BAB II

KAJIAN PUSTAKA

2.1 Pendahuluan

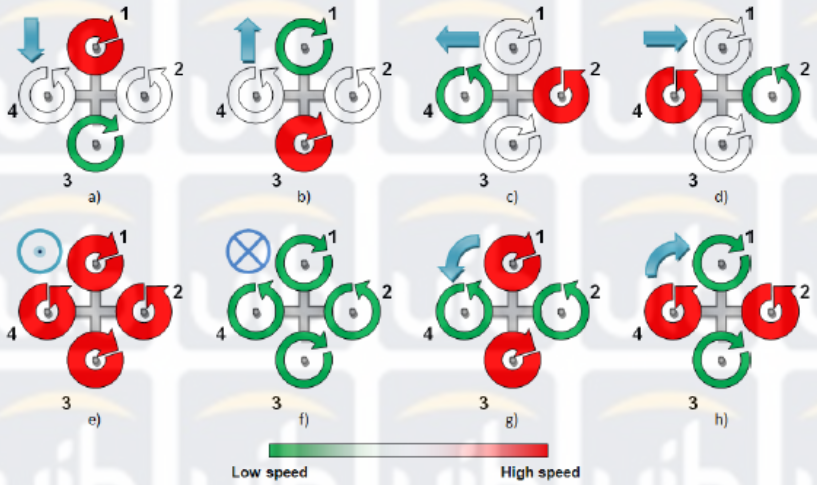
Penelitian terdahulu yang meneliti tentang pesawat UAV quadcopter dengan pembahasan kontrol *hovering* dilakukan oleh Ardy Seto Priambodo dengan judul "Perancangan dan Implementasi Sistem Kendali PID untuk Pengendalian Gerakan *Hover* pada UAV *Quadcopter*" [4]. Perancangan dan implementasi pada UAV quadcopter dilakukan menggunakan filter yaitu *complementary filter* untuk mendapatkan pembacaan sudut dari quadcopter. Hal ini dikemukakan oleh Ardy Seto Priambodo di dalam kesimpulannya, yaitu "Pembacaan sensor *accelerometer* terdapat *noise* yang nilainya cukup besar sehingga diperlukan suatu metode pengolahan data agar pembacaanya lebih baik dan tanpa *noise*. Dalam penelitian ini digunakan c*omplementary filter* dan terbukti hasil pembacaan sensor menjadi sesuai dengan sudut terukur tanpa *noise*".

Penelitian terdahulu memberikan hasil masih terdapat osilasi pada respon ketinggian yang besarnya +/-15 cm disekitar *set point* dikarenakan pembacaan sensor ketinggian yang kadang tidak sesuai dengan ketinggian terukur. Selain itu koreksi sudut *roll* & *pitch* pada kontroler *roll* & *pitch* juga masih terdapat *error* +/- 5^0. Hasil tersebut dilakukan dengan mengatur gerakan *hover* menggunakan sistem kendali PD yang memiliki konstanta *Kp* dan *Kd*. Nilai konstanta yang diberikan dari hasil tuning adalah sebesar *Kp*=40 dan *Kd*=5 untuk kontroller *roll* dan *pitch.*

Semua hasil dari penelitian terdahulu ini akan menjadi acuan dalam penelitian ini untuk melakukan pengembangan atas penelitian yang telah dilakukan tersebut, termasuk dalam sistem kontrol yang digunakan yaitu digunakan kontrol *Hybrid* PID-*Fuzzy* sebagai kontrol *hovering* yang sebelumnya hanya menggunakan kontrol PD konvensional.

2.2 Quadcopter

Quadcopter adalah robot terbang yang tersusun atas empat buah rotor dengan baling-baling yang diletakkan secara simetris pada tepi-tepi sisi quadcopter, sehingga pola peletakan dari rotor ini membentuk pola (+) atau (x) jika dilihat dari tampak atas. Pergerakan dari quadcopter ini diatur oleh pergerakan yang terjadi dari masing-masing rotor. Arah dan kecepatan rotor akan menentukan gerak, posisi dan arah terbang dari robot, termasuk perbedaan kecepatan putaran dari masing-masing rotor. Perpaduan dari keempat rotor inilah yang kemudian akan menggerakkan quadcopter dengan segala macam gerakan. Berikut beberapa gerakan dari quadcopter.



Gambar 2.1 Cara kerja *quadcopter* di ambil dari "Quadcopter Prototype"

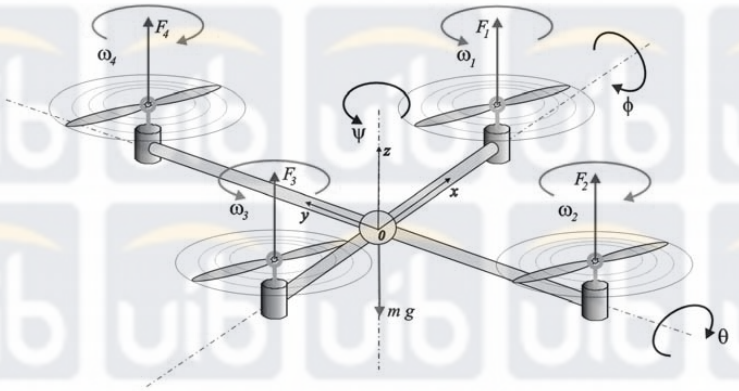
Sumber : (Domingues, 2009) [5]

