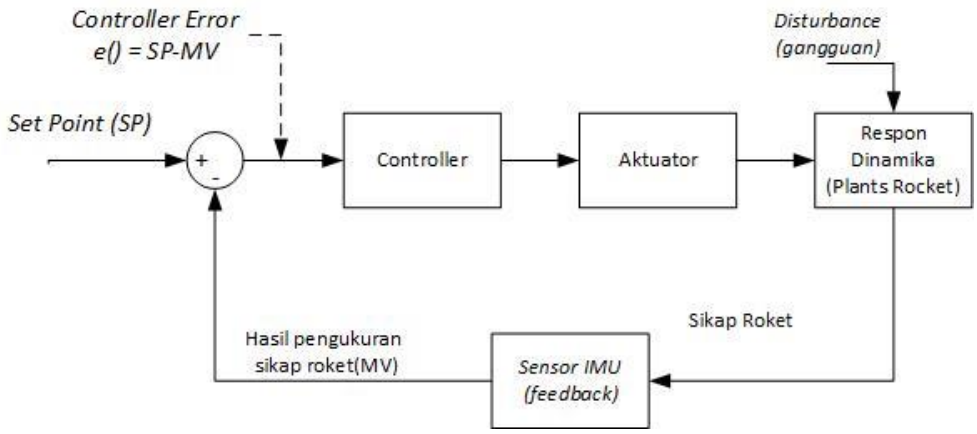
BAB III

PERANCANGAN SISTEM

Bab ini akan menjelaskan bagaimana perancangan *firmware* untuk kestabilan roket *Electric Ducted Fan* (EDF) pada saat *cruising*. Sebelum perancangan *firmware* ada beberapa hal yang dibutuhkan sebagai pendukung yaitu roket EDF, *Ground Control Station* (GCS) dan radio sebagai komunikasi. Semua pendukung itu sudah dibuat di lab divisi roket unikom. Sebelum masuk ke perancangan *firmware*, penulis akan memaparkan sistem kontrol roket, dan komponen elektronika yang dibutuhkan untuk perancangan roket. Berikut komponen yang dipakai roket.

3.1 Kontrol Roket

Pada kontrol roket memerlukan sistem pengendalian yang baik, untuk melakukan pengaturan besar defleksi yang diperlukan untuk menghasilkan momen aerodinamis sehingga dapat mengurangi gangguan internal yang berasal dari putaran motor pendorong. Hal tersebut menyebabkan perlu adanya proses timbal balik terhadap besar gangguan yang terjadi dalam pendesainan kontrol sistem. Oleh karena itu, kontrol *closed-loop* dipilih sebagai jenis sistem kontrol yang digunakan karena adanya *feedback* atau umpan balik yang mempengaruhi proses kontrol. Pendesainan sistem dari sudut pandang kontrol memiliki diagram blok seperti gambar 3.1 berikut ini.



Gambar 3.1 Diagram blok kontrol *closed-loop* sikap roket.

Berikut ini penjelasan diagram blok kontrol *closed-loop* sikap roket:

- *Controller* : proses kontrol dilakukan pada kontrol unit yang berupa mikrokontroler. Algoritma kontrol akan dimasukkan pada mikrokontroler yang akan mengontrol aktuator. Selain untuk melakukan kontrol pada aktuator, mikrokontroler akan melakukan tugas pembacaan dan pengolahan sensor yang akan sangat mempengaruhi kinerja pada kontrol sistem. [2]
- Aktuator : aktuator merupakan suatu mekanisme yang akan berpengaruh pada sikap roket, oleh karena itu untuk mengatur sudut dari fin pengontrol roket diperlukan motor servo. [2]
- *Plants Rocket* : *plants* pada sistem merupakan badan roket yang memiliki ukuran tertentu yang telah dirancang. Untuk perancangan dan ukuran roket tidak akan dibahas terlalu dalam sesuai dengan yang tercantum pada batasan masalah. [2]
- *Feedback* : *feedback* merupakan sebuah nilai umpan balik dari adanya sistem yang terjadi. Karena factor perubahan posisi yang diperlukan maka pada bagian *feedback* dapat disebut juga dengan *feedbackpositioning.* Untuk

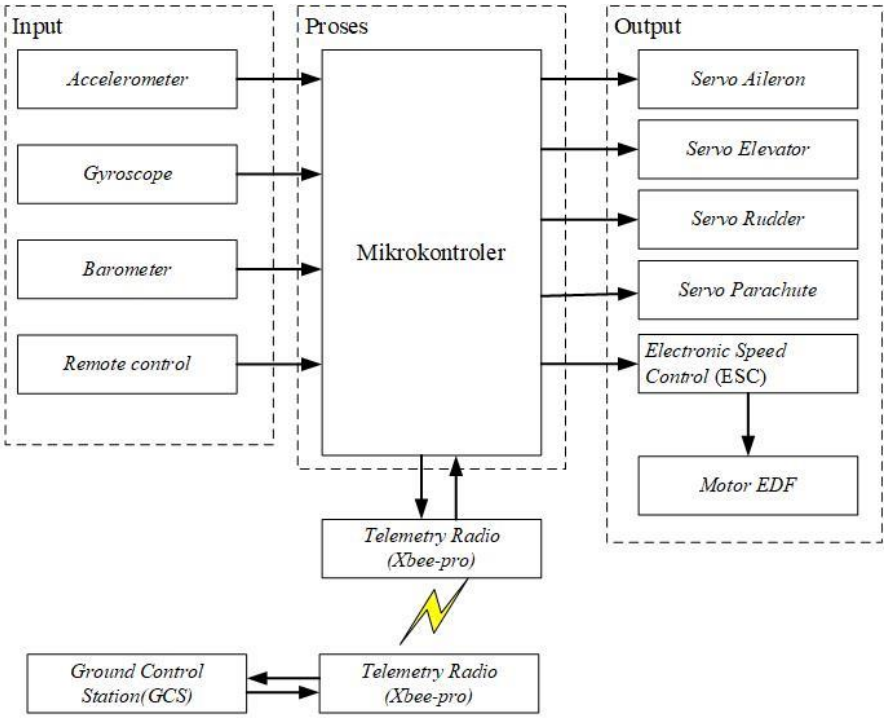
mendapatkan nilai perubahan posisi yang baik maka pada pengolahan sensor harus dilakukan dengan baik sehingga mendapatkan nilai perubahan yang akurat dan baik. [2]

