Simulasi *Hardware in the Loop* untuk Redundansi Tiga Modul Autopilot

Hardware in the Loop Simulation for the Redundance of Three Autopilot Modules

M. Rasyid Al Ghafar[1], Widyawardana Adiprawita[2]

Program Studi Teknik Elektro - STEI
Institut Teknologi Bandung, Jalan Ganesha 10, Bandung 40132, Indonesia
Email: [1]rasyidghafar@gmail.com, [2]wadiprawita@gmail.com

Abstrak. Makalah ini menjelaskan tentang perancangan dan pengujian *Hardware in the loop* (HIL) untuk tiga modul *autopilot* yang diredundansi. Redundansi autopilot merupakan upaya dalam mendesain suatu sistem yang tahan terhadap kegagalan pada sistem yang mungkin timbul pada pesawat udara, khususnya pesawat udara tanpa awak (*UCAV*). Pengujian HIL dilakukan dengan bantuan *software X-plane* dan *Qgroundcontrol*, dimaksudkan untuk uji kelayakan *autopilot* sebelum diimplementasikan pada perangkat atau pesawat sesungguhnya. Protokol komunikasi *transfer* data yang digunakan adalah UDP *to serial* dan *serial to* UDP yang dilakukan secara *realtime*. Dalam pengujian, modul redundansi *autopilot* telah berjalan dan terintegrasi dengan baik.

Kata Kunci: *HIL, redundansi, autopilot, X-plane, Qgroundcontrol, UDP* dan *serial.*

Abstract. *This study explained about the design and testing of Hardware in the loop (HIL) for three redundant autopilot modules. Autopilot redundancy is the means to design a system that is resistant of possible system failure that can occur on airplanes, especially Unmanned Combat Aerial Vehicle (UCAV). The HIL test was conducted with the help of X-Plane and Qgroundcontrol software, that were meant to conduct the autopilot feasibility test before being implemented on the device or the real airplane. The data transfer communication protocols that were being used was the UDP to serial and serial to UDP that were conducted in realtime. In the test, the redundancy autopilot module have functioned and being integrated well.*

Keywords: *HIL, redundancy, autopilot, X-plane, Qgroundcontrol, UDP, serial*

1 Pendahuluan

Keberadaan suatu negara tidak dapat lepas dari sistem pertahanan yang ada di negara tersebut. Sistem pertahanan yang kuat sangat menunjang adanya kedamaian dan ketentraman di negara tersebut. Pada era yang serba modern ini, teknologi sangat berperan penting dalam membangun sistem pertahanan yang kuat. Untuk bagian pertahanan udara sendiri, saat ini negara-negara sedang berlomba-lomba untuk mengembangkan pesawat tempur tanpa awak atau yang sering disebut dengan *Unmanned Combat Aerial Vehicle* (UCAV). Karena pesawat tersebut diterbangkan tanpa menggunakan awak, maka diperlukan suatu sistem autopilot yang mampu menjalankan misi-misi dan dapat dikendalikan dari *groundcontrol*. Keberadaan sistem autopilot sendiri sebenarnya sudah ada di Indonesia, akan tetapi sistem tersebut tidak terdapat toleransi *fault*, padahal dalam penerapannya sebuah pesawat tempur akan mengalami berbagai kegagalan sistem akibat pertempuran dan semacamnya.

Oleh karena itu, diperlukan sebuah sistem redundansi *autopilot* yang memiliki kehandalan tinggi dan tahan terhadap *fault* sehingga pesawat dapat tetap berjalan dengan baik meskipun