Gambar 2.1 menunjukkan pergerakan quadcopter yang didapat dari pergerakan masing-masing rotor. Pada label gambar (a, b, c, d) di Gambar 2.1 menunjukkan pergerakan hanya terjadi di sumbu x dan y, hal ini dapat terlihat dari arah dan kecepatan pada dua rotor yang berseberangan. Contoh label a, dimana rotor 1 akan berputar dengan kecepatan yang lebih tinggi dibandingkan dengan rotor yang berada disebrangnya yaitu rotor 3, hal ini mengakibatkan quadcopter mendorong dirinya untuk mundur menjauhi sumbu y positif yang dengan kata lain quadcopter bergerak mundur. Kemudian label b, dimana kondisi rotor berbalik dengan kondisi rotor pada label a. Dimana rotor 3 berputar dengan kecepatan yang

tinggi atau lebih tinggi dibandingkan dengan kecepatan rotor 1 dalam berputar, hal ini akan mengakibatkan quadcopter bergerak maju menjauhi sumbu y negatif. Kondisi ini sama juga halnya dengan kondisi label c dan d, perbedaanya hanya terletak pada pergerakannya yang hanya pada sumbu x atau kanan dan kiri sedangkan label a dan b hanya pada sumbu y atau maju dan mundur.

Pergerakan tersebut sama juga halnya untuk label gambar (e, f, g, h). Gerak *hover* naik dan turun ditandai dengan gerakan yang berada di label e atau f. Pergerakan dikondisi ini adalah pergerakan yang menunjukkan pergerakan quadcopter untuk bergerak di sumbu z atau naik turun. Contoh label e, dimana semua kecepatan rotor cepat dan sama kemudian hanya berbeda untuk arah pada tiap dua rotornya, hal ini akan mengakibatkan quadcopter bergerak naik atau bergerak ke arah sumbu z positif. Sedangkan pada label f yang berputar dengan arah yang sama tetapi dengan kecepatan rotor yang kali ini rendah akan mengakibatkan quadcopter bergerak turun menjauhi sumbu z positif. Untuk label g dan h, kondisi gerak ini adalah gerak berputar dari quadcopter. Label g menunjukkan gerak quadcopter berputar berlawanan arah jarum jam dan label h yang berputar dengan gerak searah jarum jam.

Dalam pergerakannya di udara, quadcopter memiliki beberapa faktor gaya yang mempengaruhi, yaitu gaya aerodinamik, inersia, efek gravitasi dan efek giroskopik [6].



Gambar 2.2 Sistem frame dengan Sistem Sumbu B(x, y, z) dan Sumbu Bumi E(x, y, z)

Sumber : (Lukmana, M.Arifudin dan Hendro, 2011) [6]

Berikut persamaan quadcopter dalam 4 DOF [7]:

$$\ddot{z} = -g + (\cos\phi\cos\theta)\frac{1}{m}U_1 \quad (2.1)$$

$$\ddot{\phi} = \dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{j_r}{I_x}\dot{\phi}\Omega + \frac{l}{I_x}U_2 \quad (2.2)$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) - \frac{j_r}{I_y}\dot{\phi}\Omega + \frac{l}{I_y}U_3 \quad (2.3)$$

$$\ddot{\psi} = \dot{\phi}\dot{\theta}\left(\frac{I_x - I_y}{I_z}\right) + \frac{l}{I_z}U_4 \quad (2.4)$$

Dimana input kendali dari keempat DOF yaitu U1 untuk ketinggian, U2 untuk roll, U3 untuk pitch, U4 untuk yaw, dan Ω untuk kecepatan motor [7].

$$U_1 = b(\Omega_1^{\,2} + \Omega_2^{\,2} + \Omega_3^{\,2} + \Omega_4^{\,2}) \quad (2.5)$$

$$U_2 = b(\Omega_4^{\,2} + \Omega_2^{\,2}) \quad (2.6)$$

$$U_3 = b(\Omega_3^{\,2} + \Omega_1^{\,2}) \quad (2.7)$$

$$U_4 = d(\Omega_2^{\,2} + \Omega_4^{\,2} - \Omega_1^{\,2} - \Omega_3^{\,2}) \quad (2.8)$$

$$\Omega = \Omega_2 + \Omega_4 - \Omega_1 - \Omega_3 \quad (2.9)$$

Dimana:

ϕ = sudut *roll*

θ = sudut *pitch*

ψ = sudut *yaw*

Ω = kecepatan *rotor*

$I_{x,y,z}$ = inersia *body*

J_r = Inersia *rotor*

τ_a = torsi pada *airframe body*

b = faktor dorong (*trust*)

d = faktor hambatan udara (*drag*)

l = panjang lengan

2.3 PID (*Proportional Integral Derivative*)

Sistem kendali PID (*Proportional Integral Derivatif*) adalah sebuah *generic controller* yang banyak dipakai pada dunia industri. Sebuah sistem kendali PID mencoba untuk memperbaiki kesalahan antara sebuah nilai proses dan nilai *set point* yang diinginkan dengan menghitung dan melakukan pembenaran sehingga dapat meminimalkan kesalahan [8].