- *Disturbances* : *disturbances* atau sebuah gangguan yang dihasilkan oleh perubahan kecepatan dan arah angin. Perubahan kecepatan dan arah angin ini akan menimbulkan gangguan pada kestabilan sikap roket pada saat melakukan *cruising.*[2]

3.2 Perancangan Perangkat Keras

Sebelum masuk ke perancangan *firmware,* maka ada baiknya memahami terlebih dahulu blok diagram dari roket EDF. Blok diagram ini menunjukan logika roket EDF secara sekuensial, proses kerja dari roket EDF yang berjalan melalui sinkronisasi. Maksudnya yaitu rangkaian logika roket dapat terbang mulai dari masukan dan di proses oleh mikrokontroler yang akan mempengaruhi keluaran aktuator.

Blok diagram roket merupakan gambaran bagaimana proses roket untuk terbang secara logika. Masukan sensor yang merupakan data sebagai acuan roket untuk terbang dan data tersebut akan mempengaruhi keadaan roket ketika terbang karena sensor akan membaca keadaan roket yang kemudian akan diproses sehingga dapat memberikan perintah terhadap aktuator untuk mengontrol roket EDF. Berikut blok diagram roket EDF pada Gambar 3.2.



Gambar 3.2 Blok Diagram roket EDF

Pada gambar 3.2 terdapat blok input, proses dan output. Blok input dari roket EDF adalah sensor *accelerometer*, *gyroscope*, *barometer* dan RC (*Remote Control*). Sensor *accelerometer*, *gyroscope* merupakan sensor orientasi untuk kestabilan roket EDF. Dari sensor tersebut dijadikan data 3 axis pada roket. 3 axis tersebut mewakili *roll*, *pitch* dan *yaw* pada roket. Berikut ini penjelasan dari blok input, proses dan output:

1. Bagian-bagian pada input blok diagram

   a) Sensor *Accelerometer*

   Sensor *Accelerometer* adalah sensor orientasi yang berfungsi untuk mendeteksi, mengukur percepatan dan getaran. *Accelerometer* digunakan untuk mengukur akselerasi yang terjadi pada suatu benda, terdapat banyak metode pengukuran pada accelerometer salah satunya adalah berjenis pegas,

dengan memanfaatkan perubahan panjang pegas yang terjadi yang kemudian nilai pembacaan tersebut dikonversikan menjadi nilai akselerasi yang terjadi, sensor accelerometer juga dapat difungsikan untuk mengukur kemiringan suatu benda dengan memanfaatkan gaya gravitasi bumi.

Accelerometer akan bekerja jika mendapat getaran, contoh pada roket ketika sedang diam lalu mendapat kecepatan dari luar, *accelerometer* akan merespon getaran yang terkait dengan kecepatan tersebut. *Accelerometer* menggunakan kristal mikroskopis yang akan mengalami perubahan tegangan ketika getaran terjadi dan dari tegangan itu dihasilkan tegangan untuk menciptakan pembacaan.

b) Sensor *Gyroscope*

Gyroscope adalah sensor orientasi yang menggunakan gravitasi bumi sebagai acuan dalam menentukan orientasi. Cara kerja gyroscope seperti piringan yang berputar bebas yang disebut rotor, dipasang pada poros yang berputar di tengah roda yang lebih besar dan lebih stabil. Ketika sumbu berputar, rotor tetap diam untuk menunjukkan tarikan gravitasi pusat. Perbedaan *gyroscope* dengan *accelerometer* yaitu *gyroscope* dapat merasakan rotasi bumi sedangkan *accelerometer* tidak.

c) Sensor *Barometer*

Barometer merupakan sensor ketinggian yang mengacu pada permukaan laut sebagai titik nol dari ketinggian. Pada *flightcontroller* pixhawk, *barometer* dapat dapat dikalibrasi secara otomatis ketika sistem menyala. Ketika sudah dikalibrasi *barometer* akan menghitung ketinggian

saat ini dan dijadikan 0 meter sebagai penanda bahwa roket berada 0 meter diatas permukaan bumi. Tujuannya agar roket dapat terbang dimana saja bukan hanya dipinggir pantai. Cara kerja *barometer* yaitu dengan mendeteksi tekanan udara pada ketinggian, dengan tekanan udara tersebut dapat diketahui berapa ketinggian dari sensor tersebut. Adapun Kelemahan dari pembacaan sensor barometer adalah sensitif terhadap angin dan suhu sekitar, sehingga sensor ini perlu ditempatkan pada bagian yang tetap mendapatkan udara bebas namun tidak terdapat angin agar mendapatkan nilai yang akurat.

d) Remote Control

Remote Control merupakan perangkat sebagai pengendali jarak jauh. *Remote Control* ini memberikan 5 data input terhadap *flightcontroller* yang digunakan untuk mengontrol gerakan roket *roll, pitch, yaw*, kecepatan motor dan *switch mode*. *Remote Control* digunakan sebagai kontrol manual yang dilakukan secara *wireless* atau tanpa kabel. Kontrol ini dilakukan secara jarak jauh karena manusia yang mengendalikan tidak dapat menjangkau roket yang sedang bermanufer atau sedang terbang. *Remote Control* yang digunakan harus dapat mengendalikan roket dengan jangkauan yang sangat jauh, karena jika jangkauan *Remote Control* yang digunakan hanya menjangkau jarak yang dekat maka ketika roket terbang terlalu tinggi atau terlalu jauh dan koneksi terputus, roket tidak dapat dikontrol secara manual.

