabil secara asimtotik. Fungsi V disebut sebagai fungsi Lyapunov untuk sistem persamaan 2-5.Lebih daripada itu jika $\frac{dV}{dt} < 0$ dan $V(x) \rightarrow \infty$ ketika $\|x\| \rightarrow \infty$ maka solusinya bersifat global stabil secara asimtotik.

Asumsikan bahwa sistem linier

$\frac{dx}{dt}$=Ax (2-8)

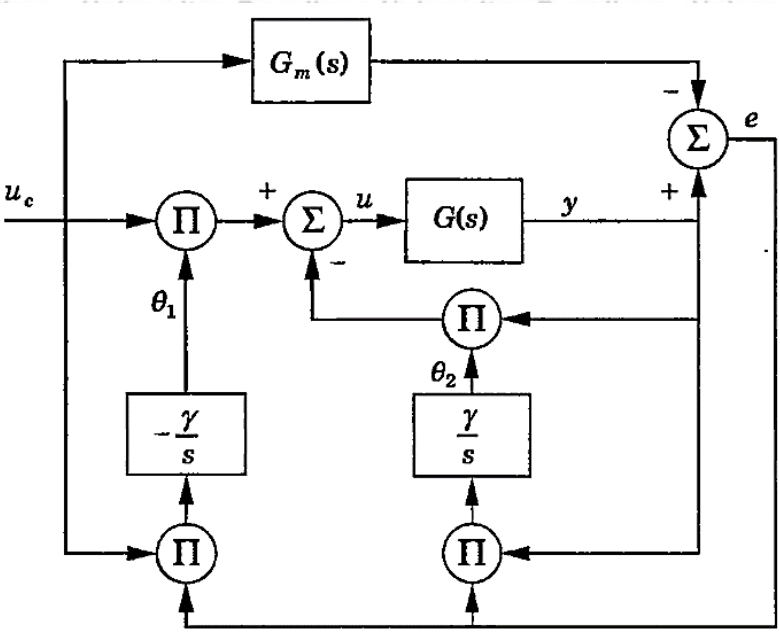
Adalah stabil secara asimtotik. Maka untuk tiap matriks Q yang terhingga positif simetris, terdapat matriks P yang unik positif simetris seperti

$A^TP+PA=-Q$ (2-9)

Sehingga fungsi *Lyanpunov* untuk fungsi 2-8 adalah

$V(x)= x^TPx$ (2-10)

Berikut adalah blok diagram kontrol menggunakan metode MRAC Lyapunov :



Gambar 2. 14 Blok Diagram Kontrol Lyapunov

Dari Gambar 2.14 terlihat bahwa terdapat terdapat parameter theta (θ) 1 dan 2 yang akan terus di*update* hingga memperoleh nilai error (e) yang mendekati 0. Besarnya error diperoleh dari selisih antara y dan y_m.

$$e = y - y_m \qquad (2\text{-}11)$$

Ketika error semakin kecil, maka secara alamiah terdapat persamaan diferensial untuk error yaitu :

$$\frac{de}{dt} = -a_m - (b\theta_2 + a - a_m)y + (b\theta_1 - b_m)u_c \qquad (2\text{-}12)$$

Untuk membentuk parameter adaptif θ_1 dan θ_2, maka dignkan fungsi kuadratik sebagai berikut:

$$V(e, \theta_1, \theta_2) = \frac{1}{2}\left(e^2 + \frac{1}{b\gamma}(b\theta_2 + a - a_m)^2 + \frac{1}{b\gamma}(b\theta_1 - b_m)^2\right) \qquad (2\text{-}13)$$

Fungsi tersebut akan bernilai 0 jika error bernilai 0 dan parameter kontrol bernilai benar. Untuk fungsi Lyapunof yang bernilai negative dV/dt, maka berlaku:

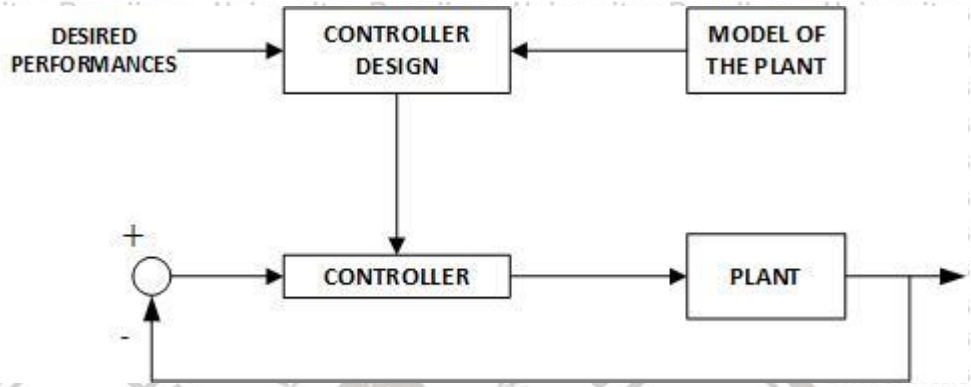
$$\begin{aligned}\frac{dV}{dt} &= e\frac{de}{dt} + \frac{1}{\gamma}(b\theta_2 + a - a_m)\frac{d\theta_2}{dt} + \frac{1}{\gamma}(b\theta_1 - b_m)\frac{d\theta_1}{dt} \\ &= -a_m e^2 + \frac{1}{\gamma}(b\theta_2 + a - a_m)\left(\frac{d\theta_2}{dt} - \gamma y e\right) \\ &\quad + \frac{1}{\gamma}(b\theta_1 - b_m)\left(\frac{d\theta_1}{dt} + \gamma u_c e\right)\end{aligned} \qquad (2\text{-}14)$$

Dan update parameter untuk theta 1 dan 2 adalah sebagai berikut:

$$\frac{d\theta_1}{dt} = -\gamma u_c e$$
$$\frac{d\theta_2}{dt} = \gamma y e \quad (2\text{-}15)$$

2.7 Identifikasi Sistem

Identifikasi dalam sistem kontrol artinya proses menentukan model dari sistem dinamik dari sinyal input/output yang terukur. Pengetahuan tentang model sangat diperlukan untuk desain dan implementasi dari kontrol untuk performa yang tinggi.



Gambar 2.15 Prinsip desain kontroler

Dalam Gambar 2.13 menunjukkan diagram secara umum prinsip dalam desan kontroler. Sesuai dengan diagram tersebut, untuk merancang desain dan *tunning* kontroler dibutuhkan langkah-langkah:

1. Menentukan performa dan keandalan kontrol loop yang diinginkan
2. Mengetahui model dinamik dari plant yang dikontrol dengan mendeskripsikan varian kontrol dan varian output
3. Mendapatkan metode desain kontrol yang cocok yang dapat memenuhi performa dan keandalan kontrol dari plant terkait.

Gagasan tentang model matematis suatu sistem atau fenomena merupakan suatu konsep yang mendasar. Secara umum, banyak tipe model ada, masing-masing digunakan untuk aplikasi tertentu. Misalnya, model tipe ilmu pengetahuan (berdasarkan hukum fisika, kimia, dll) memungkinkan deskripsi sistem yang cukup lengkap dan digunakan untuk simulasi dan desain *plant*. Model ini pada umumnya sangat kompleks dan jarang sekali langsung digunakan untuk perancangan sistem kontrol. Kontrol dinamis model yang memberikan hubungan antara variasi input dan output suatu sistem, sesuai untuk disain dan *tuning* sistem kontrol tersebut.

Meski struktur model kontrol ini bisa diperoleh dari struktur model tipe pengetahuan, pada umumnya sangat sulit untuk menentukan nilai parameter yang signifikan dari model

tersebut. Hal ini merupakan alasan dalam sebagian besar situasi praktis, perlu untuk menerapkan metodologi untuk identifikasi langsung model dinamis (kontrol) ini dari data eksperimental.

Perlu diperhatikan bahwa ada dua tipe model dinamis yaitu:

1. Model non-parametrik (contoh: respons frekuensi, respons langkah)
2. Model parametrik (contoh: fungsi transfer, diferensial atau perbedaan persamaan)

Oleh karena itu, kita harus memperhatikan identifikasi sampel diskrit model dinamis parametrik fungsi waktu, yang paling sesuai untuk disain dan *tuning* penyetelan sistem kontrol digital.

Identifikasi sistem merupakan pendekatan eksperimental untuk menentukan dinamika model sebuah sistem. Identifikasi tersebut mencakup empat langkah:

1. Akuisisi data input dan output di bawah protokol eksperimen
2. Seleksi atau estimasi struktur "model" (kompleksitas)
3. Estimasi parameter model
4. Validasi model yang diidentifikasi (struktur dan nilai parameter)

Operasi identifikasi lengkap harus terdiri dari empat tahap. Metode spesifik yang digunakan pada setiap tahap tergantung pada jenis model yang diinginkan (parametrik atau non parametrik, waktu kontinyu atau diskrit-waktu) dan pada kondisi percobaan (misalnya: hipotesis pada *noise*, loop terbuka atau identifikasi loop tertutup). Validasi adalah langkah wajib untuk memutuskan apakah model yang diidentifikasi bisa diterima atau tidak.

Karena tidak ada algoritma estimasi parameter khusus dan protokol eksperimental khusus yang selalu mengarah pada model yang teridentifikasi, model yang didapat mungkin tidak selalu lulus uji validasi. Dalam hal ini, perlu mempertimbangkan kembali algoritma estimasi, kompleksitas model atau kondisi eksperimen. Identifikasi sistem harus dianggap sebagai prosedur iteratif.

Berikut adalah ringkasan singkat elemen penting yang menjadi ciri khas langkah-langkah identifikasi sistem.

Akuisisi Data Input / Output dengan Protokol Eksperimental

Pada dasarnya harus dipilih sinyal eksitasi dengan spektrum kaya frekuensi agar mencakup bandwidth plant yang akan diidentifikasi, namun dengan besaran kecil (karena dalam prakteknya, variasi besarnya sinyal input yang diterima adalah sangat dibatasi).

Seleksi atau Estimasi Model Kompleksitas

Masalah khas yang dihadapi adalah menentukan perintah polynomial (pembilang, penyebut) fungsi transfer pulsa yang mewakili model plant. Seringkali digunakan prosedur

trial and error, namun teknik yang lain yaitu estimasi kompleksitas model telah dikembangkan.

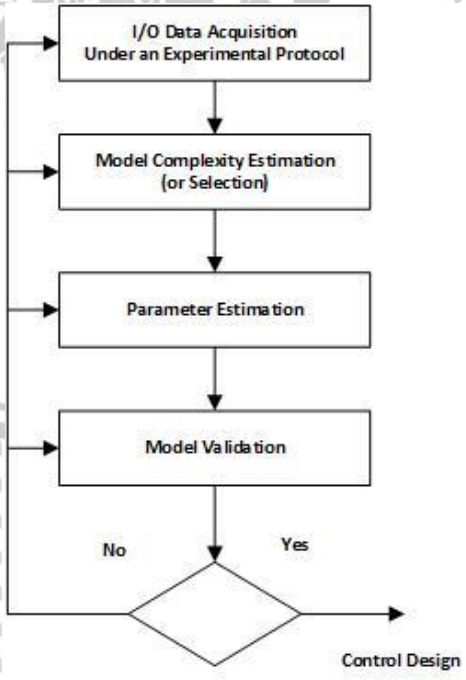
2.7.1 Estimasi Parameter Model

Metodologi identifikasi "klasik" digunakan untuk mendapatkan model parametrik berbasis pada model non-parametrik dari tipe "respons langkah" awalnya digunakan untuk mendapatkan model parametrik waktu kontinyu telah diperluas untuk identifikasi model waktu diskrit.

Dari bentuk respon step plant, untuk pemilihan tipe model dan parameter model ini ditentukan secara grafis. Jika frekuensi sampling diketahui, maka dapat diperoleh model diskrit waktu yang sesuai dari konversi tabel.

Metodologi ini memiliki beberapa kelemahan:

- Uji sinyal dengan besaran besar (jarang diterima di sistem industri)
- Mengurangi akurasi
- Buruknya pengaruh gangguan
- Model untuk gangguan tidak tersedia
- Prosedur yang panjang
- Tidak adanya validasi model



Gambar 2. 16 Metodologi Identifikasi Sistem

Kemampuan komputer digital dalam implementasi suatu algoritma dapat melakukan estimasi otomatis untuk model waktu diskrit. Identifikasi dari model parametrik - diskrit dapat dilakukan (dengan simulasi) model non-parametrik of step respon atau jenis respon

frekuensi, dengan derajat akurasi yang jauh lebih tinggi dalam pendekatan secara langsung, dan hanya menggunakan sinyal eksitasi yang sangat lemah. Identifikasi dari model data sampel parametrik membawa model menjadi dapat digunakan untuk keperluan lebih luas dan menawarkan banyak keuntungan pada pendekatan lainnya.

Saat ini telah dikembangkan algoritma dengan performa tinggi, yang memiliki formula perulangan / *recursive* lalu digabungkan pada identifikasi pada masalah *real-time* dan diimplementasikan pada mikro komputer. Faktanya identifikasi dengan metode tersebut yang yang hanya dioperasikan dengan sinyal eksitasi yang sangat lemah sangat baik pada situasi praktis atau pada saat eksperimen.

Prinsip estimasi parameter untuk model waktu diskrit dilustrasikan dalam gambar prinsip model estimasi parameter diskrit. Sampel input pada urutan *u(t)* (dimana t adalah waktu diskrit) diaplikasikan pada kondisi fisik (pada tahapan aktuator-*plant*-transduser) yang artinya *digital-to-analog converter* (DAC) dihubungkan ke *zero order hold block* (ZOH). *Output sampel plant* yang diukur y(t) dihasilkan oleh nilai rata-rata dari *analog-to-digital converter* (ADC).

Model waktu diskrit dengan parameter yang dapat diatur diimplementasikan pada komputer. *Error* dari output sistem y(t) pada t instan, dan *output* yang diprediksi oleh model (dikenal dengan *prediction error*) digunakan oleh algoritma adaptasi parameter yang akan mengubah model parameter dengan tujuan untuk meminimalisasi *error* pada *criterion* yang dipilih.

2.7.2 Estimasi Parameter Model Plant

Sesuai dengan model waktu diskrit dari *plant* yang dideskripsikan sebagai:

$$y(t+1) = -a_1 y(t) + b_1(t)u(t) = \theta^T \phi(t) \qquad (2\text{-}16)$$

dimana:

y = output sistem

u = input sistem

a_1 dan b_1 adalah parameter yang tidak diketahui.

Model *output* apabila ditulis dalam bentuk perkalian *scalar* antar *vector* parameter menghasilkan persamaan:

$$\theta^T = [a_1, b_1] \qquad (2\text{-}17)$$

Dan vector yang menunjukkan *regressor* model plant dapat dituliskan dengan persamaan:

$$\phi(t)^T = [-y(t), u(t)] \qquad (2\text{-}18)$$

Metode *Least Squared Error* digunakan dalam penelitian ini. Persamaan yang menyatakan metode tersebut adalah:

$V_t(\theta, Z^t) = \sum_{i=t}^{t} \varepsilon(i)^2$ (2-19)

dimana:

$\varepsilon(t) = y(t) - \hat{y}(t)$ (2-20)

$\hat{y} = \theta^T \varphi(i)$ (2-17)

$\varepsilon(t) = y(i) - \theta^T \varphi(i)$ (2-21)

$\varphi(t) = [-y(i-1) - y(i-2) \ldots - y(i-n_a)\ u(i-1) \ldots u(i-n_b)]^T$ (2-22)

dengan:

θ = parameter estimasi

ε = estimasi galat

y(t) = keluaran nyata

ŷ(t) = estimasi keluaran

φ = vektor regresi

Untuk meminimalkan ε (t) maka berlaku

$(\theta, Z_t) = \sum_{i=1}^{t} [y(i) - \varphi T(i)\theta]^2$ (2-23)

Sehingga

$\hat{\theta}_t = \arg\min V_t(\theta, Z^t) = [\sum_{i=1}^{t} \varphi(i-1)\varphi^T(i-1)]^{-1} \sum_{i=1}^{t} \varphi(i-1)y(i-1)$ (2-24)

Dimana

$F(t)^{-1} = \sum_{i=1}^{t} \varphi(i-1)\varphi^T(i-1)$ (2-25)

Formula di atas merupakan persamaan yang belum berulang atau rekursif. Untuk menjadikan persamaan tersebut menjadi persamaan yang rekursif atau berulang-ulang, maka menjadi:

$\hat{\theta}(t+1) = F(t+1) \sum_{i=1}^{t+1} \varphi(i-1)(y(i)$ (2-26)

$F(t+1)^{-1} = \sum_{i=1}^{t} \varphi(i-1)\varphi^T(i-1) = F(t)^{-1} + \varphi(t)\ \varphi(t)^T$ (2-27)

$\hat{\theta}(t+1) = \hat{\theta}(t) + \Delta\hat{\theta}(t+1)$ (2-28)

Dari persamaan 4.14, dengan menambahkan $(t)\ \varphi(t)^T \hat{\theta}(t)$, akan didapatkan:

$\sum_{i=1}^{t+1} y(i)(i-1) = \sum_{i=1}^{t+1} \varphi(i-1) + y(t+1)\varphi(t) \pm \varphi(t)\varphi^T(t)\hat{\theta}(t)$ (2-29)

Dari persamaan-persamaan di atas, maka dapat dituliskan persamaan:

$\sum_{i=1}^{t+1} \varphi(i-1)y(i) = F(t+1)^{-1}\hat{\theta}(t+1)$ (2-30)

$$\sum_{i=1}^{t+1} \varphi(i-1)y(i) = F(t)^{-1}\,\hat{\theta}(t) + \varphi(t)\varphi^T(t)\hat{\theta}(t) + \varphi(t)\left[y(t+1)^{-1} - \hat{\theta}(t)^T\varphi(t)\right] \quad (2\text{-}31)$$

$$F(t+1)^{-1}\hat{\theta}(t+1) = F(t+1)^{-1}\hat{\theta}(t) + \varphi(t)\varphi^T\varepsilon(t+1) \quad (2\text{-}32)$$

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F(t+1)\varphi(t)\varepsilon(t+1) \quad (2\text{-}33)$$

Dalam mewujudkan persamaan rekursif untuk F(t), maka digunakan persamaan rekursif $F(t)^{-1}$ pada persamaan sebelumnya. Untuk persamaan tersebut berlaku *invers lemma*.