Sistem kendali PID terdiri dari 3 komponen utama, yaitu : *proportional*, *integral*, *derivatif*. *Proportional* menentukan nilai reaksi terhadap kesalahan saat ini. *Derivatif* menentukan nilai perubahan kesalahan yang terjadi dari kesalahan saat ini dengan kesalahan sebelumnya. *Integral* menentukan hasil penjumlahan nilai kesalahan yang terjadi [8]. Hasil nilai dari proses PID ditentukan dengan persamaan 2.10 [8]:

$$u(t) = K_p e(t) + K_I \int e(t)\,dt + K_D \frac{d}{dt} e(t) \quad \text{..... (2.10)}$$

Gambar 2.3 Sistem Kendali PID

Sumber : (Tandil, Dhanny. Sharon M. S., Ivander. Wilyanto,Yansen. Binus, 2012) [8]

Secara sederhana rumus dari persamaan dapat ditulis [8] :

u(t) = P + I + D ..... (2.11)

$P = K_P$ * *error* ..... (2.12)

$I = K_I$ * (I + *error*) ..... (2.13)

$D = K_D$ * (*error* – *error* terakhir) ..... (2.14)

error = *setpoint* – *output* ..... (2.15)

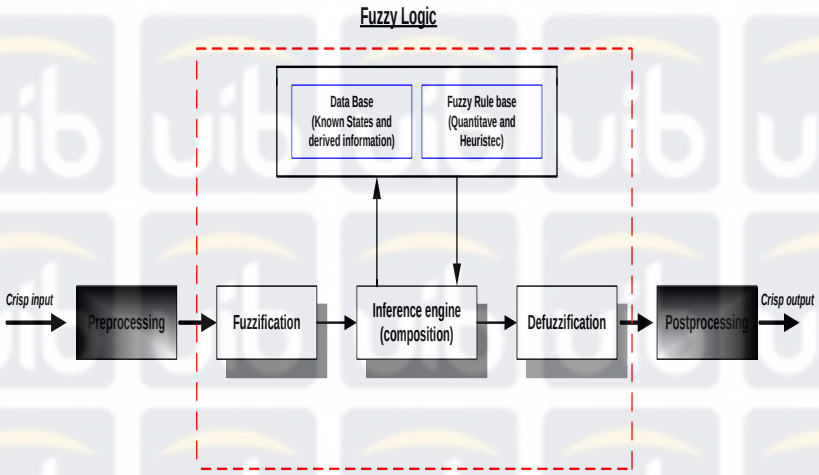
Tabel 2.1 Efek dari nilai komponen PID

Komponen	Rise Time	Overshoot	Settling Time	Error Steady State
Kp	Mengurangi	Menambah	Menimbulkan perubahan kecil	Mengurangi
Ki	Mengurangi	Menambah	Menambah	Menghilangkan
Kd	Menimbulkan perubahan kecil	Mengurangi	Mengurangi	Menimbulkan perubahan kecil

Sumber : (Jepry, FT UI, 2010) [9]

2.4 *Fuzzy Logic* Kontrol

Sistem kendali *fuzzy logic* pertama kali diperkenalkan kepada publik sekitar tahun 1965 oleh Prof.Lotfi Zadeh dari Universitas California (Barkley). Sistem ini memiliki kelebihan dimana pemodelan dari suatu sistem yang kompleks dapat dibuat dengan sedikit data numerik. Hal ini sulit dilakukan oleh pemodelan sistem yang lain [10]. Kendali logika *fuzzy* terdiri dari tiga komponen utama yaitu *fuzzification, inference engine,* dan *defuzzification* seperti pada gambar di bawah ini [11]:



Gambar 2.4 Sistem Kendali *Fuzzy Logic*

Sumber : (Zulhamdi. 2006) [11]

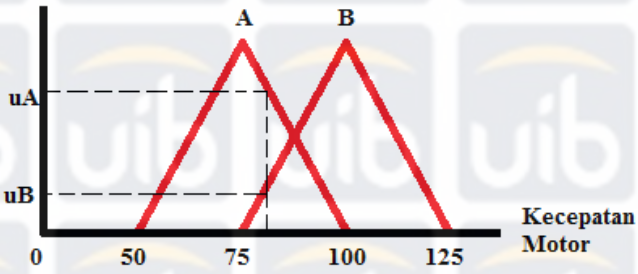
Secara fungsional blok-blok dari sistem kendali logika *fuzzy* di atas dapat dijelaskan sebagai berikut [10] :

2.4.1 *Preprocessing*

Bagian ini merupakan bagian awal dari sistem kendali *fuzzy*. Parameter yang akan dikendalikan masih merupakan nilai *crisp* yang harus dinormalisasi. Bentuk fisis blok ini berupa peralatan pengkondisi sinyal yang terdiri dari sensor dan ADC.

2.4.2 *Fuzzification*

Masing-masing parameter input yang telah ternormalisasi, oleh *fuzzification* diubah ke dalam bentuk bahasa (linguistic) yang bersesuaian dan kemudian dinyatakan pada fungsi keanggotaan yang sesuai. Dari fungsi keanggotaan ini bisa diketahui berapa derajat keanggotaan atau μ_A (x) dari masing-masing input x. Keseluruhan interval / fungsi keanggotaan parameter ini membentuk semesta himpunan *fuzzy* (*universe of discourse*).



Gambar 2.5 Fungsi Fuzzifikasi

Sumber : (*Handout* Mata Kuliah *Artificial Intelligence*, 2014)

Pada gambar di atas, contoh perhitungan fuzzifikasi dapat ditunjukkan sebagai berikut:

$$\mu A = \frac{c-x}{c-b} = \frac{100-80}{100-75} = 0{,}8 \qquad \mu B = \frac{x-b}{c-b} = \frac{80-75}{100-75} = 0{,}2$$

2.4.3 *Inference Engine* atau *Rule Evaluation*

Inference Engine memetakan *fuzzy* input menjadi *fuzzy output* atau membuat aturan-aturan yang akan terjadi yang dialami oleh *fuzzy input*. Pemetaan ini didasarkan pada sejumlah *fuzzy If – Then rules*. *If –Then rules* terdiri dari dua bagian utama, yaitu :

- *The antecedent part (premise*) yaitu kata di antara *if* dan *then* yang merupakan *input fuzzy*.
- *The consequent part (conclusion)* yaitu kata setelah *then* yang merupakan *output fuzzy*.