Tujuan dari penggunaan *Remote Control* yaitu untuk mengambil alih kontrol roket dari *flightcontroller,* ketika roket melakukan misi dan *flightcontroller* mendeteksi adanya error pada sensor. Selain untuk

mengambil alih kontrol dari *flightcontroller,* peran *Remote Control* disini adalah salah satu syarat keamanan dan keselamatan bagi manusia ketika wahana terbang sedang melakukan misi maka *Remote Control* sangat dibutuhkan untuk mengontrol roket saat terbang secara manual.

e) Telemetry Radio

Telemetry radio pada sistem ini berfungsi sebagai media perantara pengiriman data perilaku roket antara roket dan *ground station*. Radio komunikasi yang akan digunakan bersifat *full-duplex*. Pemilihan komunikasi *full-duplex* dikarenakan adanya proses permintaan dan pengiriman data yang akan dilakukan secara bersamaan. XBee-PRO merupakan pilihan untuk melakukan komunikasi pengiriman data jarak jauh. XBee-PRO dibuat berdasarkan standar IEEE 802.15.4 bekerja frekuensi 2,4GHz. Frekuensi 2,4GHz dipilih karena tidak adanya keharusan dalam pendaftaran kepada badan pemerintahan terhadap perangkat dengan komunikasi tersebut. XBee-PRO memiliki jarak jangkauan hingga 1 *mile* (sekitar 1600 meter) dalam keadaan outdoor *Line-of-Sight* (LOS). Pada Gambar 3.3 menunjukkan bentuk fisik dari radio telekomunikasi XBee-pro.

Gambar 3.3 *Radio Modem XBee-PRO*

2. Bagian-bagian pada proses blok diagram

a) Mikrokontroler

Mikrokontroler berisikan algoritma kontrol, mikrokontroler akan digunakan untuk melakukan beberapa tugas lain seperti menerima data dari *remote control*, mengolah data sensor, mengirimkan data sensor kepada *ground control station* dan mengatur kecepatan aktuator. Pada Gambar 3.4 menunjukkan gambar dari Mikrokontroler STM32F427.

Gambar 3.4 Mikrokontroler STM32F427

Mikrokontroler ini digunakan karena memiliki kemampuan yang cukup handal dan memiliki spesifikasi sebagai berikut:

- *Compitable* for RTOS
- 2 MB *programmable flash* yang ukurannya dianggap cukup besar dan tidak terlalu berlebih untuk algoritma yang akan ditanamkan.
- 256 KB SRAM memori yang dapat digunakan untuk menyimpan data ketika mikrokontroler dinyalakan atau bekerja, dan RAM jenis ini mempunyai kinerja yang lebih cepat.

- 4 KB EEPROM yang dapat digunakan untuk menyimpan hasil kalibrasi dari beberapa sensor.
- Tersedia komunikasi UART RS-232 yang mendukung beberapa jenis radio komunikasi.
- Mendukung protokol SPI dan I2C yang digunakan untuk berkomunikasi terhadap sensor-sensor.
- Memiliki fitur interupsi yang akan digunakan sebagai media untuk membaca data remote yang berupa pulsa dan menghasilkan PWM untuk memenuhi kebutuhan pin PWM untuk motor aktuator.
- Kecepatan prosesor dapat mencapai 180 MHz.
- Tegangan 1,7-3,6 VDC.

3. Bagian-bagian pada output blok diagram

a) Motor Servo

Pada sistem ini motor servo memiliki peran actuator untuk mengendalikan sikap roket. Motor servo digunakan pada sayap dan ekor roket sebagai pengendali gerakan *pitch*, *roll* dan *yaw* pada roket. Penggunaan motor brushless pada sistem roket akan mempengaruhi seberapa besar torsi yang harus dimiliki oleh motor servo yang digunakan. Untuk itu dipilih motor servo Servo Turnigy TGY-375DMG dengan beberapa pertimbangan sebagai berikut:

- Memiliki torsi sebesar 1,6 kg, torsi ini sudah memenuhi torsi yang diperlukan untuk fin penggerak kontrol roket, dimana berat roket tidak lebih dari 1 kg.

- Tegangan kerja 3V~6V. Dengan adanya regulator yang telah tertanam pada *Electronic Speed Control* (ESC) sebesar 5V/4A maka tegangan kerja untuk servo telah terpenuhi. Pada Gambar 3.5 menunjukkan bentuk fisik dari motor servo.