Dengan F matriks dimensi (n x n) dan φ vektor utuk dimensi *n* maka akan diperoleh:

$$(F^{-1} + \varphi\varphi^T)^{-1} = F - \frac{F\varphi\varphi^T F}{1+\varphi^T F\varphi} \quad (2\text{-}34)$$

Dengan persamaan sebelumnya, maka diperoleh:

$$F(t+1) = F(t) - \frac{F(t)\varphi(t)^T F(t)}{1+\varphi(t)^T F(t)\varphi(t)} \quad (2\text{-}45)$$

Untuk F adalah penguatan adaptasi, φ adalah *regression vector*, dan ε adalah prediksi galat. *Error* prediksi merupakan selisih output yang terukur dengan output hasil pemodelan.

Setelah model diestimasi maka dilakukan validasi. Validasi dapat dilakukan dengan tiga cara yaitu menggunakan *uncorrelation test*, *FPE* dan Fitness test. *Uncorrelation test* biasa disebut *whiteness test.* Yaitu dengan mencari RN(0) dan RN(i) dari galat prediksi. Dua hasil tersebut berasal dari:

$$R(0) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon^2(t) \quad (2\text{-}36)$$

$$RN(0) = \frac{R(0)}{R(0)} = 1 \quad (2\text{-}37)$$

$$R(i) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon(t)\varepsilon(t-i)\text{ , i=1,2,3,…,}i_{max} \quad (2\text{-}38)$$

$$RN(i) = \frac{R(i)}{R(0)}\text{ ; i=1,2,3,…,}i_{max} \quad (2\text{-}39)$$

N merupakan jumlah data uji.

Sifat *white* adalah sifat yang digunakan dalam *whiteness test* ini. Dengan hasil yang diinginkan adalah RN(0)=1 dan RN(i) =0. Hasil ini tidak mungkin terjadi. Dengan ketiak mungkinnya hal tersebut maka terdapat validasi yaitu:

$$RN(0) = 1 \quad (2\text{-}40)$$

$$|RN(i)| \leq \frac{2.17}{\sqrt{N}};\ i \geq 1 \quad (2\text{-}41)$$

Berikut tabel yang menunjukkan validasi pada *uncorrelation test*

Tabel 2. 1 Syarat / kriteria validasi *uncorrelation test*

Tingkat Siknifikan	kriteria	N=128	N=256	N=512
3%	$2.17/\sqrt{N}$	0.192	0.136	0.0096
5%	$1.94/\sqrt{N}$	0.173	0.122	0.087
7%	$1.808/\sqrt{N}$	0.133	0.113	0.08

Pada teori penyederhanaan oleh Landau, maka diberikan kriteria umum yaitu sebagai berikut:

$|RN(i)| \leq 0.15$ (2-42)

Untuk metode validasi selanjutnya yaitu Akaike's Final Prediction Error (Akaike's FPE). Yaitu metode validasi yang menggunakan error prediksi akhir. Kriteria Akaike Final Prediction Error (FPE) memberikan ukuran kualitas model dengan mensimulasikan model yangdiuji pada data yang berbeda. Setelah menghitung beberapa model yang berbeda, maka dapat membandingkannya menggunakan kriteria ini. Menurut teori Akaike, model yang paling akurat memiliki FPE terkecil.

Jika menggunakan kumpulan data yang sama untuk estimasi dan validasi model, kecocokan selalu meningkat saat meningkatkan urutan model yang diakibatkan fleksibilitas struktur model.

$$FPE = det\left(\frac{1}{N}\sum_1^N e(t,\hat{\theta}_N)(e(t,\hat{\theta}_N))^T\right)\left(\frac{1+d/N}{1-d/N}\right) \quad (2\text{-}43)$$

Dimana :

N = jumlah nilai dalam estimasi set data

$e(t)$ = adalah vektor prediksi error

$\hat{\theta}_N$ adalah parameter estimasi

d adalah jumlah parameter estimasi

dalam perangkat lunak yang digunakan, Jika jumlah parameter melebihi jumlah sampel, FPE tidak dihitung ketika estimasi model dilakukan (model.Report.FPE kosong). Perintah FPE akan mamberikan nilai kembalian NaN.

Validasi berikutnya adalah menggunakan uji *fitness*. Dalam kata lain, uji *fitness* juga disebut uji keakurasian. Keakuratan model respon dengan cara membandingkan beberapa input seperti step, kotak, dan prbs. Besaran akurat atau tidaknya dinyatakan dalam persen.

Jika nilai mendekati 100%, maka akan mendekati sistem yang sebenarnya. Persamaan untuk menghitung nilai fitness adalah sebagai berikut:

$$FIT = [1\text{-}NORM(Y\text{-}Y_{model})/NORM\ (Y - MEAN(Y)] *100 \quad (2\text{-}44)$$

Dimana:

FIT = nilai keakuratan (0-100%)

Y = nilai keluaran sistem hasil eksperimen

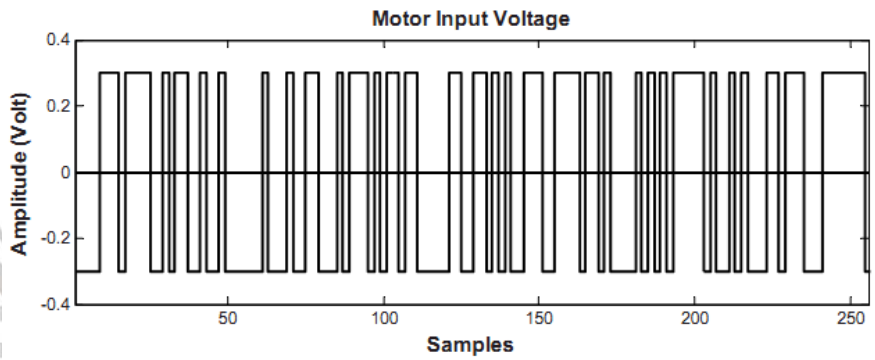
Y_{model} = nilai keluaran model

2.7.3 PRBS (*Pseudo-Random Binary Sequence*)

Pseudo-Random Binary Sequence, dikenal juga dengan nama *Maximal Length Sequence* (MLS), adalah sinyal periodik dan deterministik yang mirip seperti *white noise*. PRBS dapat dihasilkan dengan menggunakan n-bit shift register dengan umpan balik melalui fungsi XOR. Selama dilakukan proses random sinyal, urutannya akan diulang setiap nilai 2^n–1.

Ketika menggunakan keseluruhan dari periode, PRBS memiliki keuntungan matematis yang dapat mewujudkan sinyal stimulus dari sistem. Secara khusus, hal ini dapat mengaitkan variasi pada sinyal respons antara dua periode stimulus terhadap kebisingan karena sifat periodik sinyal. Seperti halnya *white noise*, PRBS memiliki faktor *crest* yang rendah Cf.

Berikut adalah contoh tampilan dari sinyal PRBS



Gambar 2. 17 Contoh input sinyal PRBS terhadap suatu sistem

Sumber: (Landau 2006)

2.8 Estimasi Kompleksitas Model

Untuk memahami estimasi kompleksitas model, terlebih dahulu harus diketahui model dalam bentuk diskrit pada sistem orde satu. Jika diasumsikan tidak ada noise, maka dapat ditulis persamaan:

$$y(t) = -a_1 y(t-1) + b_1 u(t-1) \quad (2\text{-}45)$$

Dan jika data tidak dipengaruhi noise, maka orde dari model adalah :

$$n = \max(n_A, n_B + d) = 1 \qquad (2\text{-}46)$$

Hipotesis untuk memastikan bahwa orde model telah bernilai benar berdasarkan metode menuliskan persamaan dalam matriks yang dibentuk dari variabel *delay* pada baris kedua dan seterusnya:

$$\begin{bmatrix} y(t) \vdots -y(t-1)\ u(t-1) \\ y(t-1) \vdots -y(t-2)\ u(t-2) \\ y(t-2) \vdots -y(t-3)\ u(t-3) \end{bmatrix} = [Y(t)\ R(1)]$$

$$Y(t) \qquad R(1) = R(\hat{n}) \qquad (2\text{-}47)$$

Matriks diatas dapat dipisah menjadi dua bagian, kolom pertama dinyatakan Y(t) adalah vektor yang terdiri dari output saat ini dan sebelumnya. Kolom sisanya dinyatakan dengan *R(1)*. Jika struktur dari persamaan 2-39 bernilai benar (hubungan u dan y), maka salah satu bisa digantikan dengan kolom pertama pada persamaan 2-40 yang dibentuk dari persamaan 2-39. Persamaan tersebut didapatkan dengan mengevaluasi determinan dari matriks pada persamaan 2-40.

$$rank[Y(t)R(1)] = rank \begin{bmatrix} -a_1 y(t-1) + b_1 u(t-1) \vdots -y(t-1)\ u(t-1) \\ -a_1 y(t-2) + b_1 u(t-2) \vdots -y(t-2)\ u(t-2) \\ -a_1 y(t-3) + b_1 u(t-3) \vdots -y(t-3)\ u(t-3) \end{bmatrix} = 2(<3)$$

(2-48)

Jika kolom pertama adalah kombinsi linier antara dua kolom lainnya, secara efektif Y(t) = R(1)θ dengan θ^T = *[a1,b1]* dan rank dari matriks adalah 2 (Sebagai ganti dari 3). Hal tersebut menghasilkan tes rank matrik dari persamaa 2-40 (ukuran determinan *non-null* terbesar) meloloskan validasi stuktur model yang sesuai dengan plant.

Estimasi *criterion* kompleksitas

Salah satu tujuan dari identifikasi sistem adalah untuk memperkirakan model yang memperkecil orde (prinsip *parsimony*). Oleh karena itu tujuannya untuk menambah kriteria persamaan :

$$J_{IV}(\hat{n}) = \min_{\hat{\theta}} \frac{1}{N} \left\| Y_{IV}(t) - R_{IV}(\hat{n})\hat{\theta} \right\|^2 \qquad (2\text{-}49)$$

Hal tersebut bisa dari bentuk:

$$S(\hat{n}, N) = \frac{2\hat{n}\log(N)}{N} \qquad (2\text{-}50)$$

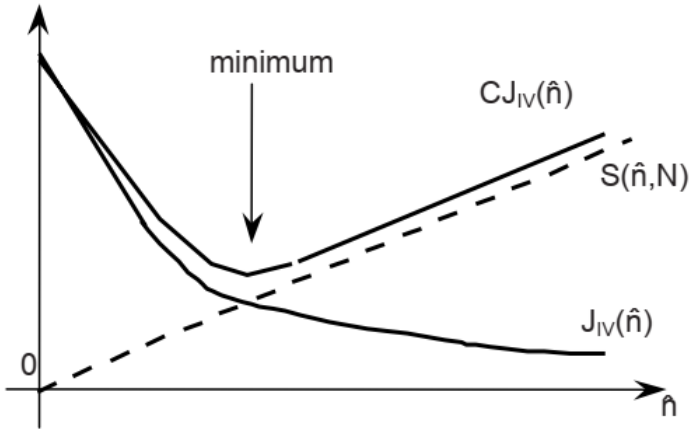
Salah satu criterion yang didefinisikan untuk estimasi model adalah

$$CJ_{IV}(\hat{n}) = J_{IV}(\hat{n}) + S(\hat{n}, N) \qquad (2\text{-}51)$$

Dimana n adalah nilai yang diberikan untuk meminimalisasi *criterion* dari persamaan di atas.

$$\hat{n} = \hat{n}* \rightarrow CJ_{IV}(\hat{n}) = min \qquad (2\text{-}52)$$

Estimasi kompleksitas yang diperoleh dengan cara ini menghasilkan nilai konsisten, yang berarti diperoleh urutan yang tepat karena jumlah data cenderung tak terhingga (N → ∞). Kurva evolusi *criterion* estimasi orde model dapat diamati pada Gambar 2.18.



Gambar 2. 18 Evolusi *criterion* estimasi orde model

2.9 KK Board 2.0

KK Board merupakan salah satu jenis kontroler *quadcopter* yang memiliki IMU (*Inertial Measurement Unit*) Sensor berupa *Accelerometer* dan *Gyro* sensor. Kontroler ini tidak memiliki sensor kompas, ketinggian maupun GPS. Sehingga dalam hal praktis, kontroler ini sebagian sensornya digantikan oleh indra manusia yaitu mata. Variabel sensor yang dapat diumpan-balikkan oleh KK Board 2.0 yaitu *roll* dan pitch atau sinyal kontrol *aileron* dan *elevator* untuk mengatur kemiringan dan keseimbangan *quadcopter* saat terbang. Variable ini menetukan kecepatan *quadcopter* bergerak ke depan-belakang maupun samping kanan-kiri. Beberapa setting parameter terdapat pada menu KK Board 2.0 yaitu

Tabel 2. 2 Menu Setting Parameter KK Board 2.0

No	Menu	Sub Menu	Keterangan
1.	PI Editor	Roll - Pitch - Yaw Axis Penguatan P Batas P Penguatan I Batas I	Untuk mengatur gain P dan I axis roll dan pitch

2.	Receiver Test	Aileron (-100 ~ 100) Elevator (-100 ~ 100) Throttle (0 ~ 100) Rudder (-100 ~ 100) Auxilary (-100 ~ 100)	Untuk verifikasi data input receiver tiap kanal
3.	Mode Setting	Self Level (Stick/Aux) Link Roll Pitch (Y/N) Auto Disarm (Y/N) CPPM Enable (Y/N)	Untuk mengatur trigger self level
4.	Stick Scaling	Roll (Ail) Pitch (Elev) Yaw (Rud) Throttle	Untuk mengatur agresifitas input
5.	Misc. Setting	Min Throttle Height Dampening Height D. Limit Alarm 1/10 volt Servo Filter	-
6.	Self - Level Setting	Penguatan P Batas P	Untuk mengatur gain saat self level aktif
7.	Camera Stab Setting	Servo Gain - Roll Pitch Gimbal	-
8.	Sensor Test	Gyro XYZ value Acc XYZ value	Untuk mengamati nilai sensor saat board digerakkan
9.	Acc Calibration	Calibrate Sensor on Flat Position	Untuk kalibrasi nilai sensor
10.	CPPM Setting		
11.	Mixer Editor	‘*image’*	Untuk mengatur sudut-sudut antar aktuator
12.	Show Motor Layout	Motor Position & Direction	Untuk mengamati posisi motor dan arah putaran
13.	Load Motor Layout	Single Copter Dual Copter Tri Copter Y6 Quad + Quad X	Untuk memilih mode multicopter
14.	Debug	-	
15.	Factory Reset	-	Mengembalikan ke setting awal

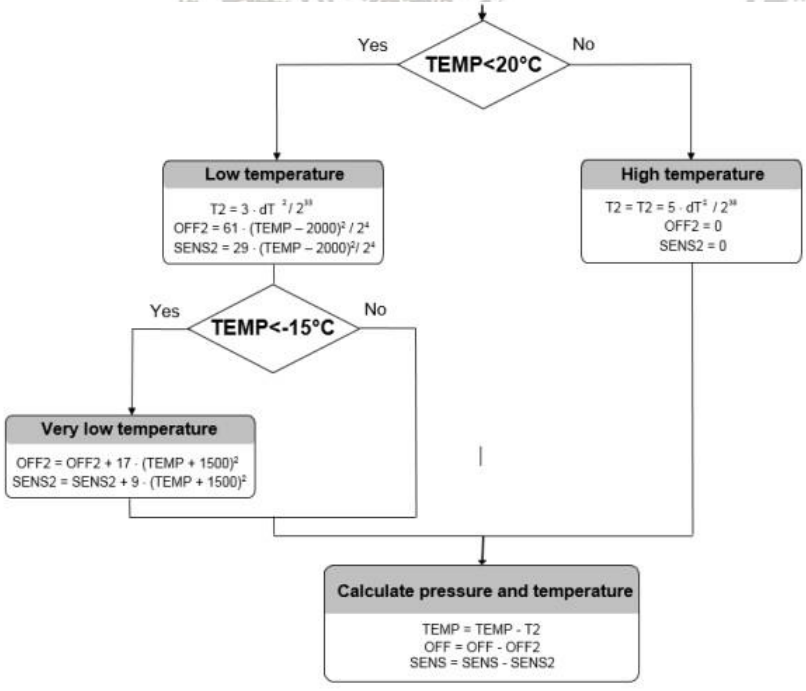
Parameter yang perlu diatur dalam penelitian ini adalah, *load motor layout*, *acc calibration*, PI *editor* dan *receiver test*. Keempat bagian dalam pengaturan KK Board tersebut dilakukan secara berurutan sebelum *quadcopter* diuj terbang manual terlebih dahulu.

2.10 Sensor Barometer MS5637

Salah satu komponen untuk mendeteksi ketinggian adalah barometer. Tekanan udara menjadi besaran utama yang dideteksi oleh sensor tipe MS5637. Spesifikasi Sensor MS5637 berdasarkan *datasheet* adalah sebagai berikut:

- Resolusi pemacaan ketinggian 13 cm
- Tegangan input 1.5 sampai 3.6 V
- Waktu konversi sebesar 0.5 ms
- Daya 0.6 µA (standby ≤ 0.1 µA at 25°C)
- Keluaran digital / menggunakan ADC (24 bit ΔΣ ADC)
- Range operasi 300 sampai 1200 mbar, -40 sampai +85 °C
- Antarmuka I2C
- Tidak memerlukan komponen luar (*internal oscillator*)

Dari spesifikasi tersebut terdapat satu variabel yang berpengaruh terhadap pembacaan tekanan udara yaitu suhu. Maka di dalam sensor MS5637 terdapat sensor suhu yang digunakan untuk mengkonversi pembacaan tekanan udara dengan suatu metode yang dijelaskan dalam Gambar 2.19



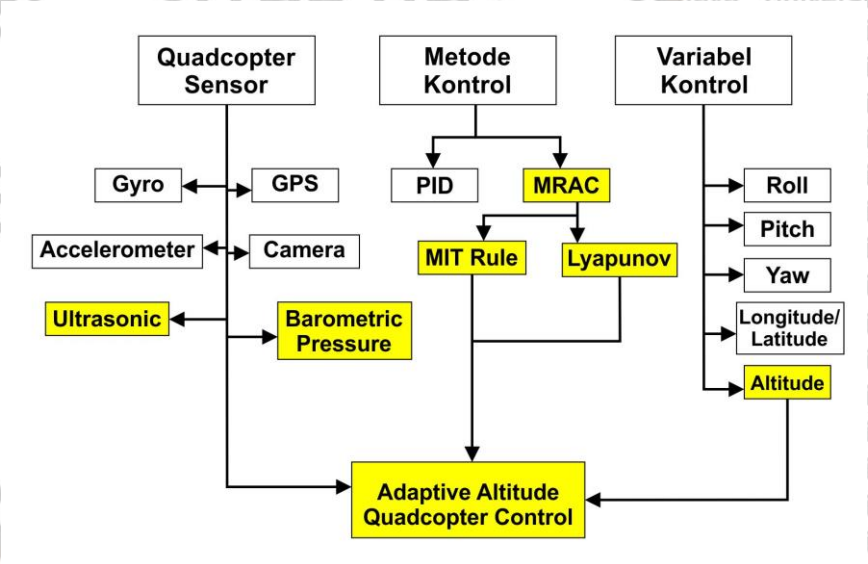
Gambar 2. 19 Flowchart pembacaan tekanan udara dan suhu MS5637

Dalam Gambar 2.19 terdapat pilihan untuk mengukur tekanan udara saat temperature rendah dan temperature tinggi dengan batas 20°C dan 15°C. Jika kurang dari batas-batas yang ditentukan, maka terdapat formula kompensasi sesuai dalam Gambar 2.19

BAB III
KERANGKA KONSEP PENELITIAN

3.1 Kerangka Pikir Penelitian

Suatu sistem *quadcopter* memiliki beberapa sudut pandang penelitian pada kontrol dinamiknya. Dari latar belakang yang telah ditulis, beberapa peneliti membahas tentang kendali *roll*, *pitch* dan *yaw* menggunakan sistem IMU (*Inertial Measurement Unit*). Beberapa peneliti juga menggunakan GPS untuk sensor posisi. Namun penulis memilih untuk fokus pada kontrol *altitude* / ketinggian. Gambar berikut menjelaskan tentang fokus penelitian tersebut.