Dengan demikian *fuzzy if then rules* merupakan penghubung antara *antecedent* (input *fuzzy*) dengan *consequent* (*output fuzzy*) yang bersesuaian. Sehingga *If then rules* dapat dituliskan dengan :

IF.. (*antecedent*)

THEN.. (*consequent*)

Pada sistem kendali yang dirancang, sistem *fuzzy* memiliki lebih dari satu input yaitu *error* dan *delta error* yang dihasilkan dari sistem kendali. Maka dalam membuat *fuzzy if – then rules*, pada bagian *antecedent* dapat digunakan operator *And* atau *Or*.

Contoh :

IF Distance is Short And Speed is Low THEN Output is Moderate

Kumpulan dari sejumlah *if – then rules* dapat dibuat ke dalam suatu tabel yang menghubungkan *fuzzy input* dan *fuzzy output* yang disebut *Fuzzy Associative Memories* (FAM's).

2.4.4 *Defuzzification*

Suatu sistem *fuzzy* akan mengeluarkan *output* berupa himpunan *fuzzy* (*fuzzy output*). Untuk dapat berinteraksi dengan sistem di luar, maka perlu mengubahnya menjadi suatu data *crisp* (tegas). Dengan proses defuzzifikasi maka himpunan *fuzzy* hasil inferensi akan diubah menjadi suatu data *crisp* tunggal. Data *crisp* tunggal inilah yang merupakan sinyal kontrol. Dalam melakukan defuzzifikasi terdapat beberapa metode, antara lain [12] :

1. Metode *Max* (*Maximum*)

Metode ini juga dikenal dengan metode puncak dimana nilai keluaran dibatasi oleh fungsi :

$$\mu c(z)^* > \mu c\ 1\ (z) \quad \text{.....(2.16)}$$

2. Metode Titik Tengah (*Center of Area*)

Metode ini juga disebut pusat area. Metode ini lazim dipakai dalam proses defuzzifikasi. Metode ini diekspresikan dengan persamaan :

$$z^* = \frac{\int \mu c(z) z dz}{\int \mu c(z) z dz} \quad \text{.....(2.17)}$$

3. Metode Rata-rata (*Average*)

Metode ini digunakan untuk fungsi keanggotaan keluaran simetris. Metode ini diekspresikan dengan persamaan :

$$z^* = \frac{\sum(z).Z}{\mu c(z)} \quad \text{.....(2.18)}$$

4. Metode Penjumlah Titik Tengah (*Summing of Center Area*)

Metode ini diekspresikan dengan persamaan :

$$z^* = \frac{\int \sum = 1 \mu ck(z) dz}{\int \sum = 1 \mu cm(z) dz} \quad \text{.....(2.19)}$$

5. Metode Titik Tengah Area Terbesar

Dalam metode ini keluaran dipilih berdasarkan titik pusat area terbesar yang ada. Metode ini diekspresikan dengan persamaan :

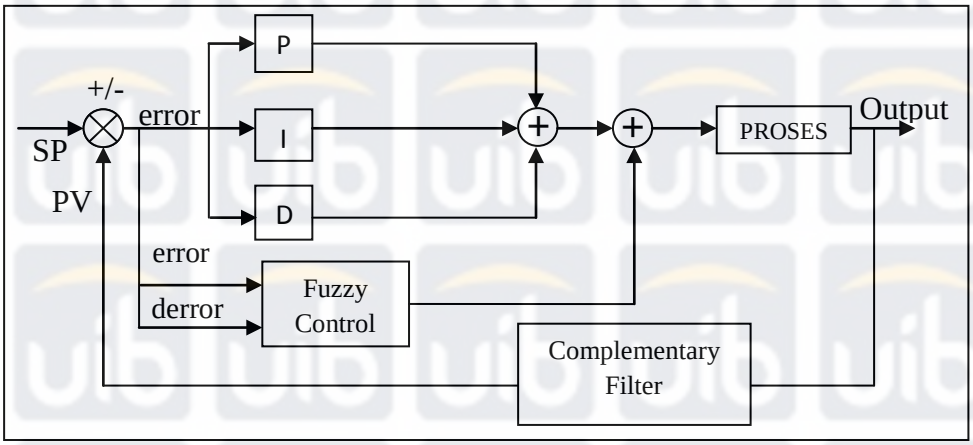
$$z^* = \frac{\int \mu c(z) z dz}{\int \mu c(z) dz} \quad \text{.....(2.20)}$$

2.4.5 *Postprocessing*

Output analog (*crisp ouput*) hasil defuzzifikasi belumlah nilai aktual untuk input pada sistem yang akan dikontrol, oleh sebab itu nilai ini perlu didenormalisasi terlebih dahulu supaya jangkauan semestanya sesuai untuk input kontroler. Dalam hal ini *crisp* output pada semesta *fuzzy* harus disesuaikan (*scalling*) dengan satuan teknik (*engineering units*) seperti volt, meter atau jam.

2.5 *Hybrid* PID-*Fuzzy* Kontrol

Hybrid PID-*Fuzzy* adalah kontrol yang mengkombinasikan antara kontrol PID konvensional dengan kontrol *fuzzy logic*. Pemakaian kombinasi dari kedua kontrol ini diharapkan dapat memberikan keunggulan-keunggulan dibandingkan dengan metode pengontrolan konvensional masing-masing kontrol.