Dikirmkan 18 Desember 2017, Revisi 2 Januari 2018, Diterima 7 Februari 2018

terdapat gangguan-gangguan pada sistem. Redundansi *autopilot* secara umum dapat dideskripsikan sebagai upaya dalam mendesain suatu sistem yang tahan terhadap kegagalan atau *fault* yang mungkin timbul pada pesawat udara. Dengan begitu, dari segi fungsional, sistem pesawat dapat lebih dioptimalkan. Selain itu, dengan mencegah kerusakan sistem, maka kecelakaan-kecelakaan akibat kerusakan dan keteledoran pilot dari pesawat udara dapat diminimalisir pula. Untuk menghindari kecelakaan saat terbang, diperlukan juga langkah pengujian pada modul redundansi autopilot. Langkah pengujian dapat berupa simulasi maupun dengan pesawat yang sudah dilengkapi dengan suatu bantalan sehingga apabila *fault* terjadi dan sistem tidak dapat menangani, maka pesawat tersebut tidak akan rusak saat jatuh. Untuk itu, selain dibuat modul redundansi, perlu dibuat suatu proses simulasi sehingga sebelum diterapkan pada pesawat sebenarnya, modul redundansi tersebut sudah benar-benar dalam keadaan baik dimana saat salah satu *autopilot* mengalami *fault*, maka *autopilot* lain yang dalam kondisi sehat dapat mengambil alih sistem.

2 Teori Pendukung

2.1 HIL Simulasi

Hardware in the Loop Simulation (HIL) merupakan mode simulasi dimana *autopilot* terhubung ke *simulator* dan semua kode penerbangan dilakukan oleh *autopilot*. Simulasi ini berfungsi untuk menguji kode penerbangan aktual pada pengujian langsung. Terdapat beberapa *software* yang dapat digunakan untuk simulasi HIL *autopilot*, diantaranya adalah *mission planner, apm planner* dan *qgroundcontrol. Firmware* yang digunakan pada *software* tersebut yaitu *arduplane* dan PX4. Generasi terbaru dari *mission planner* dan *apm planner* tidak dapat digunakan untuk pengujian HIL serta hanya dapat menjalankan *firmware arduplane.* Generasi terbaru *software qgroundcontrol* masih dapat digunakan untuk simulasi HIL.

2.2 *Qgroundcontrol*

Ground Control Station (GCS) memegang peranan penting dalam pengoperasian sebuah pesawat tanpa awak. Diperlukan komunikasi yang baik antara GCS dengan *board autopilot* yang terpasang pada *platform* agar memungkinkan pengguna memberikan input perintah berupa misi penerbangan maupun memantau kondisi penerbangan *platform* tersebut. *Qgroundcontrol* merupakan salah satu *software* GCS dengan basis pemrograman XML. *Software* tersebut digunakan karena sudah terintegrasi dengan *main controller* STM32 yang terdapat dapam *pixhawk* dan FPGA *cyclone* IV. Bahasa yang digunakan pada GCS adalah XML, sedangkan *platform* dari *board autopilot* menggunakan bahasa tingkat tinggi C++. Oleh sebab itu dibutuhkan sebuah *firmware* yang di dalamnya terdapat sebuah file konfigurasi berbasis bahasa XML yang akan memanggil fungsi pada *board* autopilot platform berlingkungan bahasa C++. Salah satu *firmware* yang dapat melakukan tugas tersebut adalah PX4.

2.3 *X-plane*

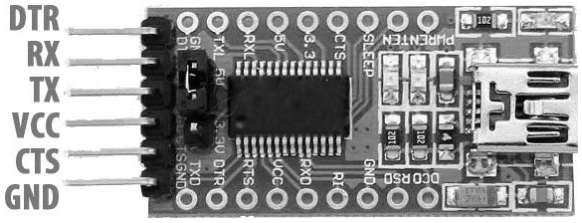
X-plane merupakan salah satu *software developer* yang dapat digunakan untuk simulasi *autopilot* sebelum dilakukan uji coba pada UCAV. *X-plane* digunakan karena sudah terintegrasi dengan *software ground control station* seperti *qgroundcontrol*, *mission planner* dan *apm planner*. Pada *X-plane*, pengguna dapat memilih pesawat yang sesuai dengan kondisi pesawat sebenarnya untuk proses simulasi. Selain itu *X-plane* direkomendasikan karena mudah dalam pengaturannya dimana pengguna hanya perlu melakukan pembuatan *port* komunikasi. Output dari sensor seperti *magnetometer* dan *gyrometer* juga dapat ditampilkan saat simulasi berlangsung.