Gambar 3. 1 Kerangka Pikir Penelitian

Berdasarkan kerangka tersebut, penelitian akan fokus kepada kontrol ketinggian *quadcopter* dengan metode MRAC. Sensor ketinggian yang digunakan adalah sensor ultrasonik. Meskipun sensor utama yang digunakan adalah ultrasonik atau *barometric pressure*, namun bagian dari IMU yaitu *accelerometer* juga akan diamati sesuai dengan sifat *accelerometer* yang merespon gerakan translasional. Pengamatan pada *accelerometer* dapat memberikan kontribusi pada pembacaan sensor secara gabungan.

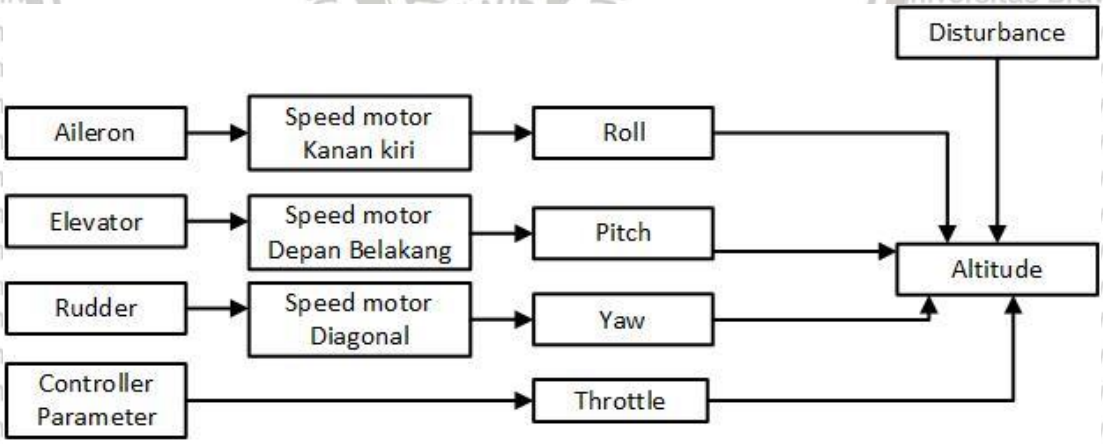
3.2 Hipotesis

Dari hubungan keterkaitan *peubah* riset yang telah dipaparkan sebelumnya, maka dapat diambil hipotesis yaitu:

1. Kontrol ketinggian menggunakan adaptif dapat menyesuaikan parameter kontrol berdasarkan ketinggian *quadcopter* dengan gangguan tertentu
2. *Quadcopter* dengan implementasi adaptif dapat dikendalikan secara *free-throttle* (otomatisasi ketinggian) dengan performa lebih baik dari kontrol non adaptif
3. Implementasi kontrol non adaptif akan menyebabkan perbedaan karakteristik respon *altitude* pada level ketinggian yang berbeda diakibatkan factor seperti densitas udara, daya baterei, dll

3.3 Definisi Operasional dan Pengukuran Peubah

Berdasarkan kerangka pikir penelitian tersebut maka ada beberapa peubah yang akan mempengaruhi proses maupun hasil dari penelitian. Peubah penelitian tersebut dapat dibagi menjadi dua tiga bagian yaitu peubah dari internal *quadcopter* (*altitude* / ketinggian), peubah dari eksternal *quadcopter* (gangguan dari luar), variabel kontrol Meskipun dalam fokus penelitian, penulis menekankan pada masalah *altitude*, naum tidak menutup kemungkinan bagian kerangka pikir yang lain juga termasuk dalam faktor yang mempengaruhi hasil penelitian seperti *roll*, *pitch*, *yaw* dan posisi. Hal tersebut dapat terjadi karena kendali gerak multicopter merupakan kendali yang terintegrasi dari keempat rotor dimana gerak rotor mempengaruhi *throttle* sehingga juga mempengaruhi ketinggian. Keterkaitan peubah dari penelitian ini sesuai dalam diagram berikut:



Gambar 3.2 Hubungan keterkaitan antar variabel penelitian

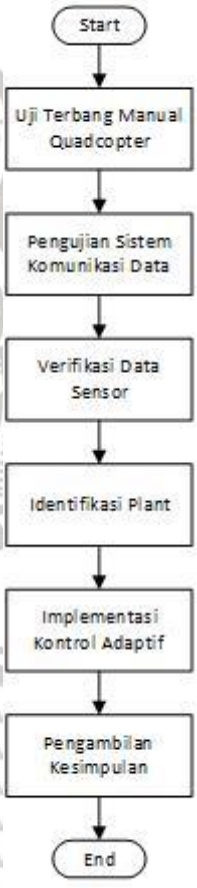
Bagian blok kontrol MRAC dan sinyal *throttle* merupakan variabel utama sebagai kendali *altitude quadcopter*. Namun gangguan/ *disturbance* serta parameter lain juga dapat mempengaruhi ketinggian. Faktor tersebut selanjutnya akan diamati korelasi terhadap *altitude* menggunakan metode analisis penelitian.

BAB IV
METODE PENELITIAN

4.1 Alur Penelitian

Untuk melaksanakan penelitian, terlebih dahulu dibuat alur penelitian agar penelitian yang dilakukan lebih terstruktur. Sebelum melakukan implementasi kontrol adaptif, terlebih dahulu dilakukan rancang bangun *quadcopter* sebagai objek penelitian, pembuatan sistem komunikasi data, pembuatan sistem pembacaan sensor, serta pengujian metode identifikasi *plant* yang akan diimplementasikan.

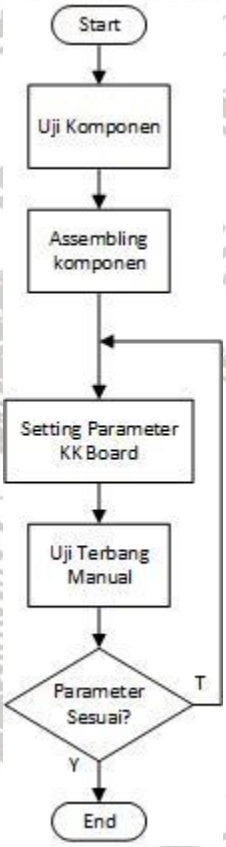
Secara garis besar, tahapan penelitian yang dilakukan tersebut dapat divisualisasikan dalam Gambar 4.1.



Gambar 4. 1 Diagram Alir Penelitian

4.1.1 Tahapan Uji Terbang Manual

Dalam diagram alir Gambar 4.1 terdapat beberapa tahapan yang dapat diuraikan menjadi sub-sub tahapan penelitian. Masing-masing tahapan tersebut harus dilakukan dengan tujuan meminimalisasi perubahan parameter baik berupa gangguan internal maupun eksternal sistem. Masing-masing alur penelitian tersebut dapat ditunjukkan dalam Gambar 4.2.



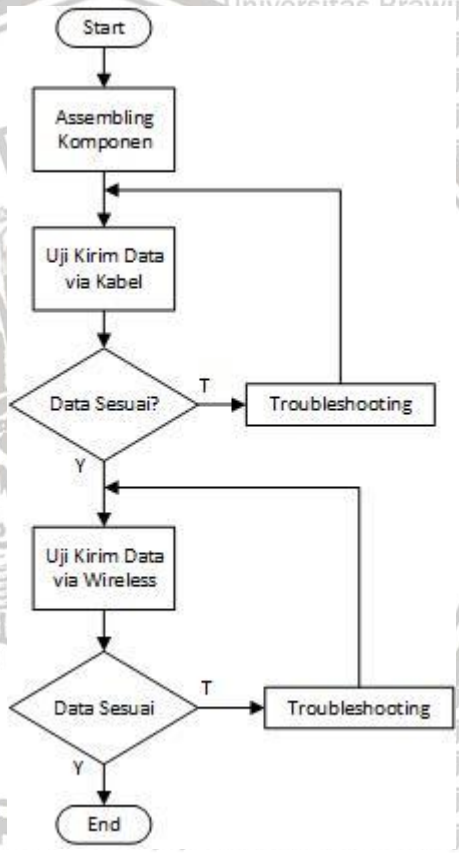
Gambar 4. 2 Diagram alur rancang bangun *quadcopter*

Dalam Gambar 4.2 terdapat beberapa tahapan dalam rancang bangun *quadcopter*. Tahapan yang paling penting adalah pada saat *setting* parameter KK Board. Seluruh *setting* yang dilakukan harus didokumentasikan dan tidak boleh berubah agar model plant nantinya tidak berubah pada saat implementasi kontrol. KK Board 2.0 memiliki *setting*/pengaturan pada beberapa parameter yang sangat berpengaruh pada model yang akan diidentifikasi. Karena penelitian ini fokus pada kontrol ketinggian, maka parameter yang paling berpengaruh adalah mode terbang dan setting konstanta PI untuk gerakan roll dan pitch. Dengan setting yang baik, maka pola terbang manual juga akan lebih stabil. Pada setting roll

pitch, apabila konstanta P maupun I terlalu besar, maka akan menimbulkan gerakan osilasi pada salah satu axis. Dan sebaliknya, jika konstanta terlalu kecil, maka akan mempersulit kontrol manual karena kurangnya peran sensor IMU (Inertial Measurement Unit) dalam mempertahankan *quadcopter* pada posisi dan orientasi tertentu.

4.1.2 Tahapan Komunikasi Data

Komunikasi data adalah bagian yang penting dari sistem kontrol *quadcopter*. Dengan adanya tahapan uji komunikasi data maka data yang dikirim dan diterima dapat diverifikasi kebenarannya sehingga mengurangi kesalahan baca dan kirim data yang berasal dari internal sistem itu sendiri. Tahapan komunikasi data tersebut sesuai dengan Gambar 4.3.

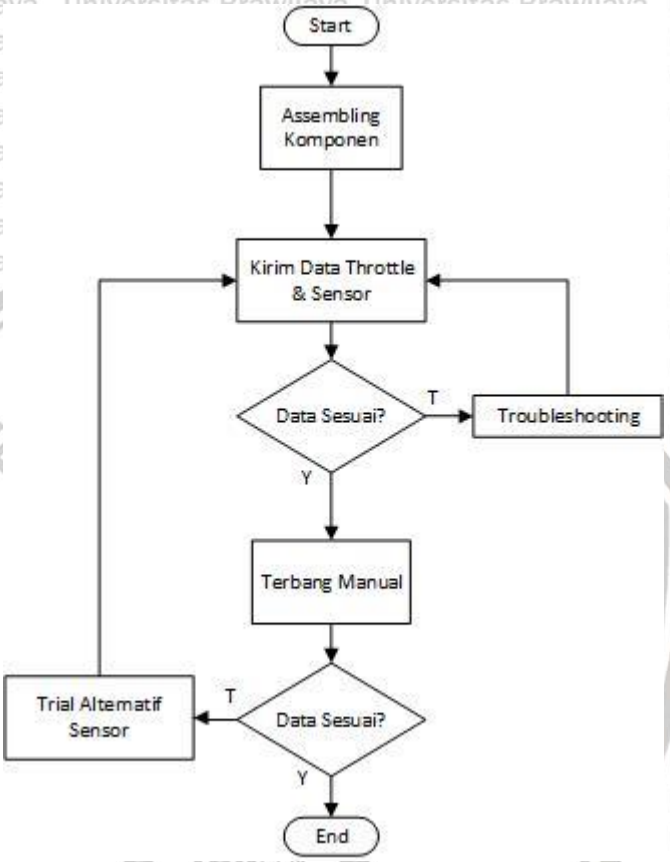


Gambar 4. 3 Diagram Alir Komunikasi Data

Dari Diagram Alir 4.3, komunikasi data dilakukan dalam dua tahap utama yaitu menggunakan kabel dan modul wireless. Hal tersebut dilakukan untuk verifikasi apakah data yang dikirim melalui kabel dan wireless dengan pengiriman data yang sama menghasilkan penerimaan data yang sama pula pada kedua metode komunikasi data. Modul wireless yang digunakan pada frekuensi 915MHz.

4.1.3 Tahapan Verifikasi Data Sensor

Sensor yang digunakan belum tentu merepresentasikan data yang diinginkan. Maka perlu adanya tahapan verifikasi data sensor. Verifikasi ini dilakukan saat *quadcopter* dalam mode terbang manual. Hasil pembacaan ketinggian yang diperoleh harus sesuai dengan hasil yang diinginkan. Apabila terdapat galat pembacaan, nilai data sensor tersebut seharusnya masih dapat diproses dengan menggunakan metode *filter* seperti *high-pass* atau *moving-average*.

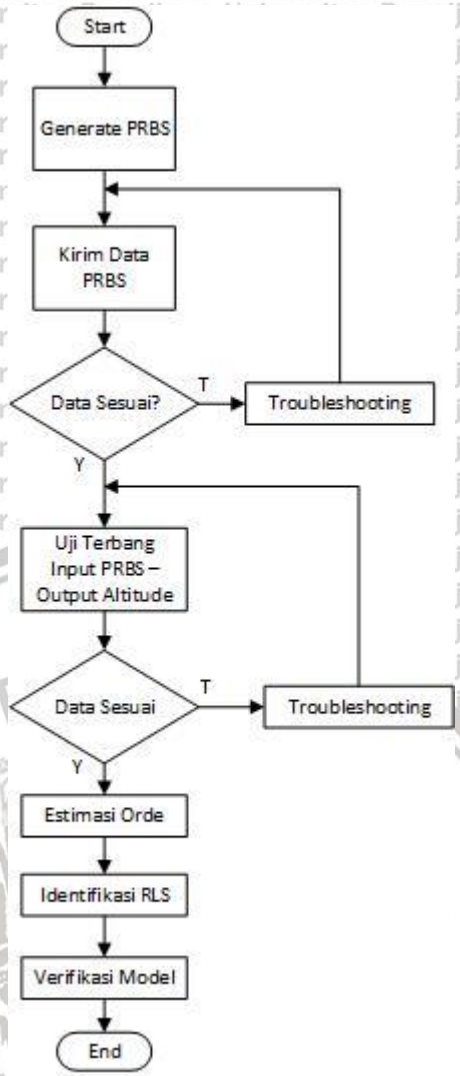


Gambar 4. 4 Diagram Alir Verifikasi Data Sensor

Dalam Gambar 4.4 terdapat bagian *trial alternative sensor*. Tahapan tersebut dilakukan apabila sensor yang digunakan sebelumnya tidak memenuhi syarat hasil pembacaan yang diinginkan atau terlalu banyak gangguan internal dari sistem. Dengan adanya alternatif sensor lain, maka hasil yang diharapkan juga berupa data pembacaan sensor yang sesuai untuk tahap selanjutnya.

4.1.4 Tahapan Identifikasi Plant

Untuk memodelkan plant dari sistem *quadcopter* saat terbang manual, diperlukan metode identifikasi. Metode ini menentukan seberapa tinggi orde plant dan mencari nilai parameter dari plant. Tahapan dari identifikasi plant *quadcopter* sesuai dalam Gambar 4.5.

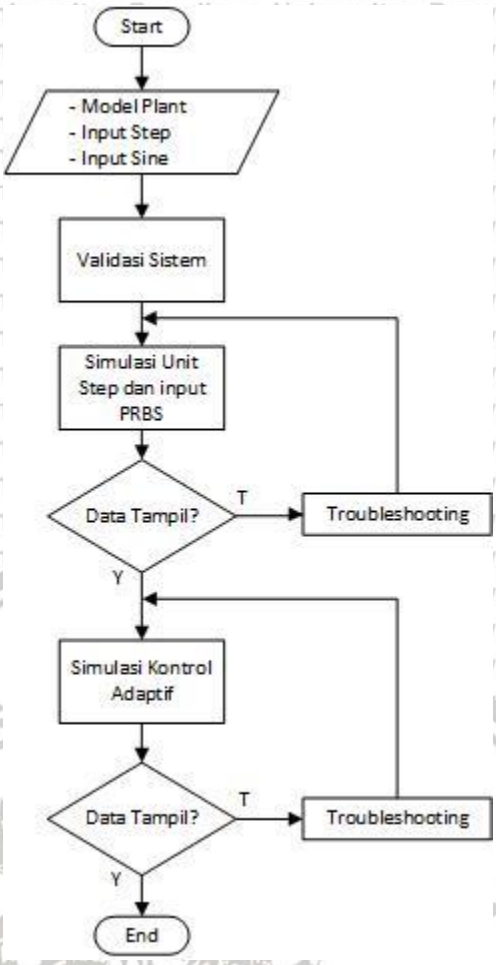


Gambar 4. 5 Diagram Alir Identifikasi Plant

Uji terbang menggunakan input PRBS menghasilkan output berupa pembacaan sensor ketinggian. Output yang diharapkan adalah hasil dari sinyal input yang kaya nilai fekuensi dimana dalam bidang kontrol, frekuensi adalah suatu besaran yang menentukan seberapa cepat sistem merespon input. Sehingga semakin kaya frekuensi yang digunakan sebagai input, maka model plant nantinya juga akan makin mendekati nilai sesungguhnya. Selanjutnya model ang dihasilkan akan diverifikasi dengan metode *whiteness test*.