Gambar 3.5 Servo Turnigy TGY-375DMG

Berikut spesifikasi dari servo Turnigy TGY-375DMG :

Modulasi : Digital

Torsi : 4.8V : 1.6 kg-cm

6.0V : 2.3 kg-cm

kecepatan : 4.8V : 0.13 sec/60°

6.0V : 0.11 sec/60°

Dimensi : 24.5mm x 11.5mm x 23mm

b) *Electronic Speed Control* (ESC)

Perangkat ESC digunakan untuk membantu peran mikrokontroler dalam mengatur kecepatan putaran dari motor brushless. Oleh karena itu, ESC juga bisa dikatakan sebagai driver untuk mengendalikan motor brushless tanpa membebani kerja dari mikrokontroler. Berdasarkan spesifikasi motor, motor brushless yang digunakan memerlukan arus hingga 70 Ampere maka dibutuhkan ESC yang memiliki kemampuan menyuplai yang lebih dari 70

Ampere. Maka dipilih Hobbywing Skywalker 80A. ESC ini dipilih dengan beberapa pertimbangan yakni:

- Tegangan Maksimal yang mampu ditangani 2-6 Cell LiPo sehingga mampu untuk disandingkan dengan BLDC yang digunakan.
- Arus maksimal mencapai 80A sehingga mampu untuk disandingkan dengan BLDC yang meminta arus sebesar 70A.
- Adanya sistem Battery Eleminator Circuit (BEC) yang dapat digunakan sebagai catu daya sistem roket. Adapun kemampuan BEC yang tertanam memiliki tegangan keluaran sebesar 5V dengan arus sebesar 4A. Gambar 3.6 berikut ini menunjukkan bentuk fisik dari ESC.

Gambar 3.6 *Electronic Speed Control(ESC)*

c) Motor *Electric Ducted Fan* (EDF)

Motor EDF memiliki peran sebagai aktuator pemberi daya dorong pada roket motor elektrik. Motor EDF menghasilkan daya dorong dari kecepatan putaran baling-baling. Daya dorong yang dihasilkan harus lebih besar dari berat roket. Pada sistem ini, daya dorong yang diharapkan adalah sebesar minimal 2 kali dari beban roket. Motor pendorong yang digunakan bersifat motor brushless, motor EDF yang digunakan roket adalah motor Turnigy tipe 2226 dengan spesifikasi sebagai berikut:

- Tegangan: 14.8V / 4 sel LiPO

- Daya maksimum: 600 Watt

- Arus maksimum: 70 Ampere

- RPM/V: 3000 kv

- Diameter shaft: 4 mm

- Diameter EDF: 70mm

- Kekuatan dorongan: 1600 gram

Gambar 3.7 Turnigy 2226-3000 EDF
[sumber : hobbyking.com]

4. *Ground Control Station (GCS)*

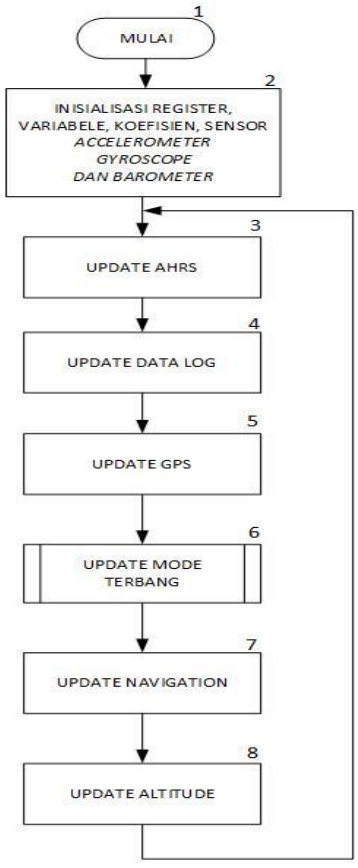
Ground Control Station (GCS) merupakan sebuah *software* penerima data sensor dan data perilaku roket yang dikirim oleh mikrokontroler melalui radio komunikasi pada roket dan diterima oleh radio komunikasi pada komputer yang akan ditampilkan pada *ground control station(GCS).* Data yang diterima akan di tampilkan berupa nilai dan visualisasi berupa pergerakan sikap roket pada saat terkoneksi melalui radio komunikasi.

3.3 Perancangan Perangkat Lunak

Pada perancangan perangkat lunak merupakan bagian implementasi algoritma kontrol yang dibuat didalam *firmware* untuk ditanam pada perangkat mikrokontroler.

3.3.1 *Flowchart*

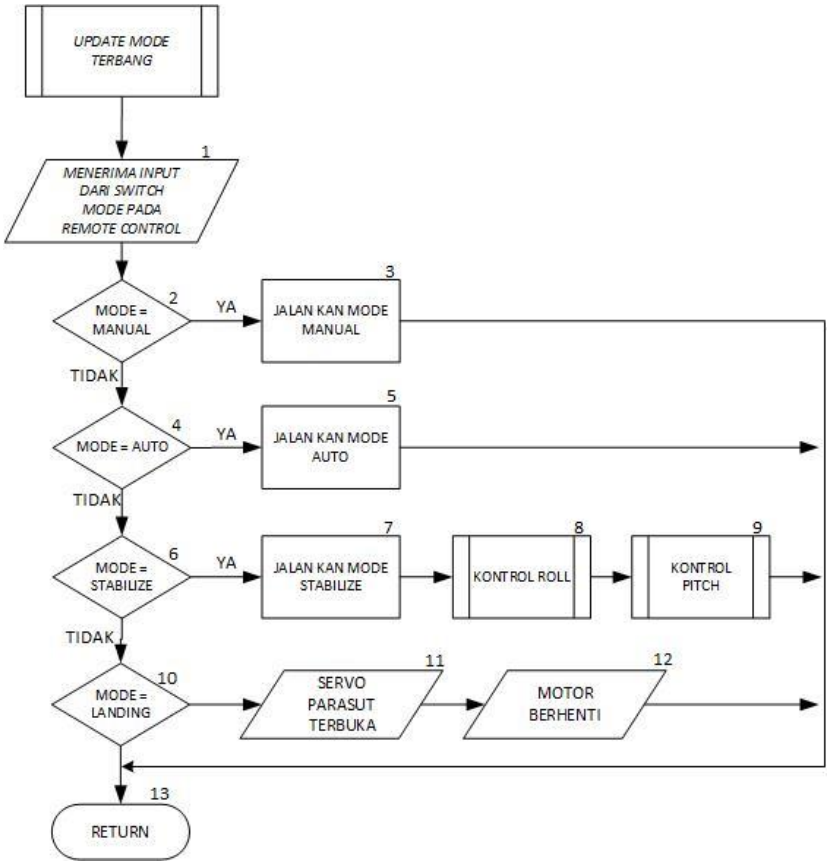
Flowchart merupakan langkah pemograman yang dapat dimengerti oleh banyak orang. Agar dapat memahami *firmware* sistem kestabilan roket maka penulis akan menunjukan terlebih dahulu *flowchart* dari *firmware* roket EDF. *Flowchart* ini merupakan main program dari *firmware* roket EDF. *Flowchar* akan ditunjukan pada Gambar 3.8, Gambar 3.9, Gambar 3.10 dan Gambar 3.11.