2.4 Pixhawk

Pada perangkat yang dibuat, untuk bagian modul *autopilot* digunakan *pixhawk 3DR* sebagai modul utama. *Pixhawk* digunakan karena dalam proses *akuisisi* data, *telemetry* maupun sensornya memiliki presisi yang sangat tinggi dan lebih akurat dibandingkan dengan modul *autopilot* yang lain, sehingga nantinya apabila digunakan pada pesawat UCAV maka kestabilan dari armada tersebut dapat terjaga dengan baik. *Pixhawk* merupakan suatu modul *autopilot* dengan performasi tinggi yang cocok digunakan dalam pesawat tanpa awak *multirotor*, mobil, kapal serta beberapa *platform* robot. Alat tersebut biasanya digunakan dalam beberapa riset berteknologi canggih, perindustrian *autopilot* maupun dalam penggabungan fungsionalitas dari *PX4FMU* dan *PX4IO.*

2.5 *Future Technology Devices International (FTDI)*

Pada perangkat redundansi *autopilot*, FTDI digunakan dalam penyebaran UART dari *software qgroundcontrol* ke masing-masing modul *autopilot*. Perangkat tersebut dipilih karena di dalam modul *autopilot* digunakan *chip* FTDI sebagai *peripheral A Universal Asynchronous Receiver/Transmitter* (UART). Pin Rx digunakan untuk menyebarkan data ke *autopilot*, sedangkan pin Tx digunakan untuk mengirimkan data *autopilot* ke *qgroundcontrol*. Berikut pada Gambar 1 merupakan tampilan pin FTDI yang digunakan.



Gambar 1 Pin pada FTDI

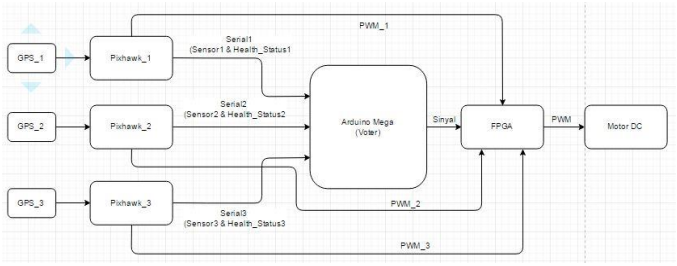
3 Perancangan

3.1 *Fault Tolerant Autopilot* dengan Sistem Redundansi

Desain sistem redundansi yaitu TMR (*Triple Modular Redundancy*). Desain ini dipilih berdasarkan beberapa pertimbangan antara lain:

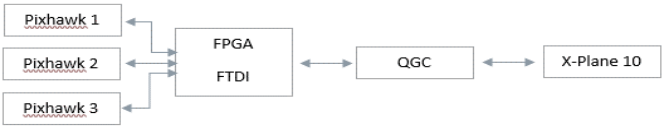
- Keandalan yang lebih tinggi dibanding *duplex*, meskipun tidak sebaik sistem *hibrid*. Akan tetapi pada sistem *hibrid* dibutuhkan rangkaian *rekonfigurasi* yang cukup kompleks.
- Kompleksitas lebih rendah dibanding dengan *hibrid*
- Karena TMR merupakan salah satu jenis NMR, maka *fault* diatasi dengan melakukan *masking* secara instan, dan karena sebagian besar *fault* yang terjadi adalah *transient fault*, maka masking adalah metode yang efektif.

Skema rancangan umum dapat dilihat pada Gambar 2 berikut.