4.1.5 Tahapan Implementasi Metode Kontrol

Metode kontrol akan diimplementasikan dalam bentuk simulasi kontrol. Input yng digunakan adalah dua macam sinyal yaitu unit step dan sinusoida. Sedangkan metode kontrol yang diimplementasikan adalah kontrol adaptif dengan Tuning menggunakan MRAC MIT Rule dan Lyapunov.



Gambar 4. 6 Diagram Alir Implementasi Kontrol

Dari Gambar 4.6 simulasi yang terlebih dahulu dilakukan adalah menggunakan input unit step. Input ini untuk memastikan bahwa plant *quadcopter* yang telah dimodelkan dapat dikontrol dengan PID konvensional dengan *set point* salah satu ketinggian. Untuk langkah implementasi yang kedua yaitu dengan menggunakan input sinusoida. Harapan keluaran yang dihasilkan adalah kontrol yang dihasilkan dapat mengikuti input ketinggian dengan fungsi tersebut. Beberapa nilai frekuensi juga akan dimplementasikan untuk mengetahui respon plant *quadcopter* pada nilai-nilai fekuensi tersebut.

4.2 Rancang Bangun *Quadcopter*

4.2.1 Bahan dan Alat

Bahan yang dipakai untuk penelitian adalah *quadcopter* dengan spesifikasi sebagai berikut:

1. *Frame* Aluminium 0.5 x 0.5 inch dengan radius diagonal motor 45 cm
2. *Propeller* SF 10X4.5 (baling-baling) CW dan CCW
3. KK 2.0 (papan kontrol utama *Quadcopter*)
4. Mikrokontroler ATMega328P

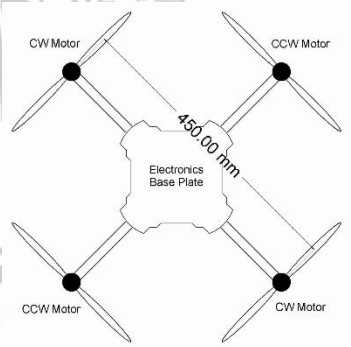
5. Motor BLDC Turnigy D2830 1000 KV
6. ESC Plush 30A (BLDC Driver)
7. *Battery* LiPo Turnigy 5000mAH
8. Module TX-RX Radio HK T6A
9. 3DR Radio *Telemetry* Kit 900MHz
10. Sensor Ultrasonik US 100
11. Sensor *Barometric Pressure* MS5637

Sebagai instrumen yang dibutuhkan untuk penelitian, digunakan alat-alat sebagai berikut:

1. PC untuk menampilkan data dari unit telemetri

4.2.2 Rancang Bangun *Frame*

Air frame merupakan konstruksi mekanik utama pembentuk *quadcopter* dengan bentuk dan ukuran yang bervariasi. *Quadcopter* memiliki dua macam konfigurasi yaitu mode '+' dan mode 'x'. Pada penelitian ini, mode konfigurasi yang dirancang adalah konfigurasi x. Konfigurasi X memiliki motor yang berputar dengan arah yang sama di setiap pasangan motor yang bersebrangan antar lengan *quadcopter*nya. Sehingga secara akumulatif, gaya aksi reaksi yang dihasilkan pada saat rpm yang sama adalah nol. Namun dengan karakter motor maupun komponen lain yang terdapat perbedaan, tidak menutup kemungkinan perbedaan karakter komponen tersebut menjadi gangguan yang nantinya juga akan masuk dalam parameter identifikasi sistem.



Gambar 4.7 Rancangan Airframe *Quadcopter*

Dari rancangan airframe, ukuran frame sesuai dengan salah satu jenis frame yaitu F450 yang ada di pasaran. Hal ini bertujuan agar skema kontrol dapat disesuaikan dengan frame *built-up* jenis 450mm. Peletakan rangka sesuai dalam Gambar 4.3 memudahkan komponen elektronika dalam mengatur posisinya.

4.2.3 Pengaturan Parameter KK Board

Parameter pada KK Board diperlukan pengaturan pada beberapa menu. Berdasarkan tabel menu KK Board yang telah dijelaskan pada Bab II, maka menu yang diatur dan diamati pada awal perancangan sebelum uji terbang antara lain:

1. Acc Calibration

   Sebelum board digunakan untuk *quadcopter*, terlebih dahulu diperlukan kalibrasi pada bidang datar sejajar permukaan bumi. Kalibrasi ini bertujuan untuk mengambil nilai sensor dan menjadikan nilai tersebut sebagai acuan pada saat *quadcopter* dalam posisi sejajar permukaan bumi. Sehingga pada saat uji terbang hingga implementasi kontrol, gangguan karena kesalahan setting point orientasi roll pitch dapat dikurangi seminimum mungkin.

Gambar 4. 8 Nilai Sensor pada posisi *flat* saat kalibrasi selesai

2. Load Motor Layout

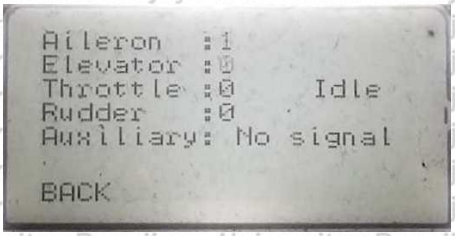
   Langkah berikutnya adalah mengatur mode layout motor pada *Quadcopter* (x) karena rangka dari *quadcopter* yang dirancang merupakan jens *quadcopter* dengan letak motor yang saling bersilangan antar pasangan motor CW dan CCW.

Gambar 4. 9 Pemilihan *Quadcopter* (x) pada layout

Untuk memastikan bahwa mode yang telah diset adalah mode *quadcopter* x, selanjutnya dilakukan pengamatan pada opsi View Motor Layout. Dari opsi tersebut, terlihat detail nomor motor dan arah putaran baling-baling / *propeller*. Jika dalam pengujian tanpa baling-baling terdapat kesalahan arah putar motor, maka terlebih dahulu harus dilakukan perubahan koneksi kabel motor ke ESC dengan menukar salah satu koneksinya. Hal tersebut sama persis seperti yang dilakukan pada pembalikan arah putar motor AC 3 fasa.

3. Receiver Test

Menu Receiver test digunakan untuk memastikan tiap kanal memperoleh sinyal dengan arah yang benar dari transmitter *joystick*. Hal ini sangat penting untuk mengetahui sinyal yang diterima oleh receiver sebelum dilakukan uji terbang.

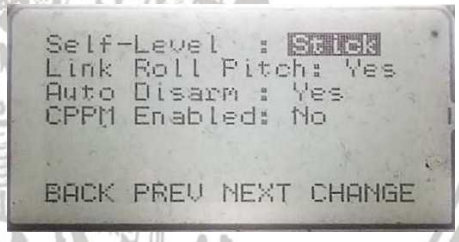


Gambar 4. 10 Tampilan nilai tiap kanal *receiver*

Pada menu receiver test ini terdapat lima bagian utama yang menampilkan nilai masing-masing kanal. Untuk kanal aileron, elevator dan rudder, nilai default yaitu nilai tengah dengan angka 0 sedangkan nilai maksimum adalah 100 dan nilai minimum adalah -100. Untuk kanal throttle, menampilkan nilai minimum 0 dan maksimum 100. Karena kanal throttle digunakan untuk mengubah kecepatan sudut dari keseluruhan motor *quadcopter*.

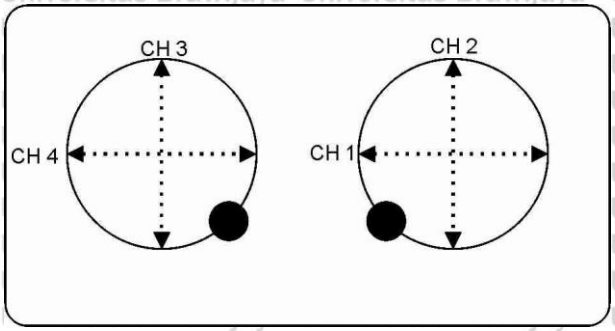
4. *Self Level* :

Untuk menu *Self Level* digunakan *stick mode*. Alasan menggunakan mode ini karena kanal yang digunakan terbatas dan mengurangi resiko kesalahan baca PWM untuk *trigger* PRBS mode pada mikrokontroler.



Gambar 4. 11 Pemilihan mode stick pada *self-level setting*

Untuk mengaktifkan *stick mode* untuk *self level* adalah dengan menggerakkan *stick* ch-1 ke kiri, ch-2 ke bawah, ch 3 ke bawah, ch 4 ke kanan.



Gambar 4. 12 Posisi *stick* untuk mengaktifkan mode *self-level*

5. Tuning PI

Untuk mengatur konstanta proporsional dan integral dari axis roll dan pitch, dilakukan setting PI pada KK Board. Untuk pengaturan besaran konstanta P dan I pada sumbu roll dan pitch, dilakukan dengan manual uji terbang.

Gambar 4. 13 Contoh tuning PI pada sumbu *aileron*

Cara manual yaitu dengan menerbangkan *quadcopter* pada nilai konstanta default. Pada saat dilakukan uji terbang, maka dapat dilakukan pengujian respon *quadcopter* pada perubahan kanal *aileron* dan *elevator*.

Metode yang paling sederhana pada *tuning* PI adalah mengamati *quadcopter* yang sedang *hovering*. Apabila pilot merasa sulit mengendalikan *quadcopter*, maka P Gain dan I Gain terlalu kecil. Apabila saat *hovering quadcopter* dapat bertahan namun terdapat osilasi gerakan baik frekuensi rendah maupun frekuensi lebih tinggi, maka P *Gain* dan I *Gain* terlalu besar.

Nilai Gain pada tuning PI yang paling tepat adalah jika joystick pada saat *quadcopter* melakukan *hovering* dapat dilepas dalam jangka waktu beberapa saat dengan hanya sedikit penyesuaian arah translasional.

4.2.4 Uji Terbang Manual

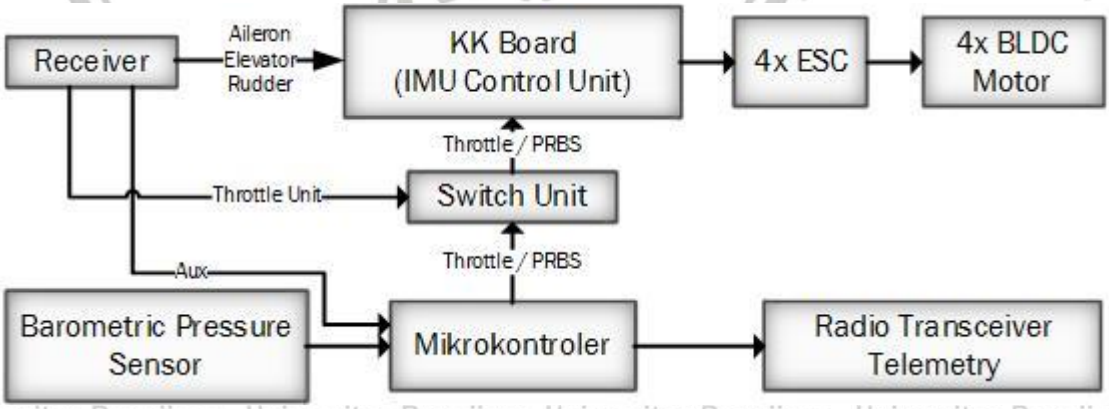
Uji terbang untuk pertama kalinya / *maiden flight* pada *quadcopter* dilakukan pada kondisi gangguan seminimum mungkin terutama gangguan angin. Untuk lokasi uji terbang dipilih lapangan outdoor dan tidak beratap. Hal ini disebabkan pada uji terbang, pola terbang dari *quadcopter* belum dapat dipastikan akan sesuai dengan yang diinginkan. Sehingga masih terdapat kemungkinan pola terbang yang sulit dikendalikan pada tiap axis-nya. Lokasi untuk uji terbang adalah di lapangan utama Politeknik Kota Malang Jl. Raya Tlogowaru no 3.

Gambar 4. 14 Lokasi Uji Terbang Manual *Quadcopter*

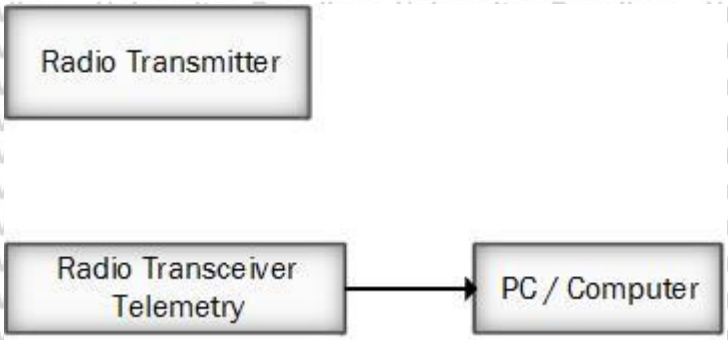
Uji terbang manual ini meliputi tuning PI pada KK Board, uji pengiriman data sensor dan sinyal throttle, serta respon ketinggian *quadcopter* terhadap perubahan sinyal throttle secara manual. Pengujian dikatakan sukses jika dengan gangguan minimum, *quadcopter* dapat terbang dan melakukan *hovering* serta melakukan perubahan gerakan translasional pada sumbu z. Dalam hal ini sumbu vertikal tegak lurus permukaan bumi.

4.3 *Telemetry* Data Sensor dan Sinyal *Throttle*

Data primer menggunakan sensor ketinggian dan sinyal input *throttle* dari *joystick* dimana input dan output ketinggian tersebut nantinya akan dijadikan data utama untuk proses identifikasi plant kontrol ketinggian *quadcopter*. Dalam komunikasi data, metode yang digunakan harus dengan cara nirkabel. Untuk blok diagram proses komunikasi data tersebut sesuai dalam diagram blok berikut.

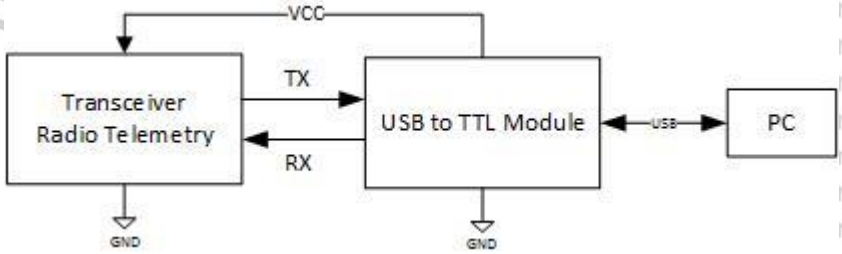


Gambar 4. 15 Blok Diagram *Air Module* Sampling Data Sensor dan Sinyal *Throttle* untuk Pemodelan Plant



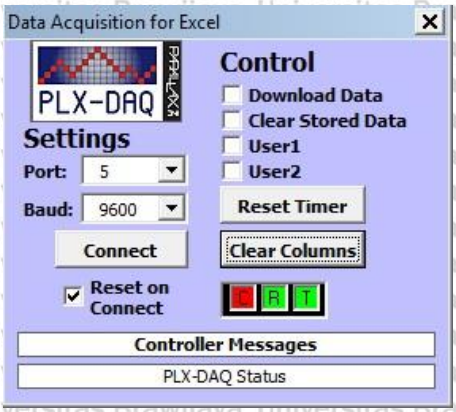
Gambar 4. 16 Blok Diagram *Ground Module* Sampling Data Sensor dan Sinyal *Throttle* untuk Pemodelan Plant

Dari Gambar 4.15 dan 4.16 terdapat dua modul yaitu *air module* dan *ground module*. Pada sisi komunikasi data sensor dan sinyal input, kedua bagian ini dihubungkan dengan *transceiver radio telemetry* berupa modul *wireless* komunikasi data pada frekuensi kerja 900MHz. Tipe yang digunakan adalah 3DR *telemetry module* 900MHz. protokol komunikasi yang digunakan adalah UART pada baudrate 57600 bps. Pengkabelan dari module ini menggunakan 4 kabel penghubung utama dengan 2 kabel jalur daya dan 2 kabel transmit dan receive (TX dan RX).



Gambar 4. 17 *Wiring Telemetry* pada *Ground Module*

Pada sisi ground module, *data logger* menggunakan software PLX DAQ. Software ini digunakan pada *active-X* Microsoft Excel, sehingga menghasilkan tabel hasil pembacaan masing - masing data sensor ketinggian dan lebar pulsa sinyal *throttle*.



Gambar 4. 18 Panel kontrol PLX-DAQ pada Microsoft Excel

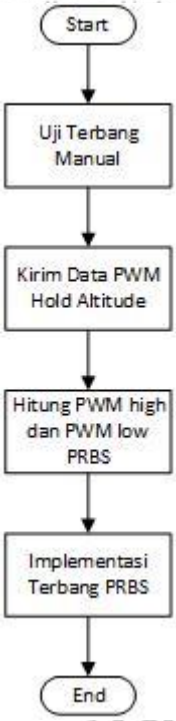
Pada panel kontrol PLX-DAQ terdapat beberapa parameter yang perlu diatur antara lain port untuk mengaktifkan jalur COM serial dan baud rate untuk sinkronisasi kecepatan pengiriman data. Pada baudrate diset 57600 karena baudrate telemetry module bekerja pada frekuensi tersebut.

4.4 Identifikasi Sistem

Identifikasi pada penelitian ini bertujuan untuk memperoleh persamaan model plant dari input sinyal throttle joystick dan output sensor ketinggian. Input berupa lebar pulsa / duty cycle dari unit pembangkit sinyal PRBS. Sedangkan output berasal dari sensor ketinggian yang nantinya akan diamati menggunakan sensor dengan berbasis sinyal ultrasonik atau *barometric pressure sensor*. Setelah didapat nilai input-output berikutnya adalah melakukan estimasi kompleksitas model. Estimasi ini bertujuan menentukan orde tertinggi dari model plant. Setelah didapatkan orde plant, maka dapat dilakukan pemodelan dengan menggunakan *Recursive Least Square* (RLS). Sebelum model digunakan dalam implementasi kontrol, langkah berikutnya adalah melakukan validasi model dengan metode *whitening* dari prediksi *error*.

4.4.1 Pembangkit Sinyal PRBS

Telah dijelaskan pada bab tinjauan pustaka bahwa *pseudo random binary sequence* (PRBS) digunakan sebagai sinyal input dari proses identifikasi sistem. Dalam implementasi pada sinyal throttle sesungguhnya, sinyal PRBS dihasilkan oleh mikrokontroler dengan mengaktifkan PWM pada dua nilai duty cycle saat *quadcopter* merespon naik dan turun. Langkah-langkah yang dilakukan untuk memperoleh nilai duty cycle tersebut sesuai dengan diagram alir implementasi sinyal PRBS.