Gambar 2.6 Diagram Blok Hybrid Kendali Fuzzy-PID

Sumber : (H. Samsul Bachri M., 2014) [13]

Blok diagram diatas adalah blok diagram yang akan diimplementasikan kendalinya ke dalam kontrol *hovering* yang menjadi tujuan dari penelitian ini. Dari uraian blok diagram diatas dapat diketahui bahwa terdapat dua kendali yaitu kendali PID konvensional dan kendali *fuzzy logic*. Input dari PID adalah nilai *error* dari sistem kendali sedangkan untuk *fuzzy logic* adalah nilai *error* dan nilai *delta error* yang dihasilkan dari kendali pula. Nilai *error* pada blok diagram ini didapat dari perbandingan antara nilai kecepatan dan arah motor dengan pembacaan yang didapat dari sensor IMU.

Masing-masing kontrol akan memproses nilainya masing-masing sesuai dengan kontrolnya hingga pada akhirnya masing-masing kontrol akan menghasilkan sebuah nilai tunggal yang dimana kontrol PID akan menghasilkan nilai PWM yang siap untuk diberikan kepada motor dan kontrol *fuzzy logic* yang menghasilkan nilai *crisp* tunggal yang sudah diadaptasi semesta pembicaraannya menjadi nilai yang sesuai yang dapat diterima oleh motor. Hasil dari dua kontrol

ini akan di-*hybrid*-kan sesuai dengan blok diagram pada Gambar 2.7 dan siap diberikan pada motor.

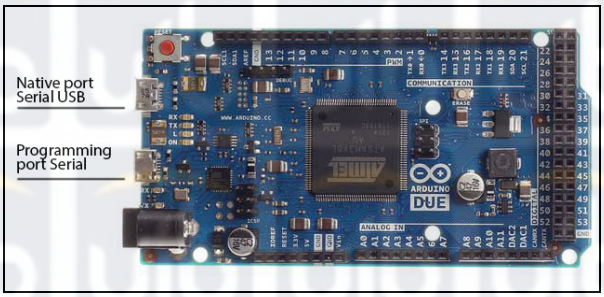
$$Hybrid = PID + FuzzyLogic \quad \text{..... (2.21)}$$

2.6 Mikrokontroller Arduino Due

Arduino Due adalah papan mikrokontroller yang didasarkan dari chip Atmel SAM3X8E ARM Cortex-M3 CPU dan beroperasi pada inti 32-bit. Memiliki 54 digital I/O dan 12 analog input. Mikrokontroller ini beroperasi pada tegangan 3.3V yang berbeda dari mikrokontroller lainnya yaitu 5V, mengartikan bahwa pin I/O hanya bisa mendapatkan input atau mengeluarkan output tegangan maksimum sebesar 3.3V dan jika input lebih dari 3.3V maka akan dapat merusak mikrokontroller arduino due ini. Sumber tegangan untuk mikrokontroller ini memiki range sebesar 6-16V.

Arduino Due memiliki dua port USB. Port USB Native (Port untuk komunikasi serial CDC menggunakan objek Serial USB) ini terhubung langsung ke SAM3X MCU. Sedangkan Port USB yang lain adalah port untuk pemrograman. Port ini terhubung ke ATMEL 16U2 yang bertindak sebagai USB-to-Serial konverter. Port untuk pemrograman ini adalah port standar untuk meng-upload program dengan Arduino. USB-to-serial konverter dari port pemrograman terhubung dengan UART pertama dari SAM3X. Ini memungkinkan untuk berkomunikasi melalui port ini menggunakan objek Serial dalam bahasa pemrograman Arduino.

Konektor USB dari port Native langsung terhubung ke pin USB host dari SAM3X. Port Native ini dapat digunakan sebagai sebagai perangkat host USB yang mana perangkat lain (seperti mouse, keyboard, atau ponsel Android) dapat terhubung ke Due. Port ini juga dapat digunakan sebagai port serial virtual menggunakan objek SerialUSB dalam bahasa pemrograman Arduino.



Gambar 2.7 Arduino Due

Sumber : (Arduino.cc,2014) [14]

Tabel 2.2 Arduino Due *Specification*

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Outputs Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz

Sumber : (Arduino.cc, 2014) [14]

2.6.1 *Input and Output*

- Digital I/O : Memiliki 54 digital pin. Setiap pin dapat digunakan sebagai input atau output dengan menggunakan fungsi *pinMode( ), digitalWrite( ),* dan *digitalRead( )* yang beroperasi pada tegangan 3,3 volt. Setiap pin juga dapat menyediakan sumber arus 3 mA atau 15 mA, dan menerima (*sink*) arus 6 mA atau 9 mA. Memiliki resistor pull-up internal 100 KOhm. Selain itu, beberapa pin memiliki fungsi tertentu.
- Serial 0 : 0 (RX) dan 1 (TX)
- Serial 1 : 19 (RX) dan 18 (TX)
- Serial 2 : 17 (RX) dan 16 (TX)
- Serial 3 : 15 (RX) dan 14 (TX)
- PWM : Pin 2 - 13, menyediakan output PWM 8-bit dengan menggunakan fungsi *analogWrite( ).* Resolusi dari PWM dapat diganti dengan menggunakan fungsi *analogWriteResolution( ).*
- Analog Input : Pin dari A0 - A11. Arduino Due memiliki 12 input analog.

2.7 Sensor IMU MPU6050 GY-86

Sensor yang digunakan pada robot *quadcopter* ini terdiri dari sensor *accelerometer, gyroscope.* Kedua sensor ini dikombinasikan oleh suatu modul hardware yang dinamakan IMU (*Inertial Measurement Unit*). IMU yang digunakan pada penelitian ini adalah IMU MPU6050 GY-86. Gambaran dari sensor IMU dapat dilihat pada Gambar 2.8.