Gambar 2 Rancangan umum sistem

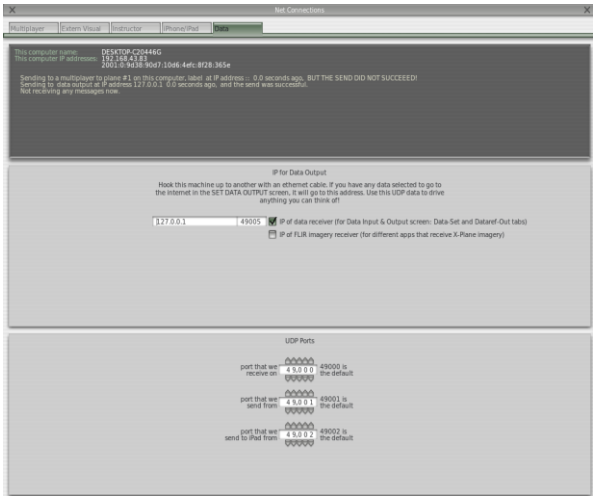
Sedangkan skema aliran data saat simulasi berjalan dapat dilihat pada Gambar 3 berikut.



Gambar 3 Aliran data sistem

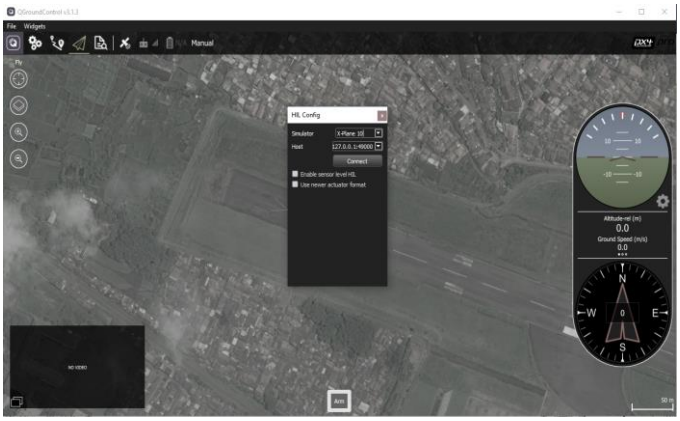
3.2 Komunikasi *Qgroundcontrol* dengan *X-plane*

Konfigurasi komunikasi *X-plane* dapat dilakukan pada menu *net connection* kemudian masuk ke sub menu *data*. Tampilan konfigurasi dapat dilihat pada Gambar 4 berikut.



Gambar 4 Pengaturan komunikasi *x-plane*

IP diisikan 127.0.0.1 saat simulasi menggunakan satu komputer. Apabila digunakan dua komputer, IP diisi sesuai dengan IP komputer yang lain. Bagian *port* diisi dengan nilai yang belum digunakan untuk komunikasi, contohnya 49005. Pada *Qgroundcontrol*, konfigurasi komunikasi dapat dilakukan pada menu *widget* kemudian masuk ke sub menu HIL. Berikut pada Gambar 5 merupakan tampilan konfigurasi pada *qgroundcontrol*.

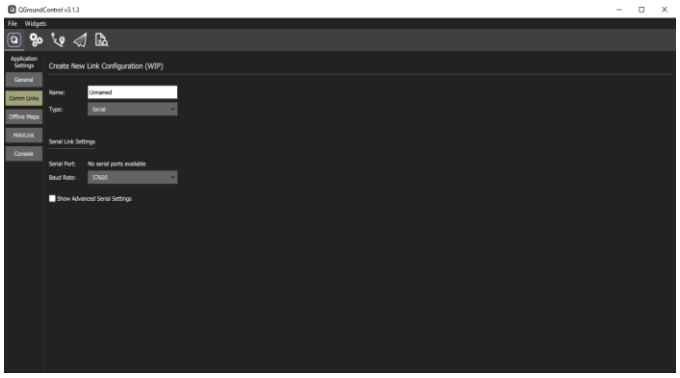


Gambar 5 Pengaturan komunikasi *Qgroundcontrol*

Bagian simulator dipilih *x-plane* 10, *host* disesuaikan dengan yang tertulis pada *x-plane. Port* pengiriman menggunakan 49000 karena secara *default software x-plane* menerima data dari *port* tersebut. *Sensor level HIL* diaktifkan dengan tujuan supaya data dari pesawat dapat digunakan sebagai acuan. Hal ini dikarenakan saat berjalannya HIL, maka kondisi dari *autopilot* adalah diam (*output* selalu 0).