Gambar 4. 19 Diagram alir implementasi terbang dengan input PRBS

Data *hold altitude* merupakan lebar pulsa PWM sinyal throttle pada saat *quadcopter* dapat menahan ketinggian tertentu pada saat uji terbang manual. Setelah didapatkan nilai sinyal throttle tersebut selanjutnya saat terbang manual, input throttle digantikan oleh sinyal PRBS dengan high PWM dan low PWM sebagai binary sequence-nya. High PWM dan low PWM diperoleh dengan menambah dan mengurangi nilai *hold altitude* dengan nilai tertentu.

Jumlah sampling periode yang digunakan pada sinyal PRBS adalah 2^8-1 = 255 periode. Metode yang digunakan adalah menggunakan perintah angka acak untuk memperoleh nilai yang acak pada data 256 bit. Untuk tipe data yang digunakan adalah tipe data array *unsigned integer* dengan jumlah 256 bit dibagi 16 bit yaitu berjumlah 16 data dalam *array*. Time sampling T_S yang digunakan sebesar 100ms. Pemilihan durasi maksimum PRBS tergantung dari respon *rise time* yang diperoleh dari kendali manual *quadcopter*.

4.4.2 Data Input-Output

Sinyal input adalah sinyal throttle, yaitu sinyal yang menentukan kecepatan keseluruhan motor berputar. Efek dari throttle adalah mengubah ketinggian *quadcopter* pada saat *quadcopter* diam maupun bergerak dalam posisi horizontal.

Untuk langkah pemrosesan sinyal output ketinggian dilakukan dengan tujuan memperoleh data kecepatan vertikal *quadcopter*. Hal tersebut sesuai dengan karakteristik input yang perubahan nilai *duty cycle* PWM-nya mengakibatkan perubahan pada kecepatan

vertikal *quadcopter*. Bukan ketinggian. Sehingga langkah untuk mengubah ketinggian menjadi kecepatan vertikal dilakukan dengan menggunakan persamaan:

$$v_{ts} = \frac{h_{ts} - h_{(ts-1)}}{t_{ts} - t_{(ts-1)}} \quad (4\text{-}1)$$

Dimana:

v_{ts} adalah kecepatan vertikal *quadcopter* saat t sampling

h_{ts} adalah ketinggian saat sampling

$h_{(ts-1)}$ adalah ketinggian sampling sebelumnya

t_{ts} adalah pewaktu saat sampling

$t_{(ts-1)}$ adalah pewaktu saat sampling sebelumnya

4.4.3 Estimasi Orde Plant

Untuk menentukan orde plant, dilakukan metode least square. Algoritma ini menggunakan Instrumental Variable (IV) dengan memasukkan nilai input PRBS dan output berupa pembacaan *vertical speed* pada *quadcopter*. Setelah nilai input output tersebut dimasukan, maka dengan instrumental variable didapatkan nilai orde plant yang nantinya akan digunakan untuk pemodelan plant menggunakan algoritma-algoritma tertentu.

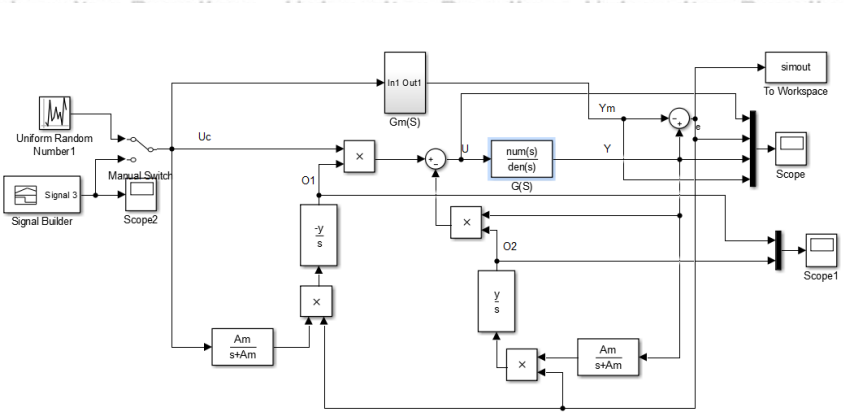
4.4.4 Pemodelan Plant

Untuk melakukan pemodelan plant terdapat beberapa metode yaitu *Recursive Least Square*, *Output Error Prediction*, dll. Untuk memodelkan sistem plant memungkinkan untuk menggunakan lebih dari satu metode. Hal ini disebabkan pada saat validasi, bisa didapatkan nilai hasil validasi yang lebih besar dari batas yang diperbolehkan, sehingga harus dicoba kembali dengan metode yang berbeda

4.5 Implementasi MRAC

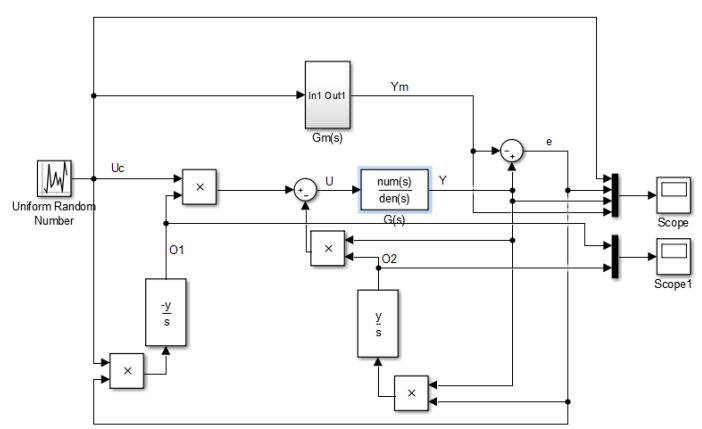
Pada implementasi MRAC dilakukan dua metode yaitu MIT Rule dan Lyapunov. Kedua implementasi ini menggunakan blok Simulink dengan memasukkan parameter yang telah divalidasi. Setelah Simulink dijalankan, maka akan didapatkan nilai-nilai parameter kontrol yang baru / *update*.

Blok diagram Simulink yang digunakan dalam MIT Rule adalah sebagai sesuai dalam Gambar 4.20.



Gambar 4. 20 Implementasi MIT Rule menggunakan simulink dari hasil pemodelan plant

Blok diagram Simulink yang digunakan dalam metode Lyapunov adalah sebagai sesuai dalam Gambar 4.21.



Gambar 4. 21 Implementasi Lyapunov menggunakan simulink dari hasil pemodelan plant

4.6 Analisis dan Kesimpulan / Diskusi

Kesimpulan diambil berdasarkan hasil pengujian dan pengolahan data. Dari tujuan penelitian dapat diasumsikan bahwa pada bab kesimpulan terdapat beberapa hal antara lain;

1. Parameter fungsi transfer output ketinggian *quadcopter* terhadap sinyal *throttle*
2. Hasil validasi pemodelan parameter plant
3. Perbandingan hasil simulasi MIT Rule dan Lyapunov
4. Grafik penurunan error yang dihasilkan dari kontrol adaptif untuk *quadcopter*

Dengan beberapa kesimpulan tersebut diharapkan dapat memenuhi kontribusi penelitian yang sebelumnya telah ditulis dalam bab pendahuluan. Tidak menutup kemungkinan terdapat bab diskusi yang ditimbulkan dari hasil penelitian ini. Diskusi dapat muncul ketika terjadi permasalahan pada saat pengujian dan pengambilan kesimpulan.

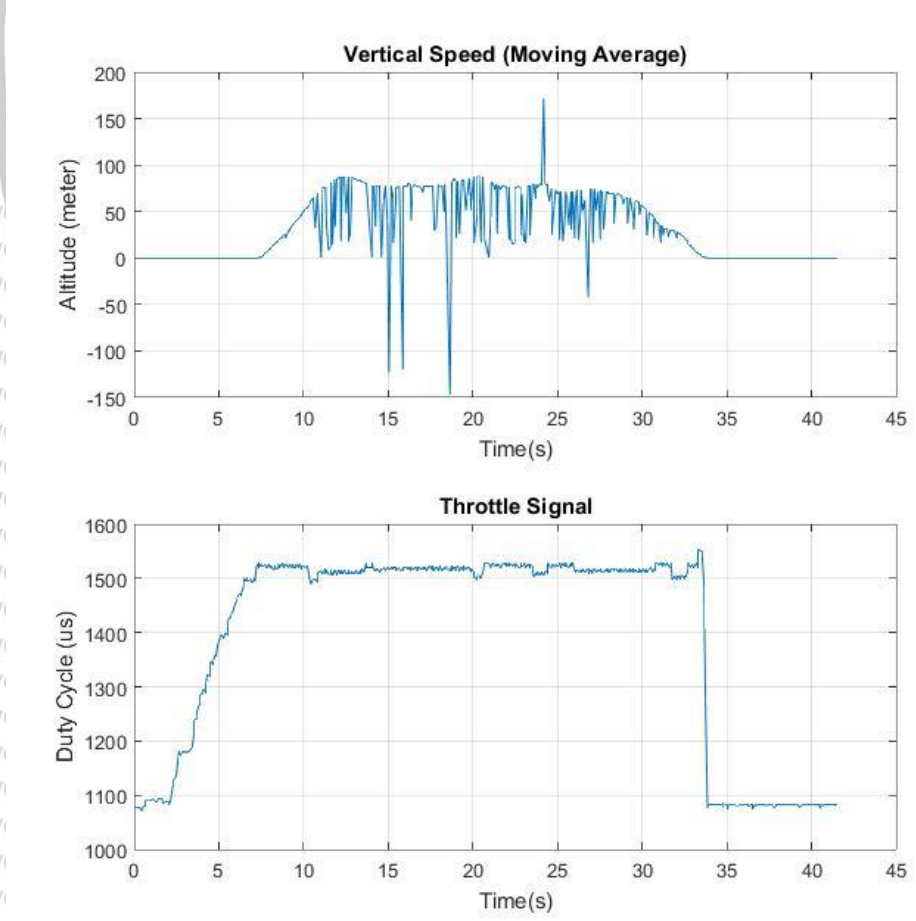
BAB V
HASIL DAN PEMBAHASAN

5.1 Pembacaan Sensor Ketinggian

Pembacaan ketinggian kedua sensor yang telah dijelaskan pada bab metode penelitian dipilih dengan hasil yang paling sedikit terdapat *error* pembacaan. *Error* pembacaan dapat berasal dari proses komunikasi data maupun internal sensor itu sendiri.

5.1.1 Sensor Ultrasonik

Dari proses rancang bangun quadcopter hingga uji terbang, dilakukan terlebih dahulu pengiriman data sensor ultrasonic untuk mendapatkan data ketinggian *quadcopter* dengan mengirim data input *throttle* dan output hasil konversi lebar pulsa yang didapatkan dari pembacaan sensor ultrasonik. Grafik hasil pembacaan input sinyal *throttle* dan *output* sensor ultrasonik untuk uji terbang adalah sebagai berikut.



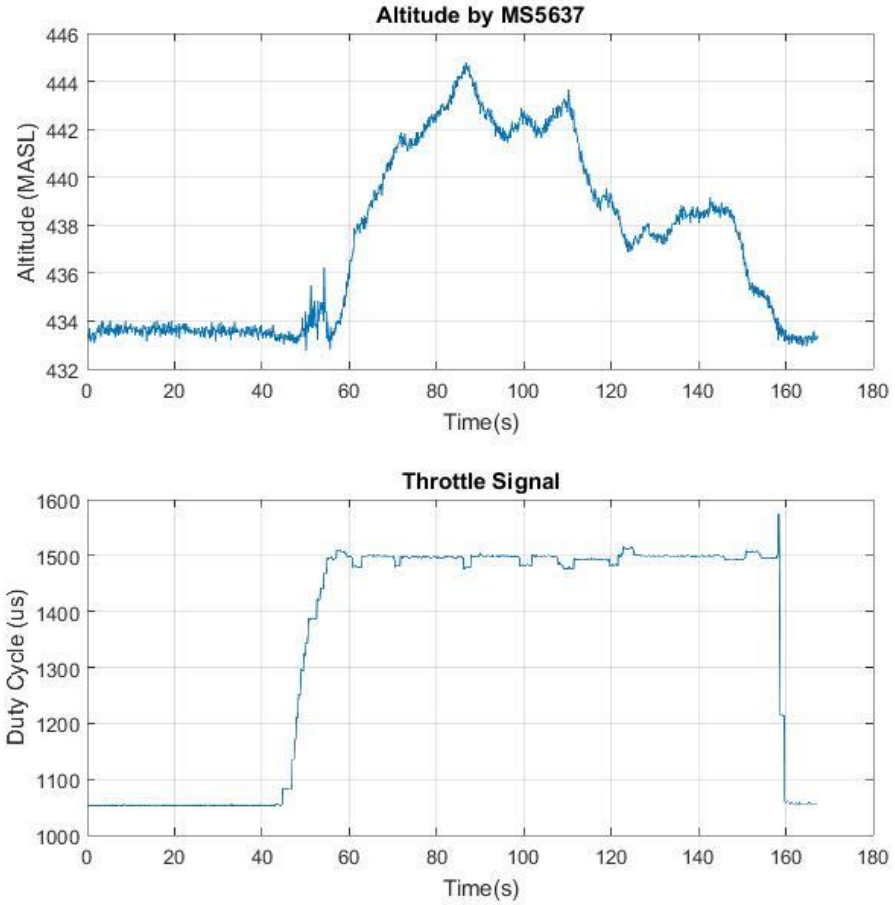
Gambar 5. 1 Grafik Pembacaan Sensor Ultrasonik dan Sinyal *Throttle*

Dari Gambar 5.1 terlihat bahwa hasil pembacaan sensor ultrasonik terdapat beberapa data yang tidak sesuai dengan kondisi nyata. Yaitu nilai data yang tiba-tiba berubah tidak *liniear* dan tidak sesuai dengan *time series* yang seharusnya.

Sehingga dari hasil grafik tersebut dapat disimpulkan bahwa penggunaan sensor ultrasonik memiliki dua kelemahan utama yaitu terbatasnya pembacaan ketinggian (3 meter) dan terdapat *error* pembacaan pada beberapa kali *sampling* data. Kesalahan baca sensor dapat disebabkan oleh kondisi permukaan tanah yang tidak memantulkan sinyal ultrasonik tegak lurus terhadap permukaan penerima sensor.

5.1.2 *Barometric Pressure Sensor*

Grafik hasil pembacaan input sinyal *throttle* dan output *barometric pressure sensor* untuk uji terbang adalah sebagai berikut.



Gambar 5. 2 Grafik Pembacaan Sensor MS5637 dan Sinyal *Throttle*

Dari Gambar 5.2 dapat dijelaskan bahwa mulai dari t = 0s adalah *quadcopter* masih berada di permukaan tanah dan belum *take off*. Hal tersebut ditandai nilai altitude sekitar 434m di atas permukaan laut cenderung diam. Setelah *quadcopter* takeoff sekitar t = 60s,

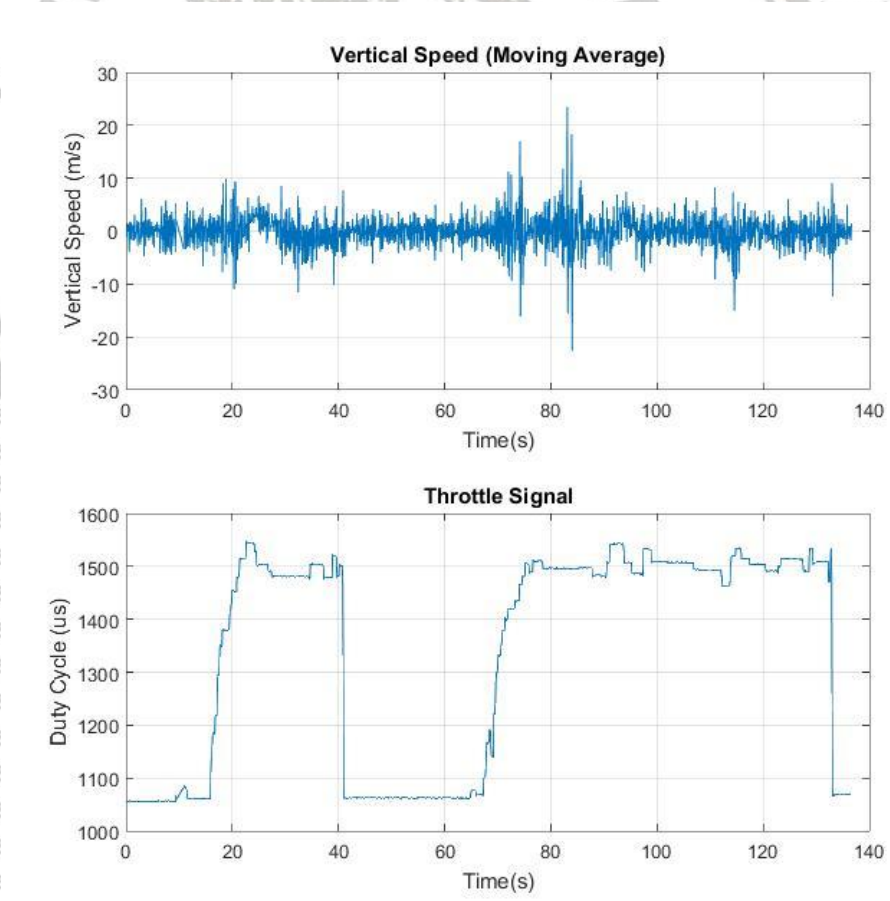
dapat diamati bahwa sinyal throttle *quadcopter* berada di sekitar 1500µs untuk mempertahankan ketinggian *quadcopter*. Sekitar t = 160s, ketinggian *quadcopter* adalah kembali lagi ke permukaan tanah, sehingga sinyal throttle harus segera diturunkan hingga sekitar 1000µs untuk menghentikan putaran *propeller quadcopter*.

Berdasarkan grafik pembacaan sensor MS5637, maka dapat disimpulkan bahwa sensor membaca ketinggian sesuai dengan *time series*. Ketika diberi input dengan sinyal *throttle*, maka ketinggian hasil pembacaan MS5637 juga mengikuti pola inputnya. Dengan demikian maka digunakan barometric pressure sensor untuk proses identifikasi dan pemodelan sistem.

5.1.3 Pemrosesan Sinyal Output Ketinggian

Pemrosesan sinyal ketinggian menjadi kecepatan vertikal dilakukan dua kali terbang dalam satu kali *sampling* data. Sehingga terlihat perubahan output terhadap input pada keduanya. Jika perubahannya memiliki kecenderungan sama, grafik tersebut dapat digunakan sebagai data untuk pross penelitian berikutnya.