Gambar 2.8 Modul IMU (*Inertial Measurement Unit*) 10 DOF

Sumber : (http://arduino.altervista.org/blog/wp-content/uploads/2014/07/ljp_2613.jpg, 2014)

Berikut karakteristik dari prinsip kerja dari sensor *gyroscope* dan *accelerometer* yang dikombinasikan pada modul IMU :

2.7.1 *Gyroscope*

Merupakan sensor yang digunakan untuk mengukur kecepatan sudut dan menghasilkan nilai output yang berubah seiring waktu. Secara umum hasil pengukuran kecepatan sudut sebuah benda dengan menggunakan sensor *gyroscope* pada bidang horisontal dapat dinyatakan dengan persamaan:

$$\dot{\theta}_T(t) = \dot{\theta}(t) + n(t) + b(t) \quad \text{(2.22)}$$

Sinyal keluaran *gyroscope* secara umum mengandung sinyal kecepatan sudut (q&(t)), *random noise* (*n(t)*), dan *noise* karena perubahan temperatur (*b(t)*). Perubahan besaran sudut diperoleh dengan mengintegralkan persamaan 2.22. Persamaan perubahan besaran sudut ditulis menjadi persamaan 2.23:

$$\theta_T(t) = \int \left(\dot{\theta}(t) + n(t) + b(t)\right) dt \quad \text{(2.23)}$$

Persamaan 2.23 dapat ditulis kembali dengan sebuah parameter kalibrasi secara sederhana menjadi:

$$\theta_T(t) = K \int \left(\dot{\theta}(t)\right) dt \quad \text{(2.24)}$$

2.7.2 *Accelerometer*

Merupakan sensor yang digunakan untuk mengukur kecepatan linear dan sensitif terhadap getaran dan *noise*. Sudut dari *accelerometer* didapat dari perhitungan sederhana trigonometri, berdasar pada rumus,

$$F = m \cdot a \quad \text{(2.25)}$$

Maka sudut terhadap sumbu x dapat didapat dengan mencari resultan gaya dari gaya pada sumbu y dan sumbu z, lalu mencari besar sudut antara resultan tersebut dengan gaya pada sumbu x. Besar resultan gaya F_{yz} dapat dicari dengan menggunakan teorema *Pythagoras*.

$$F_{yz} = \sqrt{F_y{}^2 + F_z{}^2} \quad \text{(2.26)}$$

Sedangkan untuk mencari sudut terhadap sumbu x maka resultan tersebut kemudian dihitung dengan *arctan*.

$$\theta_x = \arctan\left(\frac{F_x}{F_{yz}}\right) \quad \text{..... (2.27)}$$

Hal yang sama dilakukan pada sumbu y, mencari resultan F_z dan F_x lalu dengan *arctan* dicari sudut antar dua gaya tersebut. Sedangkan nilai massa dari alat dapat diabaikan karena persamaan 2.25 dapat diubah menjadi

$$\theta_x = \arctan\left(\frac{m \cdot a_x}{m \cdot a_{yz}}\right) \quad \text{..... (2.28)}$$

$$\theta_x = \arctan\left(\frac{a_x}{a_{yz}}\right) \quad \text{..... (2.29)}$$

2.8 Filter

2.8.1 Low Pass Filter

Adalah filter yang hanya melewatkan frekuensi yang lebih rendah dari frekuensi cut-off (f_c). Diatas frekuensi tersebut, outputnya mengecil atau dianggap tidak ada (idealnya) [15]. Rangkaian RC LPF dan tanggapan frekuensinya ditunjukkan oleh Gambar 2.9 berikut :

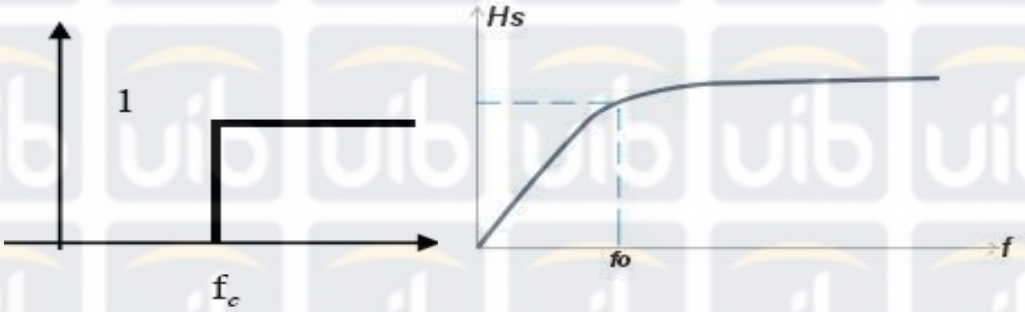
Gambar 2.9 *Low pass filter* (*ideal signal* dan *realistic signal* LPF)

Sumber : (Kusuma W., Sastra, 2014) [15]

2.8.2 High Pass Filter

Adalah filter yang outputnya hanya melewatkan frekuensi diatas frekuensi cut-off. Di bawah frekuensi itu, output mengecil atau idealnya tidak ada [15].