Gambar 3.8 *Flowchart* program utama

Berikut penjelasan mengenai *flowchart* pada gambar 3.9 :

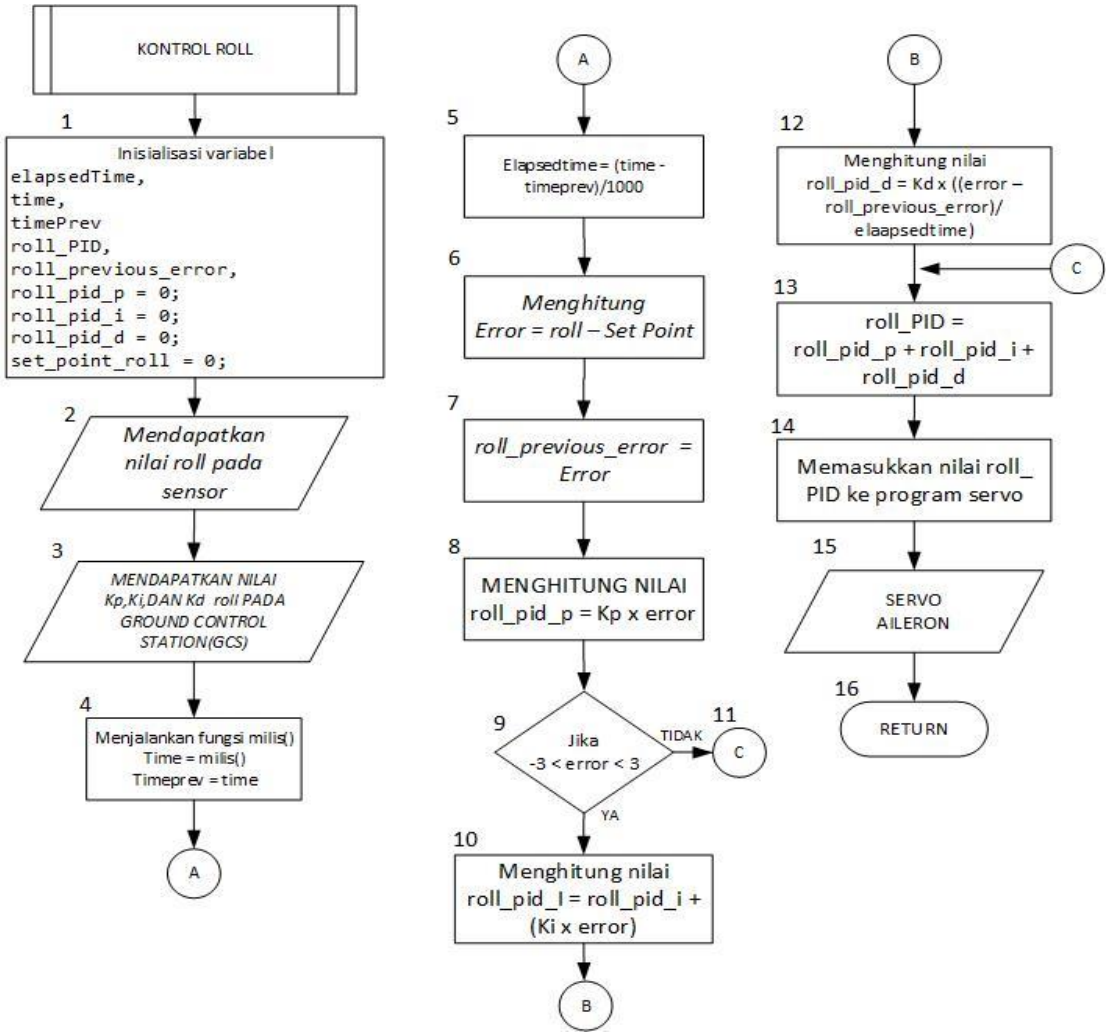
1. Memulai program.
2. Inisilisasi register, variable, koefisien, sensor *accelerometer, gyroscope dan barometer.*
3. Update data AHRS.
4. Update data log.
5. Update data gps.
6. Panggil prosedur update mode terbang.
7. Update data navigasi.
8. Update data altitude merupakan upadate data ketinggian.