Hasil dari pemrosesan sinyal ketinggian menjadi kecepatan vertikal *quadcopter* adalah sebagai berikut:



Gambar 5. 3 Grafik Pemrosesan Sinyal Ketinggian menjadi Kecepatan Vertikal

Dari hasil grafik pembacaan sinyal tersebut, dapat diamati bahwa sinyal hasil pemrosesan data belum dapat digunakan untuk validasi hubungan input output antara sinyal *throttle* dan *vertical speed*. Masih diperlukan adanya *filter* digital untuk menghilangkan sinyal output yang berfrekuensi tinggi, sehingga akan diperoleh pergerakan *time series* yang sesuai dengan sinyal inputnya.

Metode *filter* yang digunakan adalah menggunakan *moving average*. *Moving average* adalah metode untuk merata-rata pergerakan data dalam jangkauan tertentu. Dalam hal ini jangkauan yang digunakan untuk merata-rata nilai *moving average* adalah setiap 20 waktu *sampling*.

$$\hat{y}(t+1) = \frac{y(t)+y(t-1)+\cdots+y(t-n+1)}{m} \quad (5.1)$$

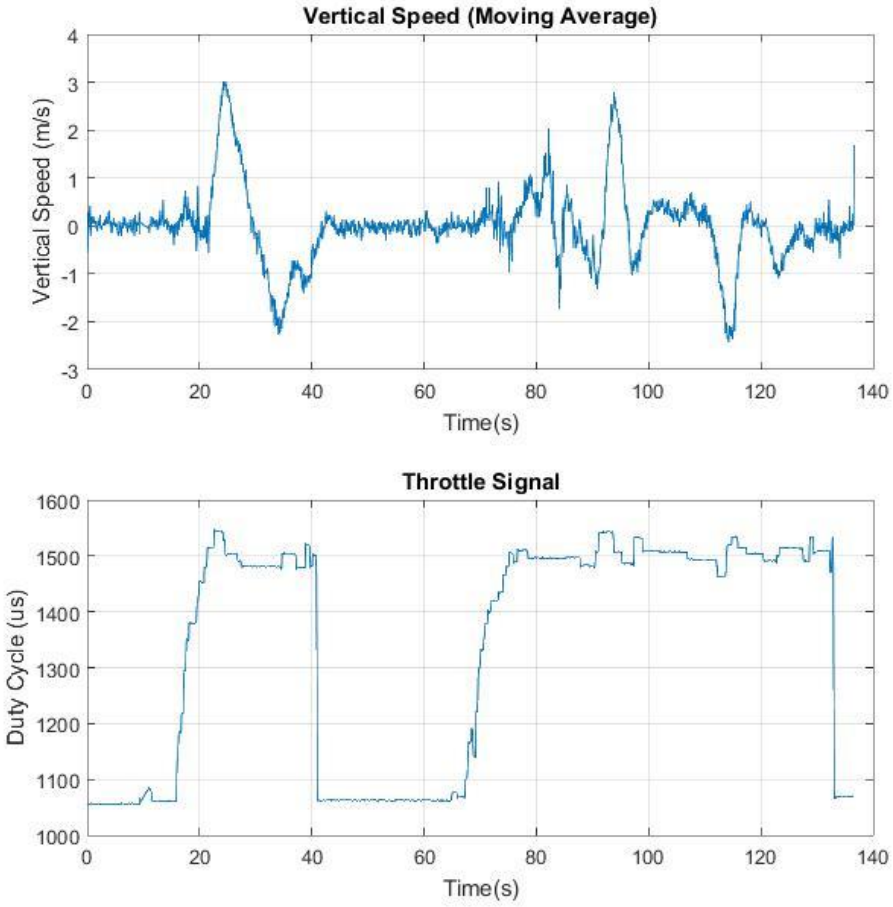
Dimana:

$\hat{y}(t+1)$ = nilai ramalan di waktu (t+1)

$y(t)$ = nilai sebenarnya di waktu t

m = banyaknya waktu dalam moving average

Dari implementasi persamaan *moving average*, maka diperoleh grafik hasil pemrosesan sinyal *vertical speed* dengan *moving average*.



Gambar 5. 4 Hasil Pemrosesan Sinyal *Vertical Speed* dengan *Moving Average*, *m*=20

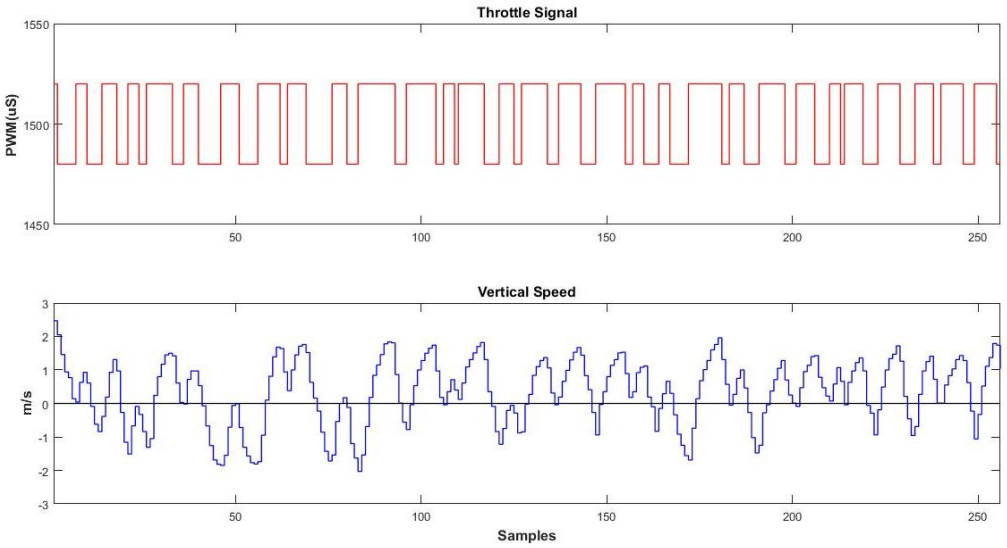
Dari hasil moving average maka dapat disimpulkan respon *output* pada grafik *vertical speed* dengan implementasi *moving average* telah menunjukkan perubahan yang sesuai dengan input. Jika ditinjau dari pada saat ketinggian *quadcopter* cenderung stabil / *vertical speed* mendekati nol, *input throttle* berada pada daerah sekitar 1500µs. Sehingga implementasi sinyal PRBS akan menahan nilai *set point* nol pada nilai sinal *throttle* tersebut.

Dari hasil sampling kontrol manual *quadcopter*, diperoleh data rata-rata *duty cycle* untuk input throttle pada saat hovering hingga ketinggian berubah sebesar 20us. Sehingga pada implementasi PRBS akan menggunakan perubahan binary sequence juga sebesar 20us untuk menjaga agar *quadcopter* masih dalam jangkaiuan dalam kendali dan tidak terjadi perubahan ketinggian yang berlebihan.

5.2 Implementasi PRBS

Setelah dilakukan beberapa tahap uji terbang manual dan validasi nilai sensor yang diperoleh, maka dihasilkan pembacaan sensor ketinggian untuk input sinyal menggunakan PRBS. Seperti dijelaskan pada Bab IV bahwa mikrokontroler menghasilkan sinal PRBS

dengan cara mengimplementasikan fungsi random pada sinyal yang akan diteruskan ke PWM *throttle* pada *quadcopter* receiver. Sinyal PRBS yang dihasilkan adalah sebagai berikut



Gambar 5. 5 Hasil uji sinyal PRBS vs OUTPUT sensor via UART, T=500ms/bit

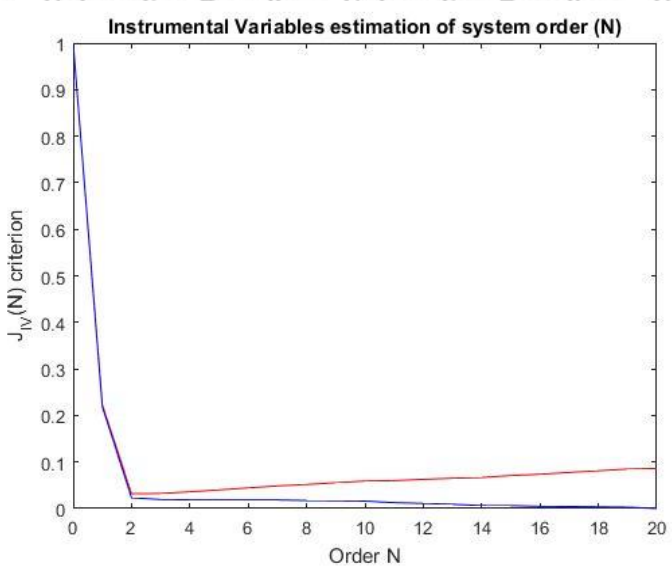
5.3 Identifikasi Sistem

Sesuai dengan diagram alir identifikasi sistem pada bagian awal bab metode penelitian, urutan yang dilakukan adalah estimasi orde plant, implementasi metode RLS, dan yang terakhir adalah validasi model. Hasil ketiga langkah ini akan dijelaskan pada bagian 5.3.1 sampai 5.3.3.

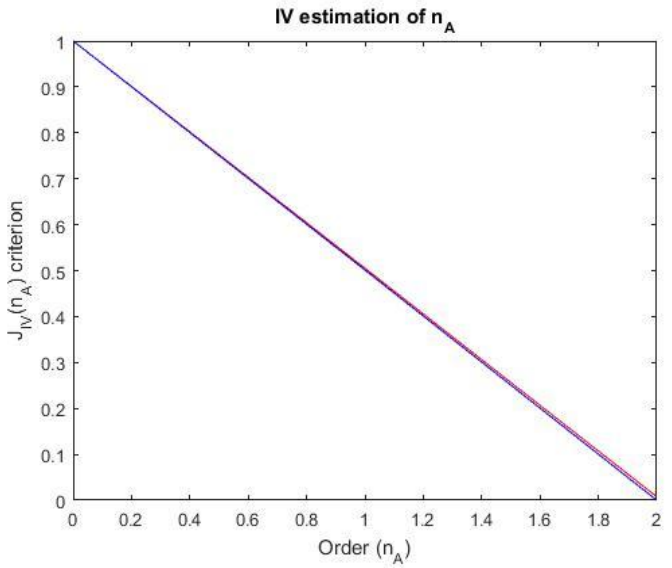
5.3.1 Estimasi Orde Plant

Estimasi order plant dilakukan dengan cara memasukkan hasil grafik input output penerapan sinyal PRBS ke dalam metode estimasi sebelum selanjutnya akan dilakukan pemodelan menggunakan metode yang telah ditentukan. Fungsi estimasi order yang digunakan adalah dari *estimation orde instrument variable* / estorderiv. Dengan parameter input minimal ada 2 yaitu sinyal output (y) dan sinyal input (u).

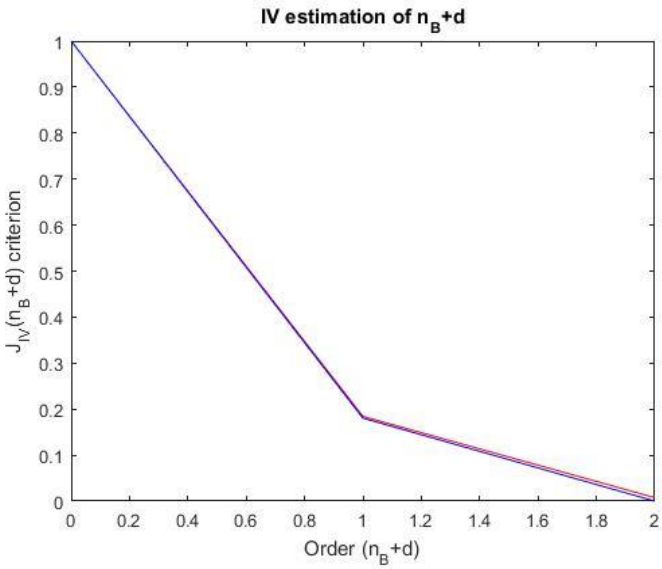
Hasil dari estimasi orde plant *quadcopter* adalah sebagai berikut:



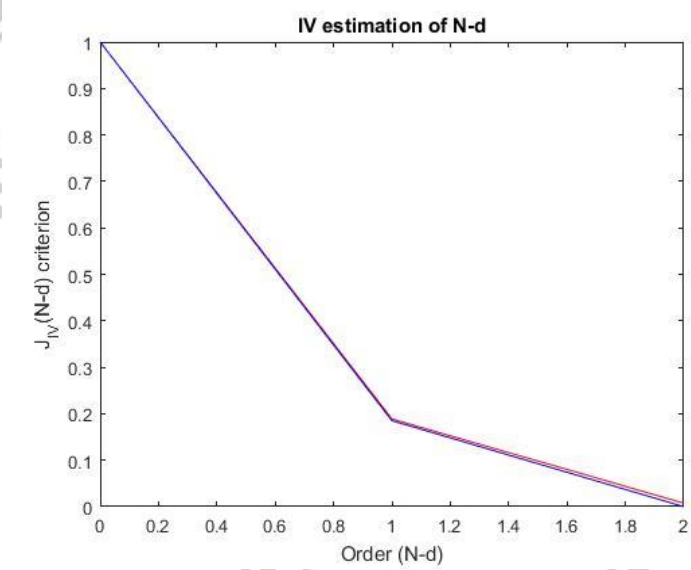
Gambar 5. 6 Estimasi Orde - Instrumental Variables Estimation of System order (N)



Gambar 5. 7 Estimasi Orde - IV estimation of NA



Gambar 5. 8 Estimasi Orde - IV estimation of nB+d



Gambar 5. 9 Estimasi Orde - IV estimation of N-d

Dari hasil estimasi orde plant *quadcopter* maka dihasilkan parameter sebagai berikut:

N = 2

d = 0

nB = 2

nA = 2

dan hasil estimasi orde plant adalah orde 2.

5.3.2 Pemodelan Plant dan Validasi

Untuk membuat model plant, pertama kali digunakan metode **RLS** (Recursive Least Square). Metode ini merupakan metode perulangan / rekursif yang mencari nilai bobot kuadrat terkecil. Hasil dari metode RLS adalah:

A1 = -1,2595

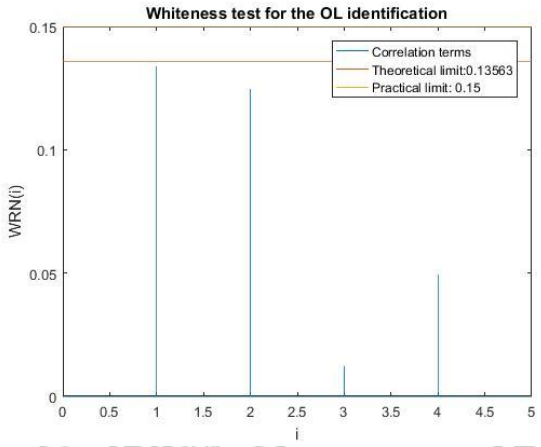
A2 = 0,4422

B1 = 0,0134

B2 = -0,0098

Fungsi transfer dalam domain waktu diskrit menjadi $A_{(z)} = 1 - 1{,}2595\, z^{-1} + 0{,}4422\, z^{-2}$ untuk numerator dan $B_{(z)} = 0{,}0134\, z^{-1} - 0{,}0098\, z^{-2}$ untuk denominatornya.

Setelah dihasilkan nilai A dan B maka dilakukan validasi menggunakan whiteness test dan uncorrelation test. Dari hasil validasi didapatkan:



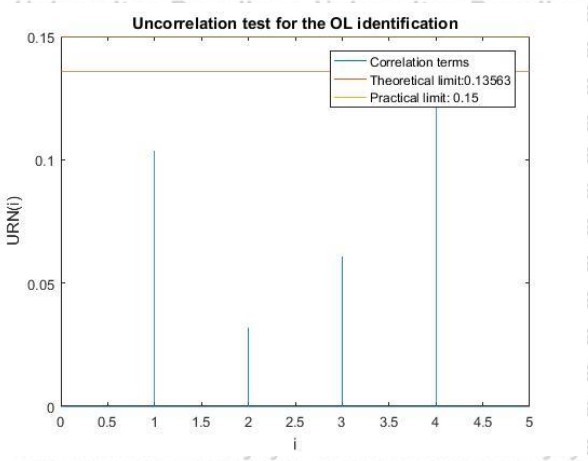
Gambar 5. 10 Hasil validasi whiteness test pada untuk identifikasi metode RLS

Hasil yang didapatkan dari validasi *whiteness test* adalah sebagai berikut:

wlossf = 0,1167

RN(0) = 0,1338 *RN(1)* = 0,1247 *RN(2)* = 0,0122 *RN(3)* = 0,0495

$|RN(i)| \leq 0{,}13563$ dengan batas praktikal yaitu sebesar $|RN(i)| \leq 0{,}15$. Sehingga dapat disimpulkan pengujian validasi sistem menggunakan *whiteness test* dinyatakan lolos.



Gambar 5. 11 Hasil validasi *uncorrelation test* pada untuk identifikasi metode RLS

Hasil yang didapatkan dari validasi *whiteness test* adalah sebagai berikut:

ulossf = 0,5857

RN(0) = 0,1035 *RN(1)* = 0,0319 *RN(2)* = 0,0604 *RN(3)* = 0,1294

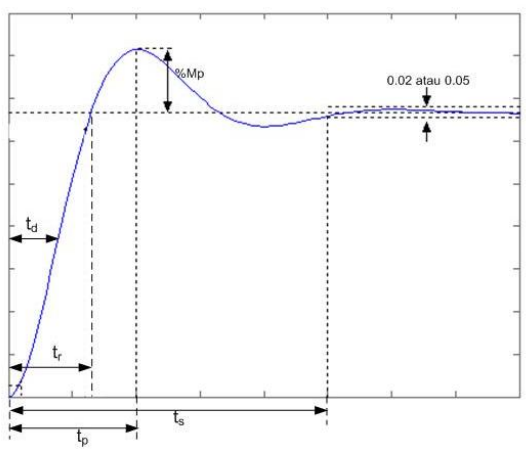
5.3.3 Fungsi Transfer dalam Bentuk Kontinyu

Dengan menggunakan fungsi *discrete to continuous* metode Tustin pada matlab (fungsi d2c / *discrete to continuous*), maka didapatkan nilai numerator dan denominator pada fungsi S. nilai tersebut yaitu:

$$Tf(s) = \frac{0{,}003477s+0{,}02155}{s^2+1{,}6320s+1{,}0930} \qquad (5\text{-}1)$$

5.4 Model Referensi

Untuk membuat model referensi yang diinginkan, terlebih dahulu berdasarkan acuan model dengan respon orde 2 terhadap masukan unit step. Terdapat beberapa parameter dalam respon sistem orde 2 menggunakan unit step yaitu *delay time*, *rise time*, *peak time*, *overshoot* maksimum (*Mp*), dan *settling time*. Masing-masing parameter respon sesuai Gambar 5.12



Gambar 5. 12 Respon sistem orde 2 dengan masukan unit step

Dari Gambar 5.12 maka dapat dinyatakan dalam persamaan fungsi transfer orde dua

$$\frac{C(s)}{R(s)} = \frac{K\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \quad (5\text{-}2)$$

Dimana:

K = konstanta gain

ω_n = Frekuensi alamiah

ξ = Rasio redaman

Sinyal yang diinginkan adalah yang memiliki parameter utama yaitu rasio redaman maks 5%, dan mencapai *Settling Time* dalam maksimum 100ms. Sehingga menggunakan persamaan:

$$\%Mp = e^{-\frac{\pi\xi}{\sqrt{1-\xi^2}}} \quad (5\text{-}3)$$

$$T_s = \frac{4}{\xi\,\omega_n} \quad (5\text{-}4)$$

Dari persamaan 5-3 didapatkan nilai ξ dengan inver eksponensial yaitu nilai ln.

ln 0.05 = -2,996

$$-2{,}996 = -\frac{\pi\xi}{\sqrt{1-\xi^2}}$$

Dengan memasukkan nilai π = 3,14 maka nilai faktor redaman ξ adalah 0,696.