3.3 Komunikasi *Qgroundcontrol* dengan Autopilot

Koneksi perlu dibuat dikarenakan modul autopilot yang digunakan sebanyak tiga buah. Pada dasarnya koneksi perangkat ke *Qgroundcontrol* dapat langsung menggunakan USB tanpa pengaturan tertentu. Akan tetapi hal tersebut berlaku jika dan hanya jika modul *autopilot* yang digunakan satu buah. Penyebaran sinyal data untuk masing-masing *autopilot* dilakukan melalui koneksi *telemetry* Rx yang terhubung paralel dengan FTDI. Berikut pada Gambar 6 merupakan tampilan dari pembuatan koneksi.



Gambar 6 Pembuatan komunikasi ke autopilot

Dalam pembuatan koneksi hal yang paling diperhatikan yaitu nilai dari *baudrate* yang digunakan. Nilai tersebut harus sama dengan nilai *baudrate* pada *source code redundancy autopilot* supaya *transfer* data *serial to USB* dapat terjalin.

3.4 Setup Autopilot

Setup autopilot dilakukan dengan menggunakan bantuan *software Qgroundcontrol*. *Setup* dilakukan untuk masing-masing *autopilot.* Hal ini bertujuan untuk mengetahui kondisi dari autopilot. Berikut merupakan urutan yang harus dilakukan.

1. *Download firmware* PX4 terbaru
2. Pemilihan jenis *airframe* yang digunakan. Untuk simulasi HIL dipilih *HILStar (Plane)*, untuk simulasi *servo* dipilih *standard plane.*
3. Kalibrasi *sensor* dan radio. Setelah kalibrasi radio selesai, pengguna dapat mengisi *flight modes* sesuai keperluan baik *manual, stabilize* maupun *mission.*
4. Pengaturan parameter dan PID *tunning.* Pada bagian ini disesuaikan dengan perangkat komputer yang digunakan. Apabila FPS saat simulasi kecil, pengaturan PID harus lebih berhati-hati supaya didapatkan jalan pesawat yang mulus saat melakukan misi.
5. Pembuatan jalur misi menggunakan *waypoint* atau *survey.*

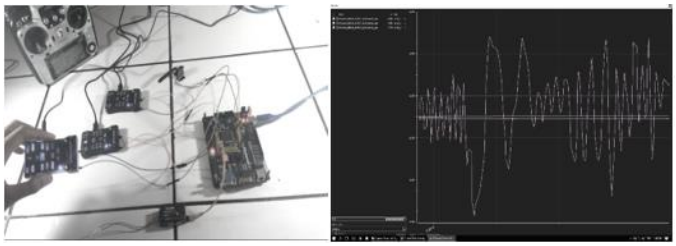
3.5 Pengujian Langsung

Proses pengujian secara langsung dilakukan dengan menyambungkan pin dari *output autopilot* dengan pind *servo.* Dalam pengujian ini masih digunakan *servo* karena dari pihak perusahaan penyelenggara tidak menyediakan pesawat siap terbang untuk prosis simulasi. Tata cara yang dilakukan dalam pengujian ini hampir sama dengan pengujian dalam HIL, akan tetapi perbedaannya terletak pada outputnya dimana *servo* dapat bergerak sesuai dengan perintah. Saat salah satu *autopilot* dibuat *error*, *servo* tetap dapat bergerak sesuai dengan perintah yang diberikan. Untuk melakukan hal tersebut pengaturan yang dilakukan secara umum sama dengan saat menggunakan HIL, perbedaannya yaitu terletak pada penekanan *switch*, dimana apabila *switch* ditekan maka *output servo* akan sesuai dengan perintah dari remote kontrol. Saat mode berubah menjadi *stabilize* maka *output servo* akan sesuai dengan arah gerak dari *autopilot.*

4 Implementasi Dan Hasil Pengujian

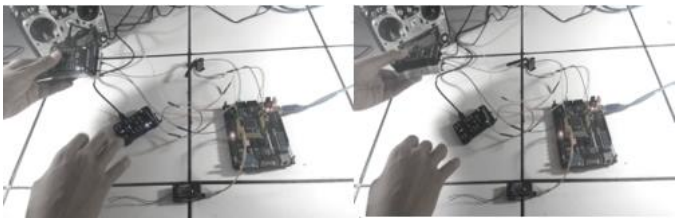
4.1 Hasil Pengujian Servo

Percobaan pertama dilakukan dengan menggerakkan secara cepat satu modul *autopilot.* Apabila servo tetap diam maka kondisi tersebut dikatakan berhasil karena *autopilot* tersebut diidentifikasi sebagai *fault*. Hal tersebut dikarenakan *output* diambil dari mayoritas *voting* modul *autopilot.* Proses kedua yaitu generasi *fault* dilakukan dengan menggerakkan dua modul *autopilot* dan membiarkan satu *autopilot* tetap diam. Kondisi berhasil apabila *servo* mengikuti gerakan dari arah gerak *autopilot*. Berikut merupakan dokumentasi hasil simulasi.



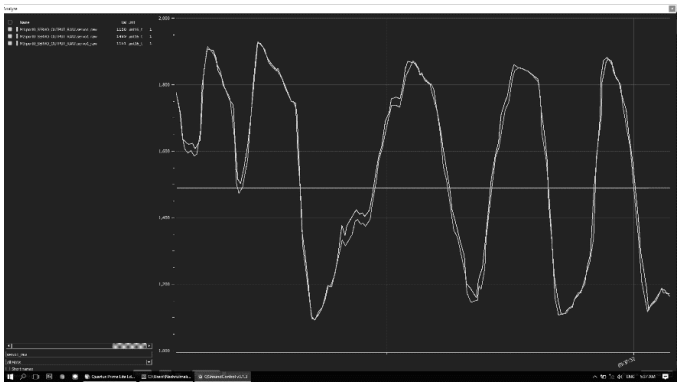
Gambar 7 Satu *servo* digerakkan beserta tampilan PWM

Terlihat pada Gambar 7 sebelah kiri bahwa saat satu *pixhawk* digerakkan maka *servo* tetap diam. Gambar 7 sebelah kanan menunjukkan *output* saat satu *pixhawk* digerakkan dan yang lain diam. *Servo* tetap diam karena mayoritas dari *autopilot* adalah diam sehingga *output* dari PWM juga otomatis akan 0.



Gambar 8 Dua *servo* digerakkan bersamaan

Terlihat pada Gambar 8 sebelah kiri, dimana saat kedua modul digerakkan ke kanan maka *servo* akan bergerak ke kanan, saat modul digerakkan ke kiri maka *servo* juga akan bergerak ke kiri. *Servo* mengikuti arah gerak dari autopilot karena *output* diambil dari mayoritas PWM. Berikut pada Gambar 9 merupakan tampilan *output* pada *Qgroundcontrol* dimana kedua modul bergerak dalam rentang PWM yang sama sedangkan modul ketiga tetap diam.



Gambar 9 PWM dua *servo* digerakkan bersamaan

Berdasarkan Gambar 8 terlihat bahwa dua *autopilot* bergerak dengan PWM yang sama. Satu *autopilot* tidak bergerak ditandai dengan *output* PWM 0.

4.2 Hasil HIL Simulasi

Pada simulasi HIL dilakukan percobaan *transient fault* dan *permanent fault* pada modul *autopilot*. *Transient fault* dilakukan dengan pelepasan sinyal Rx pada *autopilot.* Sinyal Rx disini berfungsi sebagai penerima data. Apabila dilepas maka data terbang dari pesawat tidak dapat diteruskan sehingga modul *autopilot* tersebut akan dianggap *error*. Berikut pada Gambar 10 dan Gambar 11 merupakan tampilan simulasi saat *transient fault* dilakukan.

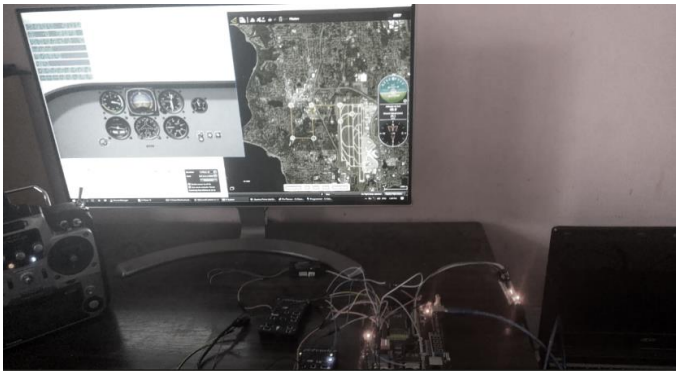
Gambar 10 *Transient fault autopilot* 1

Gambar 11 *Transient fault autopilot 2*

Terlihat pada Gambar 10 dan Gambar 11 dimana saat *autopilot* pertama atau kedua dibuat *error*, pesawat masih dapat terbang sesuai dengan misi yang diberikan. Apabila sinyal Rx disambungkan kembali maka *autopilot* akan berfungsi normal dimana data lanjutan misi akan langsung terkirim tanpa harus memuat ulang (*restart*). Hal ini membuktikan bahwa *autopilot* dapat melakukan *masking* saat terjadi *transient fault*. Selain itu juga saat terjadi *error* pada salah satu *autopilot*, maka *autopilot* yang lain akan otomatis terhubung dengan sistem ditandai dengan bergesernya nyala pada FPGA yang menandakan kondisi *autopilot* yang sedang digunakan. Hal ini membuktikan bahwa sistem redundansi telah berjalan dengan benar.

Selanjutnya yaitu *permanent fault*. *Permanent fault* dilakukan dengan melepas sambungan *power* pada *autopilot* sehingga membuat salah satu *autopilot* mati total. Kondisi berhasil apabila pesawat masih meneruskan misi yang diberikan setelah *autopilot* mati. Berikut pada Gambar 12 dan Gambar 13 merupakan tampilan saat simulasi berlangsung.

Gambar 12 *Autopilot* 3 *permanent fault*



Gambar 13 Dua *autopilot permanent fault*

Terlihat pada Gambar 12 dan Gambar 13 dimana pesawat tetap menjalankan misi sesuai dengan *waypoint* yang telah dibuat oleh pengguna baik untuk salah satu maupun dua *autopilot* mati atau *fault.* Sistem dari redundansi akan tetap bekerja dengan baik ketika minimal salah satu dari *autopilot* tetap aktif.

5 Kesimpulan

Saat terjadi *fault* pada salah satu *autopilot, autopilot* lain yang dalam keadaan sehat dapat langsung mengambil alih sistem kontrol. *Output* redundansi diambil dari mayoritas *autopilot* dalam kondisi sehat. *Autopilot* dapat melakukan *masking* saat terjadi *transient fault*. Saat terjadi *permanent fault*, kinerja dari redundansi *autopilot* tetap dalam performa baik. Akan tetapi *autopilot* yang mengalami *permanent fault* tidak dapat melakukan *masking* dengan sendirinya.

Referensi

[1] I. Koren, et.al. *Fault Tolerant Systems*. Oxford: Elsevier Inc, 2007
[2] E. F Hitt, Digital Avionics Handbook, Second Edition. Boca Raton: CRC Press, 2000
[3] E. F. Hitt, D. Mulcare, *Fault-Tolerant Avionics*, CRC Press LLC, 2001
[4] (2017, April 1), *Ardupilot*, http://ardupilot.org/copter/docs/common-rfd900.html, Ardupilot Dev Team
[5] (2017, April 1), *Flight stack*, http://px4.io/technology/px4-software-stack/
[6] L. Meier, (2017, April 1), *Pixhawk Flight Controller Hardware Project*, ETH Computer Vision and Geometry Lab., https://pixhawk.org/,
[7] (2017, April 1), *3DR GPS*, http://ardupilot.org/copter/docs/common-installing-3dr-ublox-gps-compass-module.html, Ardupilot Dev Team