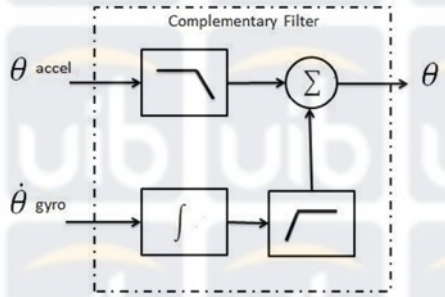
Rangkaian RC HPF dan tanggapan frekuensinya ditunjukkan pada Gambar 2.10 berikut :



Gambar 2.10 *High pass filter* (*ideal signal* dan *realistic signal* HPF)

Sumber : (Kusuma W., Sastra, 2014) [15]

2.8.3 *Complementary Filter*



Gambar 2.11 *Complementary filter*

Sumber : (Talok, Anton. Reigton H., John. 2013) [16]

Complementary filter adalah filter yang menggabungkan 2 jenis input yang berbeda yang berfungsi untuk mengisi kekurangan satu sama lain. Input dari filter ini merupakan dua buah input sinyal yang berbeda dengan karakteristik input satu yang memiliki noise tinggi dan yang lain memiliki noise rendah. Filter ini digunakan untuk menggabungkan data *accelerometer* dan *gyroscope*, dimana *accelerometer* merupakan sensor yang memiliki nilai noise yang sangat besar dan sangat bervariasi sedangkan *gyroscope* merupakan sensor yang memiliki karakter untuk mengalami sedikit pergeseran terus menerus. Nilai dari *accelerometer* yang berupa kecepatan linear di setiap sumbu diubah menjadi sudut dengan

trigonometri, dan nilai kecepatan sudut dari *gyroscope* diubah menjadi sudut dengan meng-integral-kan nilainya. Kemudian nilai dari *accelerometer* yang memiliki banyak noise di-*filter* dengan menggunakan *low-pass filter* agar perubahan mendadak yang dimilikinya (noise) dapat dihilangkan, dan nilai dari *gyroscope* yang memiliki sedikit perubahan, di *high-pass filter* agar perubahan yang sedikit dan terus menerus itu dapat dihilangkan, lalu nilai dari kedua filter ini digabungkan untuk membentuk sebuah nilai sudut.

Proses ini dilakukan dengan menggunakan sebuah persamaan berikut :

$$angle = (a) * (angle + gyro * dt) + (1 - a) * (\theta acc) \quad \text{..... (2.30)}$$

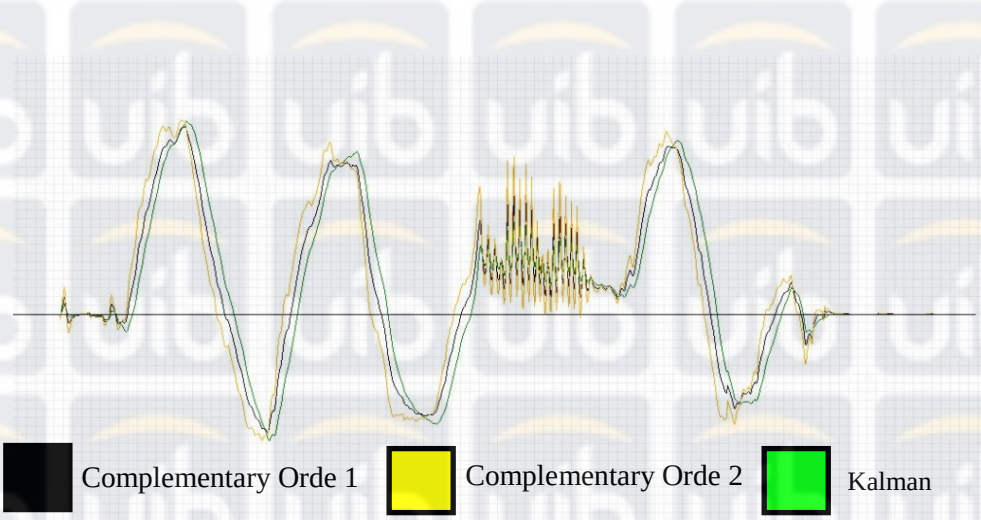
Pada persamaan ini, nilai yang didapat dari *gyroscope* di-integral agar mendapatkan nilai sudut, dan nilai sudut dari *accelerometer* didapat dengan perhitungan trigonometri. Nilai *a* merupakan nilai yang didapat dari *time constant* atau kerangka waktu dimana filter ini bekerja, dan nilai ini akan mempengaruhi nilai sudut dari *gyroscope* dan *accelerometer* sebagai *high-pass* dan *low-pass filter*. Perhitungan *time constant*(τ) didapat dari,

$$\tau = \frac{a \cdot dt}{1-a} \quad \text{..... (2.31)}$$

$$a = \frac{\tau}{\tau + dt} \quad \text{..... (2.32)}$$

Nilai *a* akan menghasilkan nilai di bawah 1 yang membuat nilai sudut merupakan gabungan dari kedua nilai sensor yang *noise*-nya telah dikurangi dengan filter.

Hasil dari *complemetary filter* orde satu dan orde dua pernah dilakukan dengan membandingkan hasilnya dengan *kalman filter* yang dikenal cukup baik. Dan filter ini diimplementasikan dengan mikrokontroller Arduino (17). Dan hasilnya dapat dilihat pada Gambar 2.12.



Gambar 2.12 Perbandingan *Kalman filter* dan *Complementary filter* orde 1 dan 2

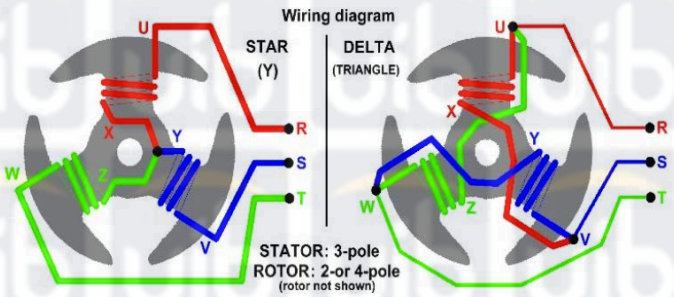
Sumber : (Alegiaco, 2011) [17]

2.9 Motor *Brushless*

Motor *brushless* atau motor DC tanpa sikat merupakan perangkat elektromagnetis yang mengubah energi listrik menjadi energi mekanik. Motor ini menggunakan bahan semikonduktor untuk merubah maupun membalik arah putarannya, serta tingkat kebisingan motor jenis ini rendah karena putarannya yang halus [18].

Motor *brushless* ini merupakan motor listrik *synchronous* AC 3 fasa, tetapi tetap disebut dengan motor DC karena pada implementasinya motor ini menggunakan sumber tegangan DC sebagai sumber energi utama yang kemudian diubah menjadi tegangan AC dengan menggunakan inverter 3 fasa. Inverter yang digunakan pada penelitian ini adalah inverter model ESC (*electronic speed control*). Motor jenis ini memiliki biaya perawatan yang lebih rendah dan kecepatan yang lebih tinggi akibat tidak digunakannya *brush* [19].

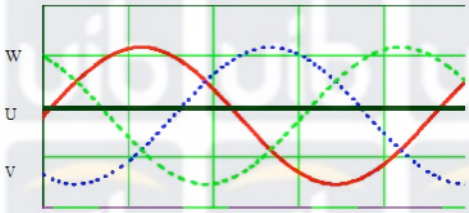
Motor *brushless* ini terdiri dari dua bagian,yakni rotor dan stator. Rotor merupakan bagian yang bergerak yang terbuat dari magnet permanen dan stator yang merupakan bagian yang diam. Stator terbuat dari kumparan 3 fasa [19].



Gambar 2.13 Wiring diagram BLDC

Sumber : (http://circuitelec.blogspot.com/2009/07/brushless-dc-motors-theory-and-driver.html**,** 2014)

Berdasarkan Gambar 2.13, medan putar magnet stator timbul akibat adanya perubahan polaritas pada stator U, V, dan W. Perubahan polaritas ini terjadi akibat adanya arus yang mengalir pada stator.



Gambar 2.14 Tegangan stator BLDC Motor

Sumber : (http://lontar.ui.ac.id/file?file=digital/20248993-R030970.pdf, 2014)

Berdasarkan Gambar 2.14, ketika stator U diberikan tegangan negative maka akan timbul medan magnet dengan polaritas negative sedangkan V dan W yang diberikan tegangan positif akan memiliki polaritas positif. Akibat adanya perbedaan polaritas antara medan magnet kumparan stator dan magnet rotor, sisi positif magnet rotor akan berputar mendekati medan magnet stator U, sedangkan sisi negatifnya akan berputar mengikuti medan magnet stator V dan W. Akibat tegangan yang digunakan berupa tegangan AC sinusoidal, medan magnet stator U, V, dan W akan berubah – ubah polaritasnya dan besarnya mengikuti perubahan tegangan sinusoidal AC. Ketika U dan V memiliki medan magnet negative akibat

mendapatkan tegangan negative dan W memiliki medan magnet positif akibat tegangan positif, magnet permanen rotor akan berputar menuju ke polaritas yang bersesuaian yakni bagian negative akan berputar menuju medan magnet stator W dan sebaliknya bagian positif akan berputar menuju medan magnet stator U dan V. Selanjutnya ketika V memiliki medan magnet negative dan U serta W memiliki medan magnet positif, bagian positif magnet permanen akan berputar menuju V dan bagian negative akan menuju U dari kumparan W. Karena tegangan AC sinusoidal yang digunakan berlangsung secara kontinu, proses perubahan polaritas tegangan pada stator ini akan terjadi secara terus menerus sehingga menciptakan medan putar magnet stator dan magnet permanen rotor akan berputar mengikuti medan putar magnet stator ini. Hal inilah yang menyebabkan rotor pada motor *brushless* dapat berputar [20].

2.10 *Electronic Speed Control*

Sebuah modul rangkaian electronik yang berfungsi untuk mengatur putaran motor dengan mengatur suplai arus yang disesuaikan dengan kebutuhan motor *brushless* [21]. Selain mengatur kecepatan motor, ESC juga berfungsi untuk menaikkan jumlah arus yang diperlukan oleh motor brushless. Pemakaian ESC pada motor brushless ini merupakan pertimbangan wajib untuk digunakan, dikarenakan motor brushless yang bergerak jika adanya sinyal sinusoidal 3 fasa dan ESC digukan sebagai inverter untuk memberikan sinyal sinusoidal 3 fasa pada motor brushless. Gambar dari ESC yang digunakan dapat dilihat pada Gambar 2.15.

Gambar 2.15 *Electronic Speed Control*

Sumber : (http://www.house4hack.co.za/wordpress/wp-content/uploads/2013/10/dji-30a-opto-esc-245-p.jpg, 2014)

2.11 *Remote Control*

Remote control merupakan bagian yang berinteraksi langsung dengan pengguna untuk memberikan sinyal perintah-perintah untuk menggerakkan robot dalam arah gerakan arah naik, turun, maju, mundur, kiri dan kanan. *Remote control* pada penelitian ini digunakan untuk menggerakkan robot quadcopter ke arah yang diinginkan. *Remote control* juga digunakan untuk membantu penulis untuk mengerakkan quadcopter pada saat pengujian atau pengambilan data.

Gambar 2.16 *Remote Control*

Sumber : (http://myrcflight.com/uploads/3/5/6/3/3563609/9664994_orig.jpg?203, 2014)