Dari persamaan 5-4, didapatkan nilai ω_n yaitu :

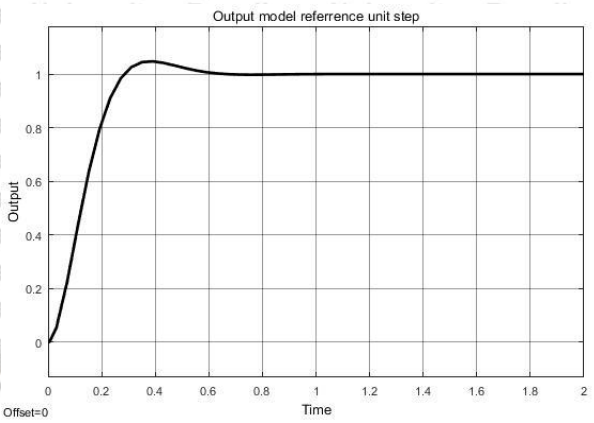
$$\omega_n = \frac{4}{\xi T_s} = \frac{4}{0{,}696\ .\ 0.1}$$

Sehingga diperoleh ω_n sebesar 11,494 rad/s

Dengan diketahui nilai rasio redaman dan frekuensi alamiah referensi model, maka dapat dituliskan fungsi transfer dari referensi model sebagai berikut:

$$\frac{C(s)}{R(s)} = \frac{\omega_n{}^2}{s^2+2\omega_n s+\omega_n{}^2} = \frac{132{,}11}{s^2+15{,}99+132{,}11} \quad (5\text{-}5)$$

Dari persamaan referensi model tersebut dapat ditampilkan grafik respon unit step sesuai dengan parameter model yang diinginkan. Grafik tersebut ditunjukkan dalam gambar 5.13



Gambar 5. 13 Respon referensi model terhadap input unit step

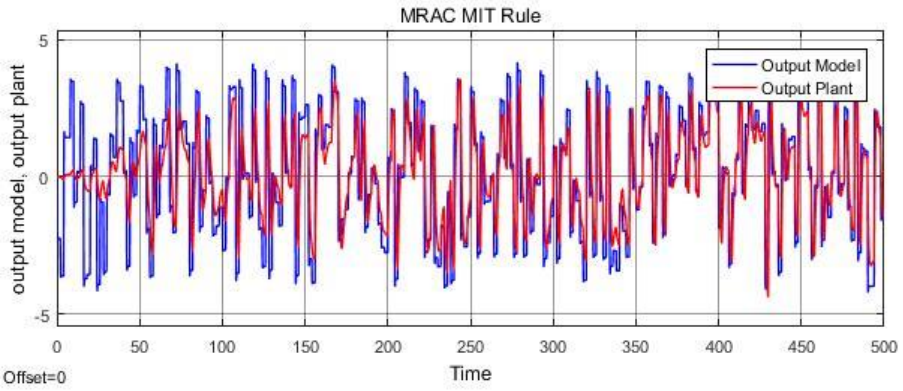
5.5 Implementasi Kontrol Adaptif

Untuk implementasi kontrol adaptif MRAC (*Model Reference Adaptif Control*) seperti yang telah dijelaskan pada BAB IV yaitu menggunakan 2 metode. Yang pertama diimplementasikan adalah metode MIT Rule dan berikutnya menggunakan metode Lyapunov.

5.5.1 MIT Rule

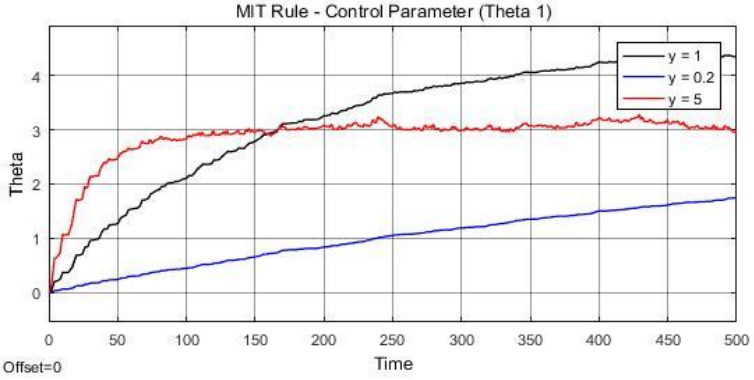
Metode MIT Rule dilakukan dengan menggunakan blok simulink dengan memasukkan numerator dan denominator dari fungsi transfer yang telah dihasilkan dari proses identifikasi sebelumnya.

Hasil dari simulasi metode MIT Rule Adalah sebagai berikut.



Gambar 5. 14 y dan y_m pada MRAC metode MIT Rule nilai $\gamma = 1$

Selisih output model dan plant sangat besar saat 50s pertama. Namun semakin mendekati 0 saat detik ke 150s. Hasil output sistem juga menunjukkan bahwa output dapat mengikuti model pada detik sekitar 150s.

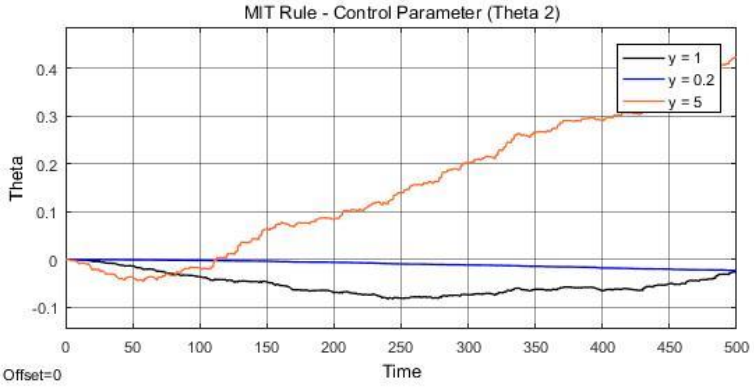


Gambar 5. 15 nilai θ_1 pada MRAC metode MIT Rule nilai $\gamma = 1$, $\gamma = 0.2$, $\gamma = 5$

θ_1, saat $\gamma = 1$ → 4,33

$\gamma = 0.2$ → 1,87

$\gamma = 5$ → 2,95



Gambar 5. 16 nilai θ_2 pada MRAC metode MIT Rule nilai $\gamma = 1$, $\gamma = 0.2$, $\gamma = 5$

θ_2 saat $\gamma = 1$ → -0,024

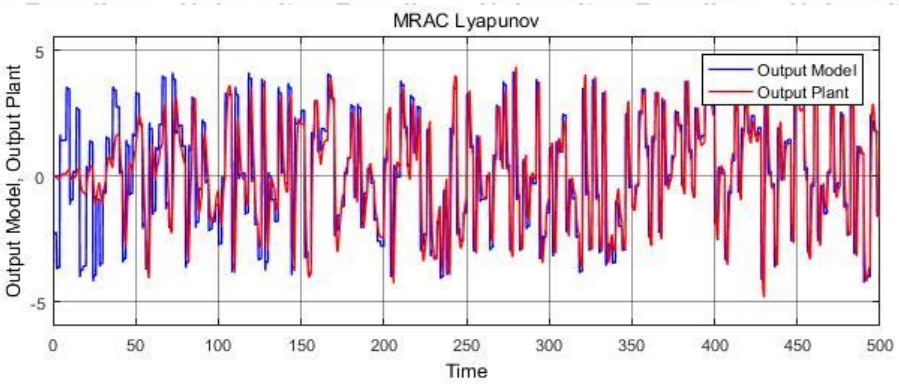
$\gamma = 0.2$ → -0,027

$\gamma = 5$ → 0,43

5.5.2 Lyapunov

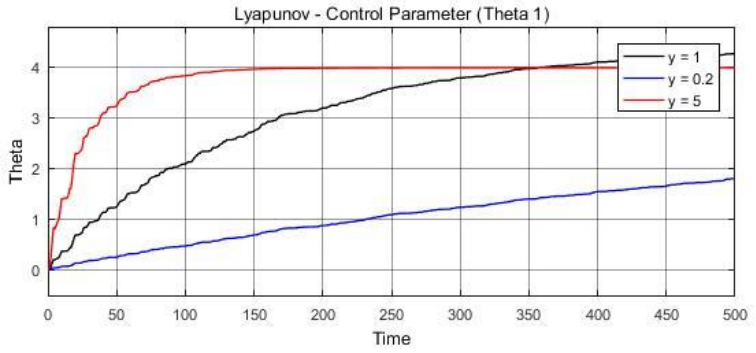
Metode Lyapunov dilakukan dengan menggunakan blok simulink dengan memasukkan numerator dan denominator dari fungsi transfer yang telah dihasilkan dari proses identifikasi sebelumnya.

Hasil dari simulasi MRAC menggunakan Lyapunov adalah sebagai berikut:



Gambar 5. 17 y dan y_m pada MRAC metode MIT Rule nilai $\gamma = 1$

Selisih output model dan plant sangat besar saat 40s pertama. Namun semakin mendekati 0 saat detik ke 70s. Hasil output sistem juga menunjukkan bahwa output dapat mengikuti model pada detik sekitar 70s.

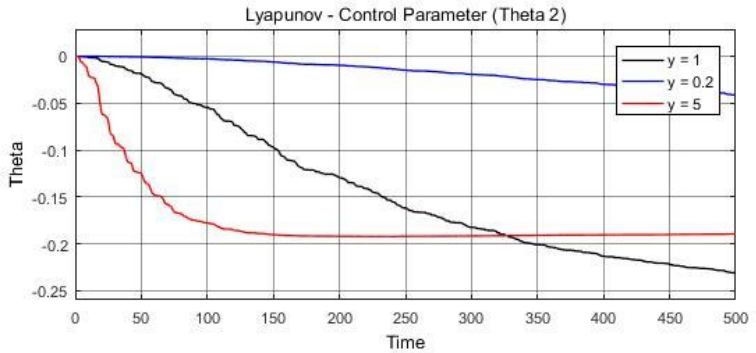


Gambar 5. 18 nilai θ_1 pada MRAC metode Lyapunov nilai $\gamma = 1$, $\gamma = 0.2$, $\gamma = 5$

θ_1, saat $\gamma = 1$ → 4,36

$\gamma = 0.2$ → 1,88

$\gamma = 5$ → 3,99



Gambar 5. 19 nilai θ_2 pada MRAC metode Lyapunov nilai $\gamma = 1$, $\gamma = 0.2$, $\gamma = 5$

θ_2 saat $\gamma = 1$ → -0,238

$\gamma = 0.2$ → -0,048

$\gamma = 5$ → -0.187

Dari hasil simulasi, maka dapat disimpulkan bahwa penggunaan metode Lyapunov lebih baik dari MIT Rule dalam sistem ini. Hal tersebut dapat diamati pada perubaha nilai error yang lebih cepat mendekati nilai nol saat menggunakan metode MRAC Lyapunov.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil penelitian kontrol adaptif terhadap kendali ketinggian *quadcopter*, diambil kesimpulan bahwa:

1. Pengambilan data kecepatan vertikan *quadcopter* dilakukan dengan menggunakan sensor *barometric pressure*
2. Parameter fungsi transfer yang didapatkan melalui metode recursive least square adalah A1 = -1.2595, A2 = 0.4422, B1 = 0.0134, B2 = -0.0098 dimana fungsi transfer diskritnya menjadi $A_{(z)} = 1 - 1.2595\, z^{-1} + 0.4422\, z^{-2}$ dan $B_{(z)} = 0.0134\, z^{-1} - 0.0098\, z^{-2}$
3. Hasil validasi menggunakan whiteness test dan uncorrelation test yaitu *RN(0)* = 0.1338 *RN(1)* = 0.1247 *RN(2)* = 0.0122 *RN(3)* = 0.0495 dengan $|RN(i)| \leq 0.13563$ kurang dari batas praktikal $|RN(i)| \leq 0.15$
4. Hasil metode MRAC yaitu untuk metode MIT Rule, parameter kontrol saat $\gamma = 1 \rightarrow \theta_1 = 4{,}33$, $\theta_2 = -0{,}024$. Untuk metode Lyapunov parameter kontrol saat $\gamma = 1 \rightarrow \theta_1 = 4{,}36$, $\theta_2 = -0{,}238$. Performa kontrol Lyapunov lebih cocok untuk sistem plant kontrol ketinggian *quadcopter*, hal tersebut ditunjukkan pada error semakin mendekati nilai 0 dengan waktu tempuh sekitar 150s pada metode MIT Rule dan sekitar 70s pada metode Lyapunov

6.2 Saran

Saran untuk mendukung pengembangan penelitian ini lebih lanjut yaitu:

1. Dapat menggunakan metode identifikasi sistem lainnya seperti OEEPM dan lainnya untuk membandingkan hasil pemodelan plant.
2. Sebaiknya digunakan beberapa jenis sinyal input pada plant yang diidentifikasi dan plant sesungguhnya untuk membandingkan nilai keluaranya.
3. Kontrol yang disimulasikan dapat diwujudkan dalam eksperimental dengan memasukkan algoritma kontrol dalam mikrokontroler.

DAFTAR PUSTAKA

Fatan, M., Sefidgari, B. L., & Barenji, A. V. (2013). An Adaptive Neuro PID for Controlling the Altitude of *Quadcopter* Robot Output of Plant, 662–665.

Gupte, S., & Conrad, J. M. (2012). A Survey of Quadrotor Unmanned Aerial Vehicles.

Joyo, M. K., Hazry, D., Ahmed, S. F., Tanveer, M. H., Warsi, F. A., & Hussain, A. T. (2013). Altitude and Horizontal Motion Control of Quadrotor UAV in the P resence of Air Turbulence, (December), 13–15.

Keun Uk Lee, Han Sol Kim, Jin Bae Park, Y. H. C. (2012). Hovering Control of a Quadrotor. *International Conference on Control, Automation and Systems*, (17-21, 2012), 162–167.

Liu, H., Liu, M., Wei, X., Song, Q. J., Ge, Y. J., & Wang, F. L. (2014). Auto altitude holding of quadrotor UAVs with Kalman filter based vertical velocity estimation. *Proceeding of the 11th World Congress on Intelligent Control and Automation*, 4765–4770. doi:10.1109/WCICA.2014.7053520

Mustapa, Z., Saat, S., Husin, S. H., & Abas, N. (2014). Altitude controller design for multi-copter UAV. *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, (I4ct), 382–387. doi:10.1109/I4CT.2014.6914210

Ostrowski, J. P., & Taylor, C. J. (2015). Control of a Quadrotor Helicopter Using Dual Camera Visual Feedback. doi:10.1177/0278364905053804

Pawar, R. J., Parvat, B. J., & D, P. I. (2015). Design and Implementation of MRAC and Modified MRAC technique for Inverted Pendulum, *00*(c), 1–6.

Rondon, E., Garcia-Carrillo, L.-R., & Fantoni, I. (2010). Vision-based altitude, position and speed regulation of a quadrotor rotorcraft. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 628–633. doi:10.1109/IROS.2010.5652745

Sebesta, K. D., & Boizot, N. (2014). A Real-Time Adaptive High-Gain EKF , Applied to a *Quadcopter* Inertial Navigation System, *61*(1), 495–503.