Gambar 3.9 *Flowchart* sub program *update mode terbang.*

Berikut penjelasan mengenai *flowchart* pada gambar 3.10 :

1. Menerima input dari *Switch mode* pada *remote control* untuk memilih mode yang akan dijalankan.
2. Mode roket = Manual, jika benar maka masuk ke no 3 dan jika salah masuk ke no 4.
3. Menjalankan mode manual pada roket.
4. Mode roket = *auto*, jika benar maka masuk ke no 5 dan jika salah masuk ke no 6.
5. Menjalankan mode auto pada roket.
6. Mode roket = *stabilize*, jika benar maka masuk ke no 7 atau jika salah masuk ke no 10.
7. Menjalankan mode *stabilize* pada roket lalu masuk ke no 8.
8. Menjalankan sub program kontrol *roll* pada mode *stabilize*.
9. Menjalankan sub program kontrol *pitch* pada mode *stabilize*.
10. Mode roket = *landing*, jika benar maka masuk ke no 11 atau jika salah masuk ke no 13.
11. Servo parasut sebagai output akan membuka parasut.
12. Motor sebagai output akan berhenti.
13. Kembali ke program utama.



Gambar 3.10 *Flowchart* sub program kontrol *roll.*

Berikut penjelasan mengenai *flowchart* pada gambar 3.11

1. Inisialisasi variable pada program kontrol *roll.*
2. Memasukkan nilai *roll* dari data sensor pada sumbu *roll* yang telah diolah.
3. Memasukkan nilai Kp = konstanta proporsional, Ki = konstanta integral, dan Kd = konstanta derivatif, yang di set pada *ground control station*(GCS).
4. Menjalankan fungsi milis atau menghitung setiap kali program berulang dan menjalankan *time* = jumlah setiap program berulang. Lalu menjadikan variabel *time* = *timeprev* dimana *timeprevius* adalah waktu sebelumnya.

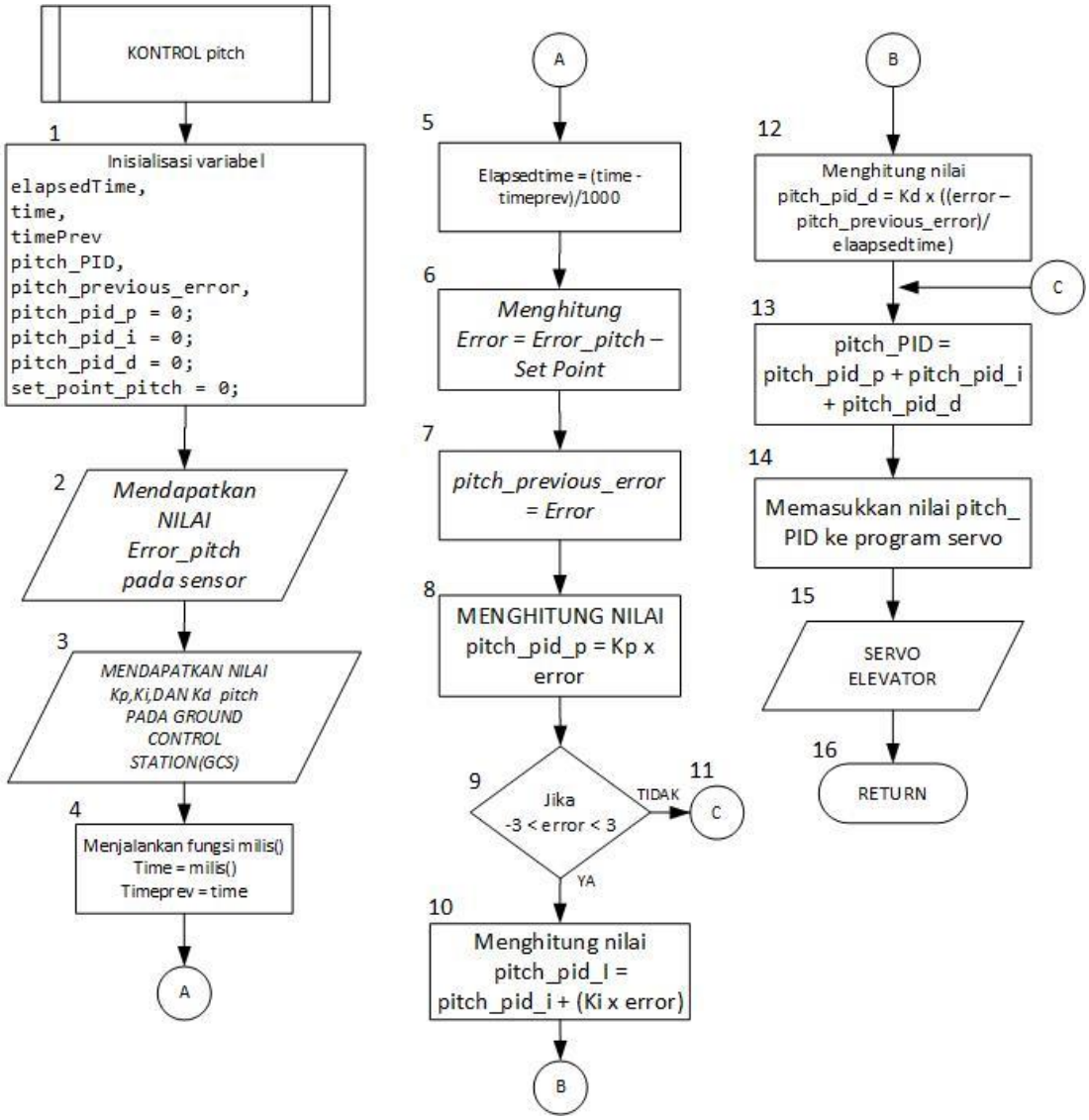
5. Menghitung nilai *elapsedtime* = $\frac{(time - timeprev)}{1000}$, dimana *elapsedtime* adalah waktu yang telah berlalu saat program berjalan.
6. Menghitung nilai *error* bertujuan untuk mendapatkan selisih nilai dari *setpoint* yang ditentukan dan nilai pada sumbu *roll* dari sensor dengan begitu dapat dihitung *error* = nilai *roll* – nilai *setpoint. Set point* adalah nilai referensi yang telah di tentukan, nilai *setpoint* = 0.
7. Mengkondisikan nilai *roll_previous_error* = nilai *error.* Dimana *Roll-_previous_error* adalah *error roll* sebelumnya.
8. Menghitung nilai proposional(P) untuk sumbu *roll*,

   roll_pid_P = Kp x *error.*
9. Membandingkan nilai *error,* apabila nilai *error* berada pada rentang nilai (-3) sampai dengan 3 , jika benar masuk ke no 10, jika tidak masuk ke no 11.
10. Menghitung nilai integral(I) untuk sumbu *roll*,

    roll_pid_I = *roll_pid_I* + (Ki x *error*).
11. Masuk ke no 13.
12. Menghitung nilai derivatif(D) untuk sumbu *roll*

    roll_pid_D = Kd x $\left(\frac{(error - roll\_previous\_error)}{elapsedtime}\right)$
13. Menghitung nilai PID untuk sumbu *roll*

    *Roll*_PID = *roll_pid_P* + *roll_pid_I* + *roll_pid_D.*
14. Memasukan nilai *Roll_PID* ke program servo kendali sumbu *roll.*
15. Servo *aileron* sebagai output yang mengatur pergerakan roket pada sumbu *roll.*
16. Kembali ke program utama.



Gambar 3.11 *Flowchart* sub program kontrol *pitch.*

Berikut penjelasan dari *flowchart* pada gambar 3.12 :

1. Inisialisasi variable pada program kontrol *pitch.*
2. Memasukkan nilai *pitch* dari data sensor pada sumbu *pitch* yang telah diolah.
3. Memasukkan nilai Kp = konstanta proporsional, Ki = konstanta integral, dan Kd = konstanta derivatif, yang di set pada *ground control station*(GCS).
4. Menjalankan fungsi milis atau menghitung setiap kali program berulang dan menjalankan *time* = jumlah setiap program berulang. Lalu menjadikan variabel *time* = *timeprev* dimana *timeprevius* adalah waktu sebelumnya.

5. Menghitung nilai *elapsedtime* = $\frac{(time - timeprev)}{1000}$, dimana *elapsedtime* adalah waktu yang telah berlalu saat program berjalan.
6. Menghitung nilai *error* bertujuan untuk mendapatkan selisih nilai dari *setpoint* yang ditentukan dan nilai pada sumbu *pitch* dari sensor dengan begitu dapat dihitung *error* = nilai *pitch* – nilai *setpoint. Set point* adalah nilai referensi yang telah di tentukan, nilai *setpoint* = 0.
7. Mengkondisikan nilai *pitch_previous_error* = nilai *error*. Dimana *Pitch-_previous_error* adalah *error pitch* sebelumnya.
8. Menghitung nilai proposional(P) untuk sumbu *pitch*,

   pitch_pid_P = Kp x *error.*
9. Membandingkan nilai *error,* apabila nilai *error* berada pada rentang nilai (-3) sampai dengan 3 , jika benar masuk ke no 10, jika tidak masuk ke no 11.
10. Menghitung nilai integral(I) untuk sumbu *pitch*,

    pitch_pid_I = *pitch_pid_I* + (Ki x *error*).
11. Masuk ke no 13.
12. Menghitung nilai derivatif(D) untuk sumbu *pitch*

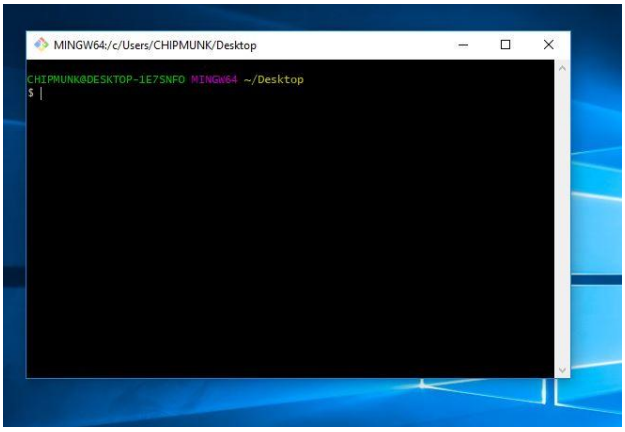
    pitch_pid_D = Kd x $\left(\frac{(error - roll\_previous\_error)}{elapsedtime}\right)$
13. Menghitung nilai PID untuk sumbu *pitch*

    *Pitch*_PID = *pitch_pid_P* + *pitch_pid_I* + *pitch_pid_D.*
14. Memasukan nilai *Pitch_PID* ke program servo kendali sumbu *pitch.*
15. Servo *elevator* sebagai output yang mengatur pergerakan roket pada sumbu *pitch.*
16. Kembali ke program utama.

Perancangan *firmware* dimulai dengan mengunduh program ardupilot menggunakan *software* git, kemudian setelah file program ardupilot selesai diunduh baru masuk pada pembuatan *firmware*. Tujuan mengunduh program ardupilot yaitu untuk melengkapi *firmware* yang akan dibuat, karena *firmware* yang dibuat hanya pada bagian kontrol kestabilan saja. Oleh karena itu *firmware*, *library* dan kelengkapan file lain untuk membuat *firmware* diambil dari program ardupilot.

3.4 Proses membuat *firmware* PX4

Proses membuat file menjadi tipe file px4 cukup mudah namun membutuhkan waktu yang cukup lama dan jaringan internet yang cukup. Berikut proses pembuatannya.

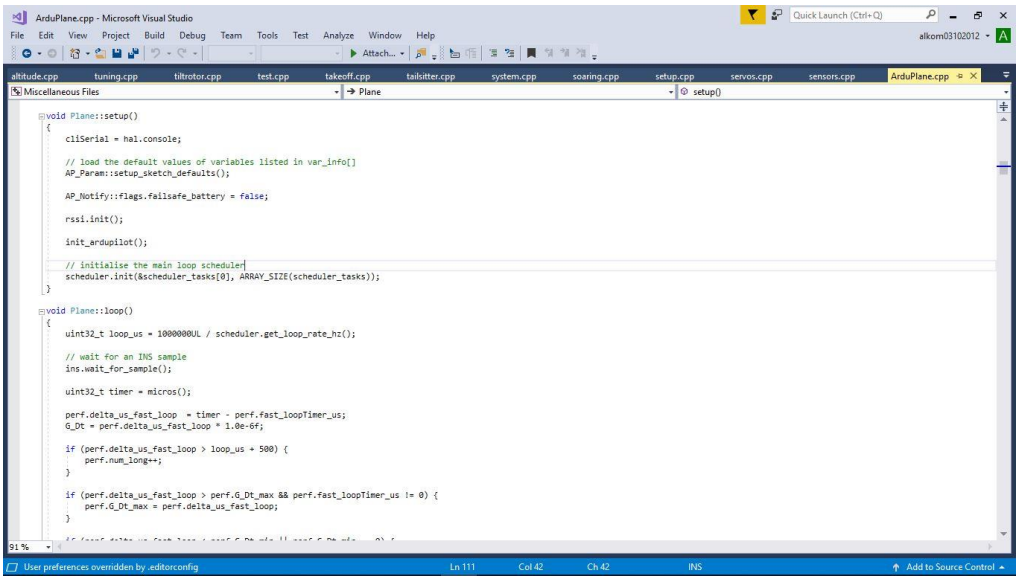


Gambar 3.12 Tampilan *software git bash*

Pada tampilan *software git bash* di gambar 3.13 masukan perintah "git clone/ardupilot/ardupilot"[8]. *Git clone* adalah perintah untuk mengambil kumpulan file-file yang diinginkan didalam alamat repository yang ingin diambil. Berikut hasil dari perintah *clone* seperti pada gambar 3.14

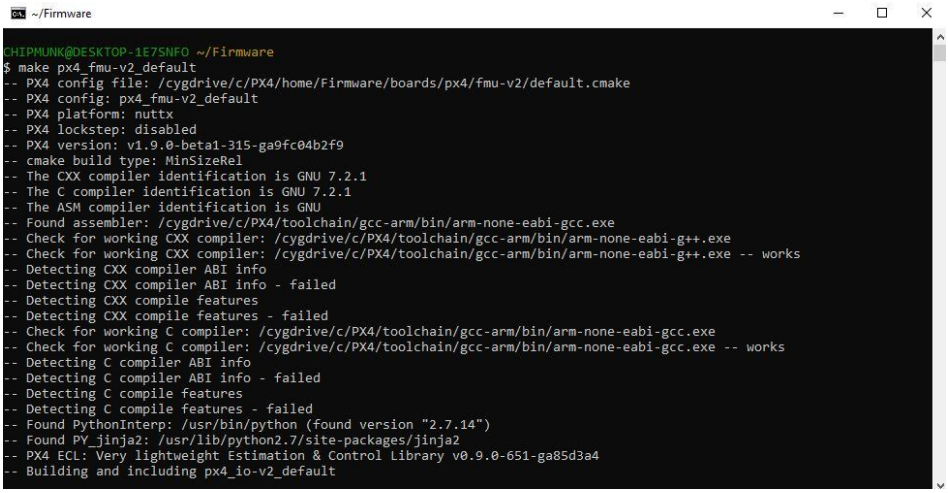
Gambar 3.13 Tampilan *git bash* setelah melakukan *clone*

Ketika *clone* file telah selesai file yang telah diunduh berupa file bertipe file cpp, file-file tersebut akan tersimpan pada folder yang telah kita pilih untuk disimpan. File-file yang telah diunduh dapat kita gunakan untuk membuat *firmware* px4, untuk membuat program yang dibutuhkan dapat menggunakan *software* visual studio. Berikut ini tampilan *software* visual studio untuk membuat program yang akan dibuat menjadi *firmware*



Gambar 3.14 Tampilan *software* visual studio

Setelah file yang telah dibuat dari hasil *clone* dari git menggunakan *software* visual studio, lalu di *build* menjadi file PX4 menggunakan *software* PX4 *toolchain.* Berikut *software* PX4 *toolchain* seperti gambar 3.16



Gambar 3.15 Tampilan *software px4 toolchain*

Kemudian masukan perintah “make px4” setelah itu *software* akan menjalankan program untuk membuat file tipe px4 yang nantinya akan di masukkan kedalam *flightcontroller.*