RANCANG BANGUN
SISTEM *MONITORING* NAVIGASI *QUADCOPTER*

ABDULLAH HAJIS

5215122638

Skripsi Ini Disusun Sebagai Salah Satu Persyaratan Untuk
Memperoleh Gelar Sarjana Pendidikan

PROGRAM STUDI PENDIDIKAN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI JAKARTA

2017

HALAMAN PENGESAHAN

Nama Dosen	Tanda Tangan	Tanggal
Drs. Wisnu Djatmiko, MT NIP. 19670214 1992 03 1 001 (Dosen Pembimbing I)	.....	21 Agustus 2017
Dr. Efri Sandi, MT NIP. 19750202 200812 1 002 (Dosen Pembimbing II)	.....	16 Agustus 2017

PENGESAHAN PANITIA UJIAN SKRIPSI

Nama Dosen	Tanda Tangan	Tanggal
Dr. Moch. Sukardjo, M.Pd NIP. 19580720 198503 1 003 (Ketua Sidang)	.....	15 Agustus 2017
Drs. Pitoyo Yuliatmojo, MT NIP. 19680708 199403 1 003 (Sekretaris)	.....	21 Agustus 2017
Syufrijal, MT NIP. 19760327 200112 1 001 (Dosen Ahli)	.....	16 Agustus 2017

Tanggal Lulus : 14 Agustus 2017

HALAMAN PERNYATAAN

Dengan ini saya menyatakan bahwa :

1. Karya tulis saya dengan judul "Rancang Bangun Sistem *Monitoring* Navigasi *Quadcopter*" adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik sarjana, baik di Universitas Negeri Jakarta maupun di perguruan tinggi lain.

2. Karya tulis ini adalah murni gagasan, rumusan dan penelitian saya sendiri dengan arahan dosen pembimbing.

3. Dalam karya tulis ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.

4. Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah di peroleh karena karya tulis ini, serta sanksi lainnya sesuai dengan norma yang berlaku di Universitas Negeri Jakarta.

Jakarta, 4 Agustus 2017

Yang Membuat Pernyataan

METERAI TEMPEL

A97C6AEF554482060

6000

ENAM RIBU RUPIAH

Abdullah Hajis

5215122638

KATA PENGANTAR

Puji dan syukur peneliti panjatkan kepada Allah Subhanahu Wa Ta'ala atas segala Rahmat dan Karunia-Nya, sehingga skripsi yang berjudul Rancang Bangun Sistem *Monitoring* Navigasi *Quadcopter* dapat diselesaikan. Penelitian ini dilaksanakan bertujuan untuk memenuhi salah satu syarat kelulusan dalam menyelesaikan studi Strata 1, Program Studi Pendidikan Teknik Elektronika, Fakultas Teknik, Universitas Negeri Jakarta. Peneliti menyadari bahwa tanpa dukungan, arahan dan bimbingan dari berbagai pihak, begitu sulit bagi peneliti untuk menyusun skripsi ini. Dengan demikian, peneliti mengucapkan terimakasih kepada :

1. Drs. Pitoyo Yuliatmojo, MT, selaku Ketua Program Studi Pendidikan Teknik Elektronika yang telah memberikan dukungan dan motivasi serta ilmu yang bermanfaat.
2. Drs. Wisnu Djatmiko, MT, selaku Dosen Pembimbing I atas segala ketulusan dan kesabaran dalam membimbing skripsi serta memberikan ilmu yang bermanfaat.
3. Dr. Efri Sandi, MT, selaku Dosen Pembimbing II atas segala ketulusan dan kesabaran dalam membimbing skripsi serta memberikan ilmu yang bermanfaat.
4. Ayahanda Mulyono dan Ibunda Darsem beserta keluarga saya yang selalu memberikan kasih sayang yang tidak ternilai harganya dan juga atas doa yang selalu diucapkan.
5. Dan seluruh pihak yang telah membantu saya dalam menyelesaikan skripsi ini, yang tidak dapat saya sebutkan satu per satu.

Akhir kata, semoga Allah Subhanahu Wa Ta'ala membalas segala kebaikan semua pihak yang telah membantu dalam penyusunan skripsi ini dengan balasan yang lebih baik. Semoga skripsi ini bermanfaat besar bagi pengembangan ilmu pengetahuan dan teknologi di Indonesia, khususnya bagi pembaca.

Jakarta, 4 Agustus 2017

Peneliti

ABSTRAK

Abdullah Hajis. Rancang Bangun Sistem *Monitoring* Navigasi *Quadcopter*. Skripsi. Jakarta, Program Studi Pendidikan Teknik Elektronika, Fakultas Teknik, Universitas Negeri Jakarta, 2017. Dosen Pembimbing : Drs. Wisnu Djatmiko, MT dan Dr. Efri Sandi, MT.

Penelitian bertujuan untuk merancang, merealisasikan dan menguji Sistem *Monitoring* Navigasi *Quadcopter* berbasis Arduino Nano menggunakan *software* Processing IDE untuk memantau pergerakan dan performa sistem *quadcopter* ketika terbang di udara. Penelitian menggunakan metode penelitian riset dan pengembangan. Penelitian dilaksanakan di Laboratorium Mekatronika dan Robotika dan Laboratorium Bengkel Mekanik, Fakultas Teknik, Universitas Negeri Jakarta pada bulan September 2016 s.d. Juni 2017.

Sistem *Monitoring* Navigasi *Quadcopter* terdiri dari subsistem *monitoring* pada *quadcopter* dan subsistem *Ground Control Station* (GCS). Subsistem *monitoring* pada *quadcopter* terdiri dari modul Arduino Nano, *Power Module*, Modul IMU (*Inertial Measurement Unit*) 10 DOF, modul *Speed Sensor*, dan modul XBee *Transmitter*. Sedangkan subsistem GCS terdiri dari modul Arduino Nano, modul XBee *Receiver*, *Laptop*, dan *software* Processing IDE. Sistem *Monitoring* Navigasi *Quadcopter* dapat memonitor 10 parameter, yaitu : 1) Arus, 2) Tegangan, 3) Sudut *yaw*, 4) Sudut *pitch*, 5) Sudut *roll*, 6) *Altitude*, 7) Kecepatan motor BLDC 1, 8) Kecepatan motor BLDC 2, 9) Kecepatan motor BLDC 3, dan 10) Kecepatan motor BLDC 4.

Sistem *monitoring* navigasi *quadcopter* sudah berhasil direalisasikan, dan sudah di uji dapat mengukur parameter yang dihasilkan oleh sensor-sensor pada *quadcopter* dan dapat diterima dengan baik oleh sistem GCS dengan jarak jangkauan maksimal sejauh 40 meter. Sistem dapat memproses 10 data parameter secara *real time* dengan *update* 200 ms. Kemudian subsistem GCS dapat mengukur dan menampilkan parameter dengan rentang ukur dan hasil pengujian sebagai berikut : 1) Rentang ukur tegangan hingga 12,6V dengan rata-rata *error* 0,215%, 2) Rentang ukur arus hingga 10A dengan rata-rata *error* 2,65%, 3) Rentang ukur sudut *yaw* dari 0º hingga 360º dengan rata-rata selisih 16,87º, 4) Rentang ukur sudut *pitch* dari -87º hingga 88º dengan rata-rata selisih 0,28º, 5) Rentang ukur sudut *roll* dari -180º hingga 180º dengan rata-rata selisih 0,57º, 6) Rentang ukur *altitude* dari 0 mdpl hingga 50 mdpl dengan rata-rata selisih 1,58 meter, 7) Rentang ukur rpm motor BLDC hingga 13440 rpm dengan masing-masing memiliki rata-rata *error*, yaitu : RPM motor 1 sebesar 1,29%, 8) RPM motor 2 sebesar 0,79%, 9) RPM motor 3 sebesar 1,38%, dan 10) RPM motor 4 sebesar 0,65%.

Kata-kata kunci : Sistem *Monitoring*, Navigasi, *Quadcopter*, Ardupilot, Telemetri, *Ground Control Station*, Arduino Nano, Processing IDE, IMU 10 DOF

ABSTRACT

Abdullah Hajis. Designed a Quadcopter Navigation Monitoring System. Skripsi. Jakarta, Education of Electronics Engineering, Faculty of Engineering, State University of Jakarta, 2017. Supervisor: Drs. Wisnu Djatmiko, MT and Dr. Efri Sandi, MT.

The research aims to design, realize and test the Quadcopter Navigation Monitoring System based Arduino Nano using the Processing IDE software to monitor the movement and performance of the quadcopter system when flying in the air. The research using research and development method. The research was conducted at the Mechatronics and Robotics Laboratory and Mechanical Workshop Laboratory, Faculty of Engineering, State University of Jakarta on September 2016 s.d. June 2017.

The Quadcopter Navigation Monitoring System consists of a monitoring subsystem on a quadcopter and a Ground Control Station (GCS) subsystem. The quadcopter monitoring subsystem consists of Arduino Nano module, Power Module, IMU Module (Inertial Measurement Unit) 10 DOF, Speed Sensor module, and transmitter XBee module. While the GCS subsystem consists of Arduino Nano module, receiver XBee module, Laptop, and IDE Processing software. Quadcopter Navigation Monitoring System can monitor 10 parameters, namely: 1) Current, 2) Voltage, 3) Yaw angle, 4) Pitch angle, 5) Roll angle, 6) Altitude, 7) Speed of BLDC motor 1, 8) Speed of BLDC motor 2, 9) Speed of BLDC motor 3, and 10) Speed of BLDC motor 4.

The Quadcopter navigation monitoring system has been succesfully realized, and tested to measure the parameters generated by the sensors on the quadcopter and can be well received by the GCS system with a maximum range of 40 meters. The system can process 10 data parameters in real time with an update of 200 ms. Then the GCS subsystem can measure and display 10 parameters with measuring ranges and test results as follows : 1) Voltage measuring range up to 12.6V with an average error of 0.215%, 2) Current measuring range up to 10A with an average error of 2.65%, 3) Yaw angle measuring range from 0º to 360º with an average error of 16.87º, 4) Pitch angle measuring range from -87 º to 88 º with an average error of 0.28 º, 5) Roll angle measuring range from -180º to 180º with an average error of 0.57º, 6) Altitude measuring range from 0 mdpl to 50 mdpl with an average error of 1.58 meters, 7) The measurement range of BLDC motor rpm up to 13440 rpm with each having an average error, namely: RPM of motor 1 by 1.29%, 8) RPM of motor 2 by 0,79%, 9) RPM of motor 3 by 1.38%, and 10) RPM of motor 4 by 0,65%.

Keywords: Monitoring System, Navigation, Quadcopter, Ardupilot, Telemetry, Ground Control Station, Arduino Nano, Processing IDE, IMU 10 DOF

DAFTAR ISI

Halaman Judul..... i
Halaman Pengesahan..... ii
Halaman Pernyataan..... iii
Kata Pengantar..... iv
Abstrak..... v
Abstract..... vi
Daftar Isi..... vii
Daftar Tabel..... xi
Daftar Gambar..... xii
Daftar Lampiran..... xiv

BAB I
PENDAHULUAN
1.1 Latar Belakang Masalah..... 1
1.2 Identifikasi Masalah..... 2
1.3 Pembatasan Masalah..... 2
1.4 Perumusan Masalah..... 3
1.5 Tujuan Penelitian..... 3
1.6 Manfaat Penelitian..... 3

BAB II
TINJAUAN PUSTAKA
2.1 Kerangka Teoritik..... 4
2.1.1 Definisi Sistem *Monitoring* Navigasi *Quadcopter*..... 4
2.1.1.1 Sistem..... 4
2.1.1.2 *Monitoring*..... 4
2.1.1.3 Navigasi..... 5
2.1.1.4 *Quadcopter*..... 5
2.1.2 Definisi Arduino Nano..... 7
2.1.2.1 Arduino..... 7
2.1.2.2 Arduino Nano..... 8
2.1.2.3 Arduino IDE..... 10
2.1.3 Definisi *Software* Processing IDE..... 11
2.1.3.1 *Software*..... 11
2.1.3.2 Processing..... 12
2.1.3.3 IDE (*Integrated Development Environment*)..... 14
2.1.3.4 Processing IDE..... 14
2.1.4 Definisi Sistem *Monitoring* Navigasi *Quadcopter* berbasis Arduino Nano menggunakan *Software* Processing IDE..... 15
2.1.5 Spesifikasi tiap Parameter *Quadcopter* yang akan di-*monitoring*.... 16
2.1.6 Ardupilot APM 2.8..... 18
2.1.7 *Frame* F450Q..... 19
2.1.8 Motor *Brushless*..... 20
2.1.9 *Propeller*..... 22

2.1.10 *Electronic Speed Controller*..... 22
2.1.11 Baterai *Lithium Polymer*..... 24
2.1.12 *Inertial Measurement Unit*..... 25
2.1.12.1 Akselerometer..... 25
2.1.12.2 Giroskop..... 26
2.1.12.3 Magnetometer..... 27
2.1.12.4 Barometer..... 27
2.1.13 Modul LM393 *Infrared Speed Sensor*..... 28
2.1.14 3DR *Power Module*..... 29
2.1.15 Telemetri..... 31
2.1.16 *Radio Control*..... 32
2.1.17 XBee *Pro* S1..... 33
2.1.18 XBee *Adapter*..... 37
2.1.19 *Ground Control Station*..... 38
2.2 Konsep Pengembangan Produk..... 39
2.2.1 Pengertian Pengembangan Produk..... 39
2.2.2 Langkah-langkah Penelitian dan Pengembangan Produk..... 40
2.2.2.1 Model Karl T. Ulrich dan Steven D. Epinger..... 40
2.2.2.2 Model Borg dan Gall..... 43
2.2.2.3 Model Sugiyono..... 47
2.2.2.4 Sintesis Langkah-langkah Penelitian dan Pengembangan Produk..... 50
2.3 Konsep Produk yang Dikembangkan..... 52
2.3.1 Mission Planner..... 54
2.3.2 OpenPilot GCS..... 55
2.3.3 QGroundControl..... 56
2.3.4 Multiwii GUI..... 58
2.3.5 Rancangan *Ground Control Station* Penelitian..... 59
2.4 Rancangan Produk..... 61
2.4.1 Diagram Blok Sistem *Monitoring* Navigasi *Quadcopter*..... 61
2.4.2 Rancangan *Flowchart* Program..... 61
2.4.2.1 *Flowchart* Arduino Nano *Transmitter*..... 61
2.4.2.2 *Flowchart* Arduino Nano *Receiver*..... 63
2.4.2.3 *Flowchart* Processing IDE..... 64

BAB III
METODOLOGI PENELITIAN

3.1 Tempat dan Waktu Penelitian..... 66
3.2 Metode Pengembangan Produk..... 66
3.2.1 Tujuan Pengembangan..... 66
3.2.2 Metode Pengembangan..... 66
3.2.3 Sasaran Produk..... 68
3.2.4 Instrumen..... 68
3.2.4.1 Instrumen Penelitian..... 68
3.2.4.2 *Software* Penelitian..... 69
3.2.4.3 Alat Penelitian..... 69
3.2.4.4 Bahan Penelitian..... 70

3.3 Prosedur Pengembangan..... 70
3.3.1 Tahap Penelitian dan Pengumpulan Informasi..... 70
3.3.2 Tahap Perencanaan dan Perancangan Sistem..... 72
3.3.2.1 Perancangan Diagram Blok..... 72
3.3.2.2 Perencanaan *Input* dan *Output*..... 74
3.3.2.3 Perancangan Rangkaian Sistem *Monitoring* pada *Quadcopter*... 75
3.3.2.4 Perancangan Rangkaian Sistem *Ground Control Station*..... 75
3.3.2.5 Perancangan *Flowhart* Program..... 76
a. Flowchat Arduino Nano *Transmitter*..... 76
b. Flowchat Arduino Nano *Receiver*..... 77
c. Flowchart Processing IDE..... 78
3.3.3 Tahap Desain Produk..... 80
3.3.3.1 Desain *Layout Shield* Modul Sensor pada *Quadcopter*..... 80
3.3.3.2 Desain Tampilan Antarmuka *Ground Control Station*..... 81
3.4 Teknik Pengumpulan Data..... 82
3.5 Teknik Analisis Data..... 82
3.5.1 Pengujian Koneksi GCS dengan *Quadcopter*..... 83
3.5.2 Pengujian Jarak Jangkauan Komunikasi..... 83
3.5.3 Pengujian Tegangan Baterai *Quadcopter*..... 84
3.5.4 Pengujian Arus Baterai *Quadcopter*..... 86
3.5.5 Pengujian Sudut *Yaw Quadcopter*..... 87
3.5.6 Pengujian Sudut *Pitch Quadcopter*..... 87
3.5.7 Pengujian Sudut *Roll Quadcopter*..... 88
3.5.8 Pengujian Altitude *Quadcopter*..... 89
3.5.9 Pengujian RPM Motor *Brushless Quadcopter*..... 90

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN
4.1 Hasil Pengembangan Produk..... 91
4.1.1 Hasil Rancangan *Hardware* Sistem *Monitoring*..... 91
4.1.1.1 Hasil Rancangan *Shield* untuk Modul-modul Sensor..... 91
4.1.1.2 Hasil Rancangan Penempatan Modul-modul *Quadcopte*r..... 92
4.1.1.3 Hasil Rakitan dan *Wiring Quadcopter*..... 94
4.1.2 Hasil Rancangan *Software* Sistem *Monitoring* (GCS)..... 96
4.2 Kelayakan Produk (Teoritik dan Empiris)..... 98
4.3 Efektifitas Produk (Melalui Uji Coba)..... 98
4.3.1 Hasil Pengujian Koneksi GCS dengan *Quadcopter*..... 98
4.3.2 Hasil Pengujian Jarak Jangkauan Komunikasi..... 100
4.3.3 Hasil Pengujian Tegangan Baterai *Quadcopter*..... 100
4.3.4 Hasil Pengujian Arus Baterai *Quadcopter*..... 101
4.3.5 Hasil Pengujian Sudut *Yaw Quadcopter*..... 102
4.3.6 Hasil Pengujian Sudut *Pitch Quadcopter*..... 103
4.3.7 Hasil Pengujian Sudut *Roll Quadcopter*..... 103
4.3.8 Hasil Pengujian *Altitude Quadcopter*..... 104
4.3.9 Hasil Pengujian RPM Motor *Brushless Quadcopter*..... 105
4.4 Pembahasan..... 106
4.4.1 Kinerja Sistem *Hardware*..... 106

4.4.2 Kinerja Sistem *Software* pada Arduino..... 107
4.4.3 Kinerja Sistem *Software* pada GCS..... 108
4.4.4 Pengujian Sistem Secara Keseluruhan..... 108
4.4.5 Kelebihan dan Kekurangan Produk..... 108
4.4.5.1 Kelebihan Produk..... 108
4.4.5.2 Kekurangan Produk..... 109

BAB V
KESIMPULAN DAN SARAN
5.1 Kesimpulan..... 110
5.2 Saran..... 111

DAFTAR PUSTAKA..... 113

LAMPIRAN..... 117

DAFTAR TABEL

2.1 Dinamika Gerak *Quadcopter*..... 6
2.2 Konfigurasi pin Arduino Nano..... 9
2.3 Spesifikasi Arduino Nano..... 10
2.4 Fitur Arduino Nano..... 10
2.5 Spesifikasi *Frame* F450 *Quadcopter*..... 20
2.6 Spesifikasi 3DR *Power Module*..... 30
2.7 Spesifikasi Modul XBee *Pro* S1..... 34
2.8 Fitur Modul XBee *Pro* S1..... 34
2.9 Konfigurasi pin XBee..... 35
3.1 Instrumen Penelitian..... 68
3.2 Software Penelitian..... 69
3.3 Alat Penelitian..... 69
3.4 Bahan Penelitian..... 70
3.5 *Input Output* Sistem *Monitoring* pada *Quadcopter*..... 74
3.6 *Input Output* Sistem *Monitoring* pada GCS..... 74
3.7 Pengujian Koneksi GCS dengan *Quadcopter*..... 83
3.8 Pengujian Jarak Jangkauan Komunikasi..... 84
3.9 Pengujian Tegangan Baterai *Quadcopter*..... 85
3.10 Pengujian Arus Baterai *Quadcopter*..... 86
3.11 Pengujian Sudut *Yaw Quadcopter*..... 87
3.12 Pengujian Sudut *Pitch Quadcopter*..... 88
3.13 Pengujian Sudut *Roll Quadcopter*..... 89
3.14 Pengujian *Altitude Quadcopter*..... 90
3.15 Pengujian RPM Motor *Brushless Quadcopter*..... 90
4.1 Hasil Pengujian Koneksi GCS dengan *Quadcopter*..... 99
4.2 Hasil Pengujian Jarak Jangkauan Komunikasi..... 100
4.3 Hasil Pengujian Tegangan Baterai *Quadcopter*..... 101
4.4 Hasil Pengujian Arus Baterai *Quadcopter*..... 101
4.5 Hasil Pengujian Sudut *Yaw Quadcopter*..... 102
4.6 Hasil Pengujian Sudut *Pitch Quadcopter*..... 103
4.7 Hasil Pengujian Sudut *Roll Quadcopter*..... 104
4.8 Hasil Pengujian Sudut *Altitude Quadcopter*..... 105
4.9 Hasil Pengujian RPM Motor *Brushless Quadcopter*..... 106

DAFTAR GAMBAR

2.1 Dinamika Gerak *Quadcopter*..... 6
2.2 Navigasi *Quadcopter*..... 7
2.3 Bentuk Fisik dan Pin-pin Arduino Nano..... 8
2.4 *Software* Arduino IDE..... 11
2.5 *Software* Processing IDE..... 15
2.6 Ardupilot APM 2.8..... 19
2.7 *Frame* F450 *Quadcopter*..... 20
2.8 Motor BLDC..... 21
2.9 *Propeller*..... 22
2.10 *Electronic Speed Controller*..... 23
2.11 Baterai *Lithium Polimer*..... 24
2.12 Ilustrasi Pembacaan Sensor Akselerometer..... 25
2.13 Ilustrasi Gerakan Sudut *Roll*, *Pitch* dan *Yaw* Giroskop..... 26
2.14 Modul LM393 *Infrared Speed Sensor*..... 28
2.15 Skema Rangkaian LM393 *Infrared Speed Sensor*..... 29
2.16 3DR *Power Module*..... 30
2.17 Skema Rangkaian 3DR *Power Module*..... 31
2.18 *Radio Control*..... 33
2.19 Modul XBee *Pro* S1..... 34
2.20 Blok Sistem Pemancar dan Penerima XBee..... 36
2.21 Ilustrasi Pola Bit Serial Data XBee..... 36
2.22 XBee *Adapter Board*..... 37
2.23 *Software* X-CTU..... 38
2.24 Contoh Tampilan GCS pada *Laptop*..... 39
2.25 Langkah-langkah Penelitian dan Pengembangan Produk menurut Ulrich dan Epinger..... 41
2.26 Langkah-langkah Penelitian dan Pengembangan Produk menurut Borg dan Gall..... 44
2.27 Langkah-langkah Penelitian dan Pengembangan Produk menurut Sugiono..... 48
2.28 Sintesis Langkah-langkah Penelitian dan Pengembangan Produk..... 51
2.29 Mission Planner..... 55
2.30 OpenPilot GCS..... 56
2.31 Qground Control..... 57
2.32 MultiWii GUI..... 58
2.33 Rancangan *Ground Control Station*..... 59
2.34 Diagram Blok Sistem *Monitoring* Navigasi *Quadcopter*..... 61
2.35 *Flowchart* Arduino Nano *Transmitter*..... 62
2.36 *Flowchart* Arduino Nano *Receiver*..... 63
2.37 *Flowchart* Processing IDE..... 65
3.1 Metode Pengembangan..... 67
3.2 Diagram Blok Sistem *Monitoring* Navigasi *Quadcopter*..... 73
3.3 Perancangan Rangkaian Sistem *Monitoring* pada *Quadcopter*..... 75
3.4 Perancangan Rangkaian Sistem *Ground Control Station*..... 76
3.5 *Flowchart* Arduino Nano *Transmitter*..... 77

3.6 *Flowchart* Arduino Nano *Receiver*..... 78
3.7 *Flowchart* Processing IDE..... 80
3.8 Desain *Layout Shield* Modul-modul Sensor..... 81
3.9 Desain *Ground Control Station*..... 82
3.10 Metode Pengujian Tegangan Baterai *Quadcopter*..... 85
3.11 Metode Pengujian Arus Baterai *Quadcopter*..... 86
3.12 Susunan Pengujian Sudut *Yaw*..... 87
3.13 Susunan Pengujian Sudut *Pitch*..... 88
3.14 Susunan Pengujian Sudut *Roll*..... 88
4.1 *Layout Shield* untuk Modul-modul Sensor..... 92
4.2 *Shield* Tampak Atas..... 92
4.3 Penempatan *Shield*, Arduino Nano dan IMU..... 92
4.4 Penempatan Sensor RPM Motor..... 93
4.5 Penempatan GPS..... 93
4.6 Penempatan XBee Pro S1..... 93
4.7 Penempatan *Radio Receiver*..... 93
4.8 Penempatan Baterai *Lithium Polymer*..... 94
4.9 Penempatan *Power Module*..... 94
4.10 *Quadcopter* Tampak Depan..... 94
4.11 *Quadcopter* Tampak Belakang..... 95
4.12 *Quadcopter* Tampak Sisi Kanan..... 95
4.13 *Quadcopter* Tampak Sisi Kiri..... 95
4.14 *Quadcopter* Tampak Atas..... 95
4.15 Tampilan Antarmuka GCS..... 96
4.16 Tampilan GCS Tidak Terkoneksi dengan *Quadcopter*..... 98
4.17 Tampilan GCS Terkoneksi dengan *Quadcopter*..... 98

DAFTAR LAMPIRAN

Lampiran 1.	*Listing* Program pada Arduino *Transmitter*.....	117
Lampiran 2.	*Listing* Program pada Arduino *Receiver*.....	121
Lampiran 3.	*Listing* Program GCS pada Processing IDE.....	122
Lampiran 4.	*Wiring* Sistem *Quadcopter* Ardupilot APM.....	137
Lampiran 5.	Riwayat Hidup.....	138

BAB 1

PENDAHULUAN

1.1 Latar Belakang Masalah

Pesawat Tanpa Awak (*Unmanned Aerial Vehicle*, UAV) atau *Unmanned Aircraft System* (UAS) adalah wahana terbang nir-awak yang dalam satu dasawarsa terakhir ini berkembang kian pesat di ranah riset *unmanned system* (sistem nir-awak) di dunia. Bukan hanya mereka yang berada di ranah departemen pertahanan atau badan-badan riset, termasuk di perguruan tinggi, yang meneliti, mengkaji dan mengembangkan, namun dunia industri dan bidang sipil pun telah mulai banyak memanfaatkan teknologi UAV dalam mendukung kegiatan keseharian mereka (LAPAN, 2016). Contoh penggunaan UAV untuk pencarian korban bencana pada kondisi ekstrim, penginderaan jarak jauh seperti sistem *monitoring* serta bermanfaat sebagai alat pemetaan dan pengawasan pada suatu wilayah (Sirajuddin, 2013). Salah satu jenis UAV yang banyak dikembangkan dan dimanfaatkan untuk mempermudah aktivitas manusia dalam kehidupan sehari-hari adalah *quadcopter*.

Quadcopter merupakan salah satu jenis pesawat tanpa awak atau UAV *(Unmanned Aerial Vehicle)* yang memiliki empat buah baling-baling *(propeller)* dan empat buah motor *brushless* yang berfungsi sebagai penggerak (*actuator*) yang dikendalikan oleh *flight controller* agar robot dapat terbang dengan stabil. (Swamardika, IBA. 2014). Dalam melakukan misi tertentu menggunakan *quadcopter*, agar suatu misi dapat berjalan dengan baik, diperlukan sistem pengawasan atau *monitoring* terhadap pergerakan dan performa *quadcopter* untuk mengurangi kesalahan-kesalahan yang memungkinkan dapat terjadi dilapangan. Sebagai contoh, ketika mengoperasikan *quadcopter* secara jarak jauh dan tidak

terpantau secara langsung oleh penginderaan manusia atau pengguna (*pilot*), hal ini pengguna sulit untuk menentukan dan memperkirakan navigasi (arah) *quadcopter* itu sendiri, serta contoh lain adalah untuk mengawasi performa kapasitas baterai pada *quadcopter*.

1.2 Identifikasi Masalah

Berdasarkan uraian latar belakang masalah, maka masalah dapat diidentifikasi sebagai berikut :

1. Diperlukan rancangan dan realisasi sistem *monitoring* navigasi *quadcopter* berbasis Arduino Nano.
2. Diperlukan rancangan dan realisasi antarmuka sistem *monitoring* navigasi *quadcopter* menggunakan *software Processing* IDE.
3. Mengolah data yang akan dikirim dari *quadcopter* dan diterima oleh komputer serta ditampilkan pada antarmuka yang telah dibuat.

1.3 Pembatasan Masalah

Penelitian rancang bangun sistem *monitoring* navigasi *quadcopter* dibatasi oleh sebagai berikut :

1. Perancangan dan realisasi sistem *monitoring* navigasi *quadcopter* berbasis Arduino Nano.
2. Hanya terdapat 10 parameter yang akan di-*monitoring*, yaitu: (1) tegangan; (2) arus; (3) sudut *yaw*; (4) sudut *pitch*; (5) sudut *roll*; (6) *altitude* (ketinggian); (7) RPM motor 1; (8) RPM motor 2; (9) RPM motor 3; dan (10) RPM motor 4.
3. Perancangan dan realisasi antarmuka *Ground Control Station* (GCS) sistem *monitoring* navigasi *quadcopter* menggunakan *software Processing* IDE.

1.4 Perumusan Masalah

Ditinjau dari latar belakang, identifikasi dan pembatasan masalah, maka dapat dirumuskan masalah sebagai berikut : Bagaimana cara merancang, merealisasikan dan menguji sebuah sistem *monitoring* navigasi *quadcopter* berbasis Arduino Nano menggunakan *software Processing* IDE?

1.5 Tujuan Penelitian

Tujuan penelitian ini adalah merancang, merealisasikan dan menguji sistem *monitoring* navigasi *quadcopter* untuk memantau pergerakan dan performa sistem *quadcopter* ketika terbang diudara.

1.6 Manfaat Penelitian

Berdasarkan topik penelitian yang dibahas, maka manfaat dari penelitian ini adalah :

1. Manfaat dalam lingkup pendidikan :
   a. Dapat menambah keterampilan dalam membuat karya tulis ilmiah.
   b. Dapat diterapkan pada mata kuliah pemrograman komputer.
   c. Dapat diterapkan pada mata kuliah teknik antarmuka.
2. Manfaat dalam lingkup teknologi :
   a. Dapat menambah wawasan dan keterampilan dibidang teknologi penerbangan, khususnya teknologi UAV.
   b. Dapat me-*monitoring* navigasi atau pergerakan *quadcopter* untuk penelitian lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1 Kerangka Teoritik

2.1.1 Definisi Sistem *Monitoring* Navigasi *Quadcopter*

2.1.1.1 Sistem

Sistem menurut KBBI *online* berarti perangkat unsur yang secara teratur saling berkaitan, sehingga membentuk suatu totalitas (KBBI *online*, 2017).

Sedangkan menurut wikipedia, sistem berasal dari bahasa latin (*systēma*) dan bahasa Yunani (*sustēma*) adalah seperangkat yang berinteraksi atau komponen yang saling bergantung membentuk keseluruhan yang kompleks atau rumit. Setiap sistem digambarkan oleh batas-batas ruang dan waktu, dikelilingi dan dipengaruhi oleh lingkungannya, dijelaskan oleh struktur dan tujuan dan dinyatakan dalam fungsinya (Wikipedia, 2017).

Dari dua teori di atas, dapat disimpulkan sistem adalah sekumpulan unsur-unsur atau komponen sistem yang saling bergantung dan secara teratur berkaitan membentuk satu kesatuan dengan menghasilkan fungsi baru yang lebih kompleks atau rumit.

2.1.1.2 *Monitoring*

Kata *monitoring* dalam bahasa Indonesia berarti memonitor atau memantau. Memonitor atau memantau menurut KBBI *online* adalah mengawasi, mengamati, atau mengecek dengan cermat, terutama untuk tujuan khusus (KBBI *online*, 2017).

Sedangkan menurut wikipedia, *Monitoring* (bahasa Indonesia : *pemantauan*) adalah pemantauan yang dapat dijelaskan sebagai kesadaran (*awareness*) tentang

apa yang ingin diketahui. *Monitoring* merupakan proses rutin pengumpulan data dan pengukuran kemajuan atas objektif program. Memantau perubahan, yang fokus pada proses dan keluaran (Wikipedia, 2017).

Dengan menggunakan teori di atas, dapat disimpulkan *monitoring* adalah mengawasi, mengamati atau mengecek secara rutin sesuatu objek yang ingin diketahui untuk memperoleh suatu informasi untuk tujuan tertentu.

2.1.1.3 Navigasi

Navigasi dalam KBBI *online* adalah tindakan menempatkan haluan kapal/pesawat atau arah terbang (KBBI *online*, 2017).

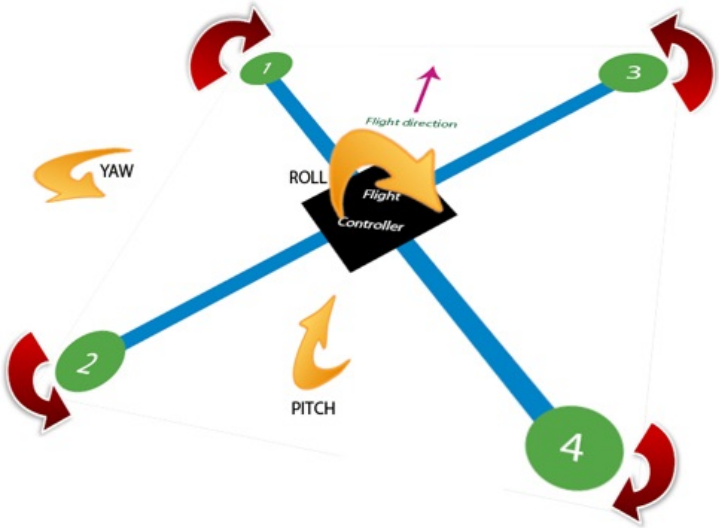
Sedangkan menurut wikipedia, navigasi atau pandu arah adalah penentuan kedudukan (posisi) dan atau arah perjalanan baik di medan sebenarnya atau di peta, dan oleh sebab itulah pengetahuan tentang pedoman arah (kompas) dan peta serta teknik penggunaannya haruslah dimiliki dan dipahami (Wikipedia, 2017).

Kesimpulan definisi navigasi adalah tindakan untuk menentukan posisi dan atau arah perjalanan, baik di medan sebenarnya ataupun di peta.

2.1.1.4 *Quadcopter*

Quadcopter merupakan pesawat *multirotor*/*multiwing*s yang memiliki rotor sejumlah 4 buah yang memiliki gerakan lebih leluasa dibandingkan dengan *helicopter* dengan 2 buah rotor. Terdapat empat gerakan dasar pada *quadcopter* yaitu gerakan *altitude* (*throuttle*), gerakan sudut (*roll, pitch),* dan gerakan sudut *yaw.* Gerakan *throttle* merupakan gerak translasi *quadcopter* sepanjang sumbu z, Gerakan ini dipengaruhi oleh perubahan kecepatan keempat rotor denngan nilai kecepatan putaran yang sama. Dengan pergerakan *throttle*, *quadcopter* akan dapat

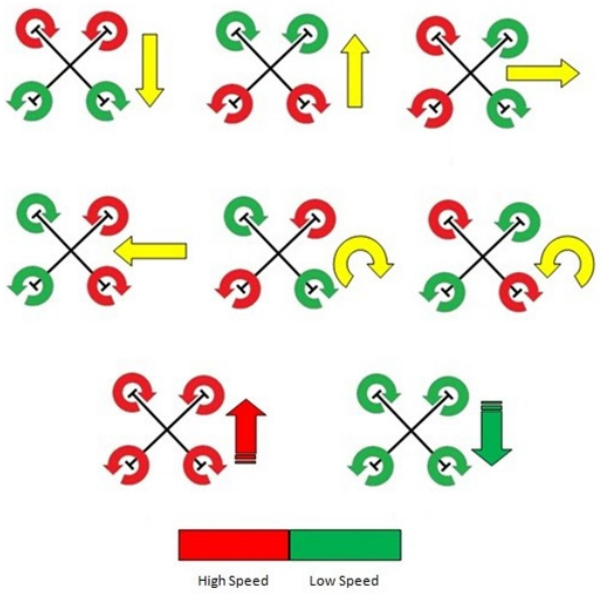
melakukan akselerasi naik atau turun sesuai dengan tingkat kecepatan rotor. Gerakan *roll* merupakan gerakan rotasi pada sumbu x, gerakan ini dipengaruhi oleh perubahan kecepatan rotor kanan dan kiri. Gerakan *pitch* merupakan gerakan rotasi pada sumbu y, gerakan ini dipengaruhi oleh perubahan kecepatan pada motor depan dan belakang, sedangkan *yaw* merupakan gerakan rotasi pada sumbu z (Arifin dkk., 2015:6). Dinamika gerak *quadcopter* ditunjukkan pada Gambar 2.1 dan Tabel 2.1 serta konsep navigasi *quadcopter* ditunjukkan pada Gambar 2.2.



Gambar 2.1 Dinamika Gerak *Quadcopter*
(Sumber: https://creativentechno.files.wordpress.com/2012/06/quadcopter_x_flight_dynamics_yaw_pitch_roll.jpg)

Tabel 2.1 Dinamika Gerak *Quadcopter*

Gerak	Motor 1	Motor 2	Motor 3	Motor 4
Pitch Up	Cepat	Pelan	Cepat	Pelan
Pitch Down	Pelan	Cepat	Pelan	Cepat
Roll Left	Pelan	Pelan	Cepat	Cepat
Roll Right	Cepat	Cepat	Pelan	Pelan
Yaw CW	Pelan	Cepat	Cepat	Pelan
Yaw CCW	Cepat	Pelan	Pelan	Cepat



Gambar 2.2 Navigasi *Quadcopter*
(Sumber: https://fahmizaleeits.wordpress.com/2013/02/15/dasar-dasar-quadcopter-dan-setup-kkboard-2-0-pada-quadcopter-mode-x/)

2.1.2 Definisi Arduino Nano

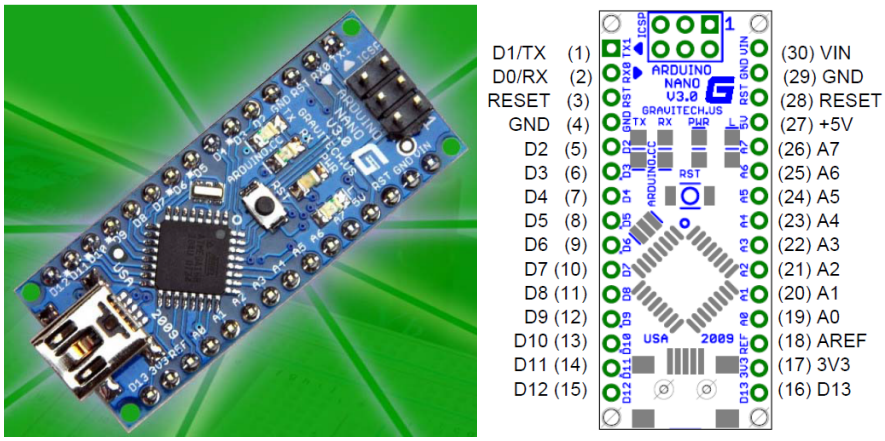
2.1.2.1 Arduino

Arduino adalah sebuah *platform* elektronik *opensource* yang terdiri dari mikrokontroler, bahasa pemrograman, dan IDE. Arduino adalah alat untuk membuat aplikasi interaktif, yang dirancang untuk mempermudah proyek bagi pemula tetapi masih cukup fleksibel bagi para ahli untuk mengembangkan proyek-proyek yang kompleks (Arduino, 2017).

Pemrograman arduino menggunakan bahasa processing. Processing merupakan bahasa pemrograman tingkat tinggi yang dialeknya sangat mirip dengan C++ dan Java, sehingga pengguna yang sudah terbiasa dengan kedua bahasa tersebut tidak akan menemui kesulitan dengan processing. Proyek arduino dapat berdiri sendiri atau dapat berkomunikasi dengan perangkat lunak yang sedang berjalan pada komputer (Banzi, 2008:1).

2.1.2.2 Arduino Nano

Arduino Nano adalah sebuah papan mikrokontroler dalam kemasan mini yang dapat diprogram ulang melalui USB yang terintegrasi dengan komputer. Jenis arduino ini tergolong kecil, lengkap, dan mudah digunakan. Arduino nano memiliki kurang lebih fungsi yang sama dengan diecimila / duemilanove. Secara fisik, arduino nano tidak memiliki *jack power* (jenis konektor sumber tegangan), namun untuk masukan/konsumsi sumber tegangan dapat melalui konektor USB mini atau pada pin 30 (V input, 6-20V) dan pin 4 atau 29 (*ground*). Arduino nano dapat dipasang pada *breadboard* (papan proyek) dan pada *socket* IC, sehingga dapat di pasang dan di lepas dengan mudah. Arduino nano didesain menggunakan USB mini dan tata letak komponen pada PCB dipasang secara *double layer* (dua lapis). Arduino nano banyak terdapat di pasaran dengan harga yang terjangkau (Arduino, 2017). Bentuk fisik dan pin-pin arduino nano ditunjukkan pada Gambar 2.3 dan Konfigurasi pin arduino nano dideskripsikan pada Tabel 2.2.



Gambar 2.3 Bentuk Fisik dan Pin-pin Arduino Nano
(Sumber: Gravitech, 2016)

Arduino nano versi 3.0 hadir dengan ATmega328 yang menawarkan lebih banyak pemrograman dan ruang memori data. Memori ATmega328 memiliki 32 KB, (juga dengan 2 KB digunakan untuk *bootloader*) serta memiliki 2 KB SRAM

dan 1 KB EEPROM. Pada I/O arduino nano ada 14 pin digital yang dapat digunakan sebagai *input* atau *output*. *Input* dan *output* beroperasi pada tegangan 5V yang mana setiap pin dapat memberi dan menerima arus maksimum 40 mA dan memiliki resistor *pull-up* internal (tidak *floating* atau mengambang) dari 20-50 Kohm.

Tabel 2.2 Konfigurasi Pin Arduino Nano

Pin No.	Name	Type	Description
1-2, 5-16	D0-D13	I/O	Digital input/output port 0 to 13
3, 28	RESET	Input	Reset (active low)
4, 29	GND	PWR	Supply ground
17	3V3	Output	+3.3V output (from FTDI)
18	AREF	Input	ADC reference
19-26	A7-A0	Input	Analog input channel 0 to 7
27	+5V	Output or Input	+5V output (from on-board regulator) or +5V (input from external power supply)
30	VIN	PWR	Supply voltage

Sumber: Gravitech, 2016

Arduino Nano memiliki sejumlah fasilitas untuk berkomunikasi dengan komputer, arduino lain, atau mikrokontroler lainnya. ATmega328 menyediakan UART TTL (5V) komunikasi serial, yang tersedia pada pin 0 (RX) dan 1 (TX). FTDI FT232RL pada saluran papan komunikasi serial ini melalui USB dan driver FTDI (termasuk dengan *software* arduino) menyediakan *virtual com port* untuk perangkat lunak pada komputer. ATmega328 juga mendukung I2C (TWI) dan komunikasi SPI. Perangkat lunak arduino termasuk *wire library* untuk menyederhanakan penggunaan bus I2C. Spesifikasi dan fitur dari mikrokontroler arduino nano versi 3.0 ditunjukkan pada Tabel 2.3 dan Tabel 2.4.

Tabel 2.3 Spesifikasi Arduino Nano

No	Nama spesifikasi	Keterangan Spesifikasi
1.	Mikrokontroler	Atmel ATmega 328
2.	Tegangan operasi (tingkat logika)	5 V
3.	Tegangan masukan (dianjurkan)	7 – 12 V
4.	Tegangan masukan (batas)	6 – 20 V
5.	Pin *digital* I/O	14 (6 pin sebagai keluaran PWM)
6.	Pin masukan *analog*	8
7.	Arus DC per pin I/O	40 mA
8.	*Flash Memory*	32 KB (2KB digunakan untuk *bootloader*)
9.	SRAM	2 KB
10.	EEPROM	1 KB
11.	Kecepatan *Clock* (detak)	16 MHz
12.	Dimensi	0,70” x 1,70”

Tabel 2.4 Fitur Arduino Nano

No	Fitur
1.	Secara otomatis me-*reset* program yang sebelumnya ketika men-*download* program baru
2.	*Power* bekerja ditandai dengan nyalanya LED biru
3.	LED hijau untuk TX, merah untuk RX dan oranye untuk L
4.	Sumber daya secara otomatis dipilih untuk sumber tegangan tertinggi
5.	USB kecil jenis mini-B untuk pemrograman dan *serial monitor*
6.	ICSP *header* untuk men-*download* program secara langsung
7.	Jarak spasi standar 0,1” DIP (sesuai *breadboard*)
8.	tombol *reset* manual

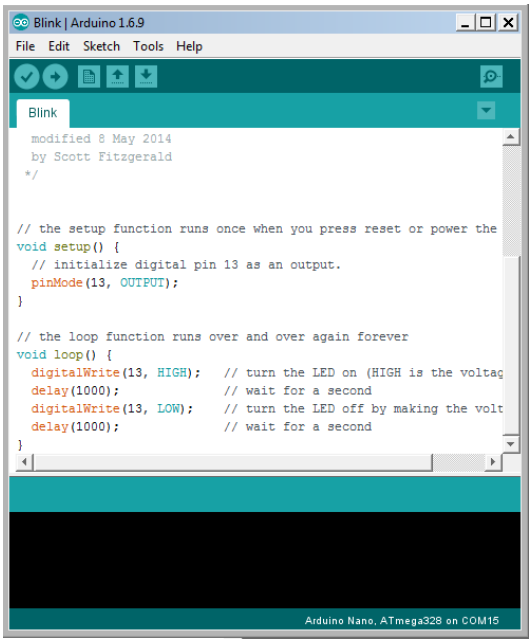
2.1.2.3 Arduino IDE

Arduino IDE (*Integrated Development Environment*) adalah perangkat lunak yang ditulis menggunakan bahasa java yang mencakup *editor* program, *compiler*, dan *uploader* (Djuandi, 2011). Berikut penjelasan singkat fitur utama dari perangkat lunak Arduino IDE :

1. *Editor* program, berfungsi untuk menulis dan mengedit program dalam bahasa processing.
2. *Compiler*, berfungsi untuk mengubah kode program dari bahasa processing menjadi kode biner.

3. *Uploader*, berfungsi untuk memuat kode biner dari komputer ke dalam memori yang ada pada papan arduino.

Editor sketch pada arduino IDE juga mendukung fungsi penomoran baris, *syntax highlighting*, yaitu pengecekan sintaksis kode *sketch*. Tampilan perangkat lunak arduino IDE ditunjukkan pada Gambar 2.4.



Gambar 2.4 *Software* Arduino IDE

2.1.3 Definisi *Software* Processing IDE

2.1.3.1 *Software*

Software (perangkat lunak) adalah bagian dari sistem komputer yang terdiri dari data sebagai instruksi komputer, berbeda dengan *hardware* (perangkat keras) yang mana sistem fisik yang dibangun. Dalam ilmu komputer dan rekayasa perangkat lunak, perangkat lunak komputer adalah semua informasi diproses oleh sistem komputer, program dan data. Perangkat lunak komputer meliputi program komputer, *libraries* (perpustakaan program) dan data *non-executable* terkait, seperti dokumentasi *online* atau media digital. *Hardware* komputer dan *software*

membutuhkan satu sama lain dan tidak dapat secara realistis digunakan sendiri (Wikipedia, 2017).

Sedangkan definisi *software* menurut teori yang dijelaskan oleh *website* softwaredetail.wordpress.com, *Software* adalah kumpulan instruksi yang berfungsi untuk menjalankan suatu perintah, seperti memberikan informasi tentang *hardware*, menentukan fungsi *hardware*, dan menjalankan sistem. Pengertian *software* menunjuk pada program dan alat bantu lain yang bersifat menambah kemampuan komputer sebagai alat untuk melaksanakan tugas atau operasi tertentu (Softwaredetail, 2017).

Berdasarkan kedua teori di atas, dapat disimpulkan *software* adalah bagian dari sistem komputer yang terdiri dari kumpulan data instruksi yang berfungsi untuk menjalankan suatu perintah, seperti memberikan informasi tentang *hardware*, menentukan fungsi *hardware*, dan menjalankan sistem.

2.1.3.2 Processing

Processing adalah bahasa pemrograman *open source* dan *Integrated Development Environment* (IDE) yang menghubungkan konsep perangkat lunak dengan prinsip bentuk visual, gerak/animasi, interaksi dan suara. Processing digunakan oleh mahasiswa, seniman, desainer, arsitek, peneliti, dan penggemar untuk belajar, membuat prototipe, dan produksi. Processing dibuat untuk mengajarkan dasar-dasar pemrograman komputer dalam konteks visual dan berfungsi sebagai buku sketsa perangkat lunak dan alat produksi untuk konteks tertentu (Reas & Fry, 2007: 1).

Bahasa processing adalah bahasa pemrograman teks yang dirancang khusus untuk menghasilkan dan memodifikasi gambar. Processing berusaha mencapai

keseimbangan antara kejernihan dan fitur canggih. Pemula dapat menulis program mereka sendiri hanya setelah beberapa menit pengajaran, namun pengguna yang lebih mahir dapat menggunakan dan menulis *library* dengan fungsi tambahan. Banyak teknik grafis dan interaksi komputer dapat didiskusikan, termasuk gambar vektor/raster, pengolahan gambar, model warna, *events mouse* dan *keyboard*, komunikasi jaringan, dan pemrograman berorientasi objek. *Library* dengan mudah memperluas kemampuan processing untuk menghasilkan suara, mengirim/menerima data dalam berbagai format, dan mengimpor/mengekspor format file 2D dan 3D (Reas & Fry, 2007: 1).

Kesimpulan processing adalah bahasa pemrograman *open source* dan *Integrated Development Environment* (IDE) yang digunakan untuk membuat aplikasi atau antarmuka berorientasi objek berupa gambar, gerak/animasi, suara dan dapat berinteraksi dengan perangkat mikrokontroler.

Proyek Processing dimulai pada tahun 2001 oleh Casey Reas dan Benjamin Fry, sementara keduanya adalah mahasiswa pascasarjana di MIT Media Lab dalam kelompok penelitian Komputasi dan Estetika John Maeda. Pada tahun 2012, mereka memulai Yayasan Processing bersama dengan Daniel Shiffman, yang bergabung sebagai kepala proyek ketiga. Salah satu tujuan dari processing adalah untuk memungkinkan *non-programmer* untuk memulai pemrograman komputer dibantu oleh umpan balik visual. Bahasa processing dibangun di atas bahasa java, tetapi menggunakan sintaks yang disederhanakan dan antarmuka pengguna grafis (Processing, 2017). Berikut adalah keunggulan menggunakan processing :

1. Gratis untuk *download* dan *open source*
2. Program interaktif dengan output 2D, 3D atau PDF

3. Integrasi OpenGL untuk akselerasi 2D dan 3D
4. Untuk GNU / Linux, Mac OS X, Windows, Android, dan ARM
5. Lebih dari 100 perpustakaan memperluas perangkat lunak inti
6. Didokumentasikan dengan baik, dengan banyak buku tersedia

2.1.3.3 IDE (*Integrated Development Environment*)

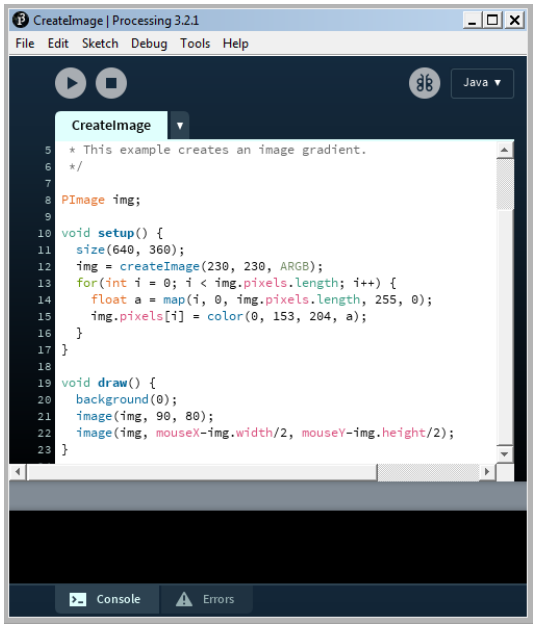
Lingkungan pengembangan terintegrasi (IDE) adalah aplikasi perangkat lunak yang menyediakan fasilitas yang lengkap untuk *programmer* komputer untuk pengembangan perangkat lunak. IDE biasanya terdiri dari *editor* kode sumber, membangun alat otomatisasi dan *debugger*. Kebanyakan IDE modern memiliki penyelesaian kode cerdas. Beberapa IDE, seperti NetBeans dan Eclipse, mengandung *compiler*, *interpreter*, atau kedua lainnya seperti SharpDevelop dan Lazarus. Batas antara lingkungan pengembangan terintegrasi dan bagian lain dari lingkungan pengembangan perangkat lunak yang lebih luas tidak didefinisikan dengan baik. Kadang-kadang sistem kontrol versi, atau berbagai alat untuk menyederhanakan pembangunan *Graphical User Interface* (GUI), yang terintegrasi. Banyak IDE modern juga memiliki *browser* kelas, *browser* objek, dan diagram hirarki kelas, untuk digunakan dalam pengembangan perangkat lunak berorientasi objek (Wikipedia, 2017).

2.1.3.4 Processing IDE

Processing Development Environment (PDE) atau IDE adalah sebuah perangkat lunak untuk membuat aplikasi yang terdiri dari editor teks sederhana untuk menulis kode, area pesan, konsol teks, *tab* untuk mengelola *file*, *toolbar* dengan tombol untuk tindakan umum, dan serangkaian menu. Ketika program

dijalankan, mereka membuka di jendela baru disebut *window display* (Reas & Fry, 2007: 9).

Buah perangkat lunak yang ditulis menggunakan processing disebut sketsa. sketsa ini ditulis dalam *editor* teks. Ini memiliki fitur untuk meng-*cut*/*paste* dan untuk mencari/mengganti teks. Daerah pesan memberikan umpan balik saat menyimpan dan mengekspor dan juga menampilkan *error* (kesalahan). Konsol menampilkan teks *output* dengan program processing termasuk pesan *error* lengkap dan *output* teks dari program dengan fungsi print() dan println(). Tombol *toolbar* memungkinkan Anda untuk menjalankan dan menghentikan program, membuat sketsa baru, membuka, menyimpan, dan ekspor (Reas & Fry, 2007: 9). Tampilan *software* processing IDE ditunjukkan pada Gambar 2.5.



Gambar 2.5 *Software* Processing IDE

2.1.4 Definisi Sistem *Monitoring* Navigasi *Quadcopter* berbasis Arduino Nano menggunakan *software* Processing IDE

Berdasarkan penjelasan definisi tentang sistem *monitoring* navigasi *quadcopter* yang telah dibahas sebelumnya, dapat dikatakan bahwa sistem

monitoring navigasi *quadcopter* dalam penelitian ini adalah sebuah sistem yang digunakan untuk