LAMPIRAN 1
PROGRAM MATLAB

Program Matlab Identifikasi

```
function[N0,nA0,nB0,d0,Tab]=estorderiv(y,u,
nmax,sel)
% Function to estimate the orders of a discrete
time system.
%
[N0,nA0,nB0,d0,Tab]=estorderiv(y,u,nmax,sel
)
%
% Inputs:
% y = output data vector
% u = input data vector
% nmax = maximum system order (default 20)
% If sel is 1, the user will be asked to select the
order N of the system
% based on the order estimation curve.
% By default, sel=0. The value of N, minimizing
the estimation criterion
% will be used.
%
% Outputs:
% N0 = global order of the system
% nA0 = order of the denominator of the
system
% nB0 = order of the numerator of the system
% d0 = identified delay
% Tab contains 3 columns [N, VS, V]
% where: N=estimated order, VS= penalized
error criterion,
% V= error criterion
%
% Written by: A. Castellanos Silva, T.B.
Airimitoaie, I.D. Landau and H. Prochazka
% Last updated 19th September 2014

if nargin<2, error('"estorderiv" needs a
minimum of 2 parameters. Write "help
estorderiv" to the console for more
informations.'); end
if nargin==2, nmax=20; sel=0; end
if nargin==3, sel=0; end

[V,S,VS]=estorderN(y,u,nmax);
[VS V];
figure
plot(0:nmax,VS,'r',0:nmax,V,'b')
title('Instrumental Variables estimation of
system order (N)')
xlabel('Order N'), ylabel('J_{IV}(N) criterion')
Tab=[(0:1:nmax)' VS V];
disp(Tab);
if sel
  N0=input('Please input here the order of the
system N = ');
else
  [vmin,imin]=min(VS);
  N0 = imin-1;
  disp(['N = ' int2str(N0)]);
end

[V,S,VS]=estdiv(y,u,N0);
[VS V];
figure
plot(0:length(VS)-1,VS,'r',0:length(V)-1,V,'b')
title('IV estimation of N-d')
xlabel('Order (N-d)'), ylabel('J_{IV}(N-d)
criterion')
[vmin2,imin2]=min(VS);
d0 = N0-imin2+1;
disp(['d = ' int2str(d0)]);

[V,S,VS]=estnBiv(y,u,N0,d0);
[VS V];
figure
plot(0:length(VS)-1,VS,'r',0:length(V)-1,V,'b')
title('IV estimation of n_B+d')
xlabel('Order (n_B+d)'), ylabel('J_{IV}(n_B+d)
criterion')
[vmin2,imin2]=min(VS);
nB0 = imin2-1-d0;
disp(['nB = ' int2str(nB0)]);

[Va,Sa,VSa]=estnAiv(y,u,N0, d0, nB0);
[VSa Va];
figure
plot(0:length(VSa)-1,VSa,'r',0:length(Va)-
1,Va,'b')
title('IV estimation of n_A')
xlabel('Order (n_A)'), ylabel('J_{IV}(n_A)
criterion')
[vmin2,imin2]=min(VSa);
nA0 = imin2-1;
disp(['nA = ' int2str(nA0)]);

function [VLs,S,VS]=estorderN(y,u,nmax)
if nargin==3,
elseif nargin==4,
else return;
end;

sz=size(y);
if(sz(2)~=1), y=y';end;%column
sz=size(u);
if(sz(2)~=1), u=u';end;

VLs=[];
S=[];
VS=[];
```

```
L=2*nmax;
N=length(y)+1-L;
%construction of Z and y0
Z=[];
for k=1:2*nmax
  Z=[Z u(2*nmax-k+1:2*nmax-k+N-1)];
end
y0=y(L+1:length(y));

%QR-triangularization
[Q,R]=qr(Z);
Z1=R(1:L,:);
Zm=Z1'\Z';
Y0=Zm*y0;
for nn=0:nmax
  %construction of R(nn)
  Rnny=[];
  Rnnu=[];
  for k=1:nn
    yp=y(L-k+1:L-k+N-1);
    up=u(L-k+1:L-k+N-1);
    Rnny=[Rnny yp];
    Rnnu=[Rnnu up];
  end;
  if nn==0
    R1nn = 0;
    %theta computation
    theta_min=0;%inv(R1nn)*Y0;
  else
    R1nn=Zm*[Rnny Rnnu];
    %theta computation

theta_min=(R1nn'*R1nn)\R1nn'*Y0;%inv(R1n
n)*Y0;
  end
  criter=sum((Y0-R1nn*theta_min).^2)/N;
  VLs=[VLs;criter];
  penal=0.4*nn*log10(N)/N;
  S=[S;penal];
end
VLs=VLs/VLs(1);
VS=VLs+S;
VS=VS/VS(1);

function [VLs,S,VS]=estdiv(y,u,nmax)
if nargin==3,
elseif nargin==4,
else return;
end;

sz=size(y);
if(sz(2)~=1), y=y';end;%column
sz=size(u);
if(sz(2)~=1), u=u';end;

VLs=[];
S=[];
VS=[];
L=2*nmax;
N=length(y)+1-L;
%construction of R(nn) and Z
Z=[];
uz=u;
for k=1:2*nmax
  Z=[Z uz(2*nmax-k+1:2*nmax-k+N-1)];
end;
y0=y(L+1:length(y));

%QR-triangularization
[Q,R]=qr(Z);
Z1=R(1:L,:);
Zm=Z1'\Z';
Y0=Zm*y0;
for nn=0:nmax
  Rnny=[];
  Rnnu=[];
  for k=1:nmax
    yp=y(L-k+1:L-k+N-1);
    Rnny=[Rnny yp];
  end
  for k=1:nn
    up=u(L+(-nmax+k-1)+1:L+(-nmax+k-1)+N-
1);
    Rnnu=[Rnnu up];
  end
  R1nn=Zm*[Rnny Rnnu];
  if size(R1nn)==[0 0], R1nn=1; end
  %theta computation

theta_min=inv(R1nn'*R1nn)*R1nn'*Y0;%inv(
R1nn)*Y0;
  criter=sum((Y0-R1nn*theta_min).^2)/N;
  VLs=[VLs;criter];
  penal=0.4*nn*log10(N)/N;
  S=[S;penal];
end
VLs=VLs/VLs(1);
VS=VLs+S;
VS=VS/VS(1);

function [VLs,S,VS]=estnBiv(y,u,nmax,d0)
if nargin~=4
  return;
end

sz=size(y);
if(sz(2)~=1), y=y';end;%column
sz=size(u);
if(sz(2)~=1), u=u';end;

VLs=[];
S=[];
VS=[];
L=2*nmax;
```

```
N=length(y)+1-L;
%construction of R(nn) and Z
Z=[];
uz=u;
for k=1:2*nmax,
  Z=[Z uz(2*nmax-k+1:2*nmax-k+N-1)];
end;
y0=y(L+1:length(y));

%QR-triangularization
[Q,R]=qr(Z);
Z1=R(1:L,:);
Zm=Z1'\Z';
Y0=Zm*y0;

for nn=0:nmax
  Rnny=[];
  Rnnu=[];
  for k=1:nmax
    yp=y(L-k+1:L-k+N-1);
    Rnny=[Rnny yp];
  end
  for k=1:nn
    up=u(L-k+1:L-k+N-1);
    Rnnu=[Rnnu up];
  end
  R1nn=Zm*[Rnny Rnnu];
  if size(R1nn)==[0 0], R1nn=1; end
  %theta computation

theta_min=inv(R1nn'*R1nn)*R1nn'*Y0;%inv(
R1nn)*Y0;
  criter=sum((Y0-R1nn*theta_min).^2)/N;
  VLs=[VLs;criter];
  penal=0.4*nn*log10(N)/N;
  S=[S;penal];
end
VLs=VLs/VLs(1);
VS=VLs+S;
VS=VS/VS(1);

function [VLs,S,VS]=estnAiv(y,u,nmax,d0,nB0)
if nargin~=5
  return;
end
sz=size(y);
if(sz(2)~=1), y=y';end;%column
sz=size(u);
if(sz(2)~=1), u=u';end;

VLs=[];
S=[];
VS=[];
L=2*nmax;
N=length(y)+1-L;
%construction of R(nn) and Z
Z=[];
for k=1:2*nmax,
  Z=[Z u(2*nmax-k+1:2*nmax-k+N-1)];
end;
y0=y(L+1:length(y));
%QR-triangularization
[Q,R]=qr(Z);
Z1=R(1:L,:);
Zm=Z1'\Z';
Y0=Zm*y0;

Rnnu=[];
for k=d0+1:d0+nB0
  up=u(L-k+1:L-k+N-1);
  Rnnu=[Rnnu up];
end

for nn=0:nmax
  Rnny=[];
  for k=1:nn
    yp=y(L-k+1:L-k+N-1);
    Rnny=[Rnny yp];
  end
  R1nn=Zm*[Rnny Rnnu];
  if size(R1nn)==[0 0], R1nn=1; end
  %theta computation

theta_min=inv(R1nn'*R1nn)*R1nn'*Y0;%inv(
R1nn)*Y0;
  criter=sum((Y0-R1nn*theta_min).^2)/N;
  VLs=[VLs;criter];
  penal=0.4*nn*log10(N)/N;
  S=[S;penal];
end
VLs=VLs/VLs(1);
VS=VLs+S;
VS=VS/VS(1);
```

Program RLS

```
function [B,A]=rls(y,u,na,nb,d,Fin,lam1,lam0)
% RLS  is used to identify a discrete time model of a plant operating in
%       open-loop based on the recursive least squares method.
%
%       [B,A]=rls(y,u,na,nb,d,Fin,lam1,lam0)
%
%       y and u are the column vectors containing respectively the output and the
%       excitation signal.
%
%       na, nb are the order of the polynomials A,B and d is the pure time delay
%
%
%                   Fin is the initial gain F0=Fin*(na+nb)*eye(na+nb)  (Fin=1000 by default)
%
%       lam1 and lam0 make different adaptation algoritms as follows:
%
%       lam1=1;lam0=1       :decreasing gain (default algorithm)
%       0.95<lam1<1;lam0=1  :decreasing gain with fixed forgetting factor
%       0.95<lam1,lam0<1    :decreasing gain with variable forgetting factor
%
%       See also FOLOE, AFOLOE, XOLOE, OLOE and RELS.
%
%
written by: A. Karimi, I.D. Landau
%                                   7th june 2002

[nl,nc]=size(y);
if nc>2, error('This routine is only for SISO systems'),end
[nl,nc]=size(u);
if nc>2, error('This routine is only for SISO systems'),end
if (na<0 | nb<0 | d<0), error('The order of A,B and d should not be negative!!'),end

nd=min(length(u),length(y));     % number of data
nth=na+nb;
if nd<nth, error('Number of data should be greater than the number of parameters!'),end
np=max(na+1,nb+d);

if nargin<5, error('This routin needs more parameters!'),end
if nargin<6, lam1=1;lam0=1;Fin=1000;end
if nargin<7, lam1=1;lam0=1;end
if nargin<8, lam0=1;end

if isempty(Fin), Fin=1000;end
if isempty(lam1),lam1=1;end
if isempty(lam0), lam0=1;end

if (lam1>1 | lam0>1), error('lam1 and lam0 should be less than 1');end
if (lam1<0.95 | lam0<0.95), disp ('warning :lam1 and lam0 are normally greater than 0.95');end


theta=zeros(nth,1);
phi=zeros(2*np,1);


F=Fin*nth*eye(nth);


i=[1:np-1 np+1:2*np-1];      %shift index for vector phi
j=[1:na np+d+1:np+d+nb];     %phi index for theta


for t=1:nd
    yhat=theta'*phi(j);
    e_apri=y(t)-yhat;
    e_apost=e_apri/(1+phi(j)'*F*phi(j));
    theta=theta+F*phi(j)*e_apost;
    F=1/lam1*(F-((F*phi(j)*phi(j)'*F)/(lam1+phi(j)'*F*phi(j))));
    lam1=lam0*lam1+1-lam0;
    phi(i+1)=phi(i);phi(1)=-y(t);phi(np+1)=u(t);
end
A=[1;theta(1:na)]';
B=[zeros(d+1,1);theta(na+1:na+nb)]';
```

Program Validasi (Whiteness Test)

```
function [wlossf,ulossf,wrni,urni,wyhat,uyhat]=olvalid(B,A,C,y,u)
%    OLVALID is used for validation of plant models identified in open loop.
%
%  [wlossf,ulossf,wrni,urni,wyhat,uyhat]=olvalid(B,A,C,y,u)
%
%   The estimation of the auto-correlation of the
%   1-step prediction error and the estimated output.
%
%    The estimation of the cross-correlation function between the
%    output error prediction and the corresponding prediction error.
%
%   The upper bound for the cross-correlation and autocorrelation estimations is given
%   for a confidence interval of %97.
%
%    The plant output y and the estimated ones (1-step predictor and output error predictor)
%   are compared and the loss functions are computed.
%
%    B and A are respectively the numerator (including the pure time delay) and denominator of the plant
%    model to be validated.
%
%  C is the numerator of the noise model. If no noise model is avalaible C must be set to 1.
%
%    y and u are the column vectors of output and input signals in
%    open loop operation.
%
%
%    wyhat   : Open loop 1-step estimated output
%    wlossf : Loss function (the sum of (y-wyhat)^2 divided by number of data)
%
%    uyhat   : Open loop output error prediction
%    ulossf : Loss function (the sum of (y-uyhat)^2 divided by number of data)
%
%   wrni   : Auto-correlations
%   urni   : Uncorrelations
%
%
written by: G. Zito, I.D. Landau
%                Last modified on November 12, 2004

na=length(A);
nB=length(B);

d=-1;i=1;
while B(i)==0,d=d+1;i=i+1;end
if d==-1, error('B should contain at least one zero!');end
if d==nB-1, error('B should be a nonzero vector!');end
nb=nB-d-1;


if isempty(C)
   nC=0;
else
   nC=length(C);
end


nd=min(length(u),length(y));    % number of data
nc=max(max(nC,max(na-1,nb)),4);

   th_lim=num2str(2.17/sqrt(nd));

   disp([' ']);

   disp('Whiteness Test')


[wyhat,err_pred]=modelsim(B,A,C,y,u,d,nd);

[wlossf,wrni]=correl(err_pred,err_pred,nc,nd)
   disp(['Theoretical upper limit : ' th_lim])
   disp([' ']);
   if (max(wrni)>0.15)
      beep;
      disp('WARNING!');
      disp('The model does not pass the whiteness test');
      disp([' ']);
   end

%%%%%%%%%%%%%%%%%%%%%%

figure;
x1=1:nc;
xx=0;yy=0;
```

```
for i=1:nc,xx=[xx x1(i) x1(i) x1(i)];yy=[yy 0
wrni(i) 0];end
xx=[xx nc+1];yy=[yy 0];
plot(xx,yy,0:nc+1,2.17/sqrt(nd)*ones(1,length
(0:nc+1)),0:nc+1,0.15*ones(1,length(0:nc+1)),
'-')
hold on
xlabel('i');
ylabel('WRN(i)');
title('Whiteness test for the OL identification')
legend('Correlation terms',strcat('Theoretical
limit: ',th_lim),'Practical limit: 0.15',0)
%%%%%%%%%%%%%%%%%%%%
  disp([' '])
  disp('Uncorrelation Test')
  disp([' '])

  sys_hat=filt(B,A,1);
  uyhat=lsim(sys_hat,u);
  err_dec=uyhat-y;
  ulossf=err_dec'*err_dec/nd
  [normr,urni]=correl(err_dec,uyhat,nc,nd);
  urni
  disp(['Theoretical upper limit : ' th_lim])
  disp([' '])
  r0=abs(err_dec'*uyhat/nd);

  % Warning if  max(URNI) > 0.15

  if (max(urni)>0.15)
     beep;
     disp('WARNING!');
     disp('The model does not pass the
uncorrelation test');
     disp([' '])
  end


%%%%%%%%%%%%%%%%%%%%
figure;
x1=1:nc;
xx=0;yy=0;
for i=1:nc,xx=[xx x1(i) x1(i) x1(i)];yy=[yy 0
urni(i) 0];end
xx=[xx nc+1];yy=[yy 0];
plot(xx,yy,0:nc+1,2.17/sqrt(nd)*ones(1,length
(0:nc+1)),0:nc+1,0.15*ones(1,length(0:nc+1)),
'-')
hold on
xlabel('i');
ylabel('URN(i)');
title('Uncorrelation test for the OL
identification')
legend('Correlation terms',strcat('Theoretical
limit: ',th_lim),'Practical limit: 0.15',0)
%%%%%%%%%%%%%%%%%%%%
% Warning if ( max(WRNI) and max(URNI) ) >
0.15

if (max(wrni)>0.15 & max(urni)>0.15)
  beep;
  disp('WARNING!');
  disp('The model has to be improved');
end

function
[y_pred,err_pred]=modelsim(B,A,C,y,u,d,nd)

na=length(A)-1;
nb=length(B)-1-d;


nc=length(C)-1;
nth=na+nb+nc;
np=max([na+1,nb+d,nc+1]);

i=[1:np-1    np+1:2*np-1    2*np+1:3*np-1];
%shift index for vector phi
j=[1:na  np+d+1:np+d+nb  2*np+1:2*np+nc];
%phi index for theta

theta=[A(2:end) B(2+d:end) C(2:end)]';
phi=zeros(3*np,1);

for t=1:nd
     y_pred(t)=theta'*phi(j);
     err_pred(t)=y(t)-y_pred(t);
     phi(i+1)=phi(i);phi(1)=-
y(t);phi(np+1)=u(t);phi(2*np+1)=err_pred(t);
end

err_pred=err_pred';

function [normr,rni] = correl(x,y,nc,nd)

switch nargin
case 4
case 3
case 2
  nc=4;
case 1
  y=x;
  nc=4;
```

```
case 0
    error('This function needs at least one
argument')
otherwise
    error('Too many arguments')
end

normr=sqrt((y'*y)*(x'*x))/nd;

for i=1:nc
    ri(i)=x(1+i:nd)'*y(1:nd-i)/(nd-i);
    rni(i)=abs(ri(i)/normr);
end
```

Program d2c pada matlab untuk mengubah sinyal diskrit to continuous

```
function [a, b] = d2c(phi, gamma, t)
%D2C  Converts discrete-time dynamic system to continuous time.
%
%   SYSC = D2C(SYSD,METHOD) computes a continuous-time model SYSC that
%   approximates the discrete-time model SYSD. The string METHOD selects
%   the conversion method among the following:
%     'zoh'     Zero-order hold on the inputs
%     'foh'     Linear interpolation of inputs
%     'tustin'  Bilinear (Tustin) approximation
%     'matched' Matched pole-zero method (for SISO systems only)
%   The default is 'zoh' when METHOD is omitted.
%
%   D2C(SYSD,OPTIONS) gives access to additional conversion options. Use
%   D2COPTIONS to create and configure the option set OPTIONS. For example,
%   you can specify a prewarping frequency for the Tustin method by:
%                                   opt = d2cOptions('Method','tustin','PrewarpFrequency',0.5);
%     sysc = d2c(sysd,opt);
%
%   For state-space models,
%     [SYSC,G] = D2C(SYSD,METHOD)
%   also returns the matrix G mapping the states xd[k] of SYSD to the states
%   xc(t) of SYSC:
%     xc(k*Ts) = G * [xd[k] ; u[k]] .
%   Given an initial condition x0 for SYSD and an initial input value u0=u[0],
%   the equivalent initial condition for SYSC is (assuming u[k]=0 for k<0):
%     xc(0) = G * [x0;u0] .
%
%   See also D2COPTIONS, C2D, D2D, DYNAMICSYSTEM.

%   J.N. Little 4-21-85
%   Copyright 1986-2011 The MathWorks, Inc.

narginchk(3,3)
[msg,phi,gamma]=abcdchk(phi,gamma);
error(msg);

[m,n] = size(phi);
[m,nb] = size(gamma);

% phi = 1 case cannot be computed through matrix logarithm.  Handle
% as a special case.
if m == 1
  if phi == 1
    a = 0; b = gamma/t;
    return
  end
end

% Remove rows in gamma that correspond to all zeros
b = zeros(m,nb);
nz = 0;
nonzero = [];
for i=1:nb
  if any(gamma(:,i)~=0)
    nonzero = [nonzero, i];
    nz = nz + 1;
  end
end

% Do rest of cases using matrix logarithm.
[s, exitflag] = logm([[phi gamma(:,nonzero)]; zeros(nz,n) eye(nz)]);
s = s/t;
if exitflag || norm(imag(s),'inf') > sqrt(eps)
  warning('Accuracy of d2c conversion may be poor.')
end
s = real(s);
a = s(1:n,1:n);
if length(b)
  b(:,nonzero) = s(1:n,n+1:n+nz);
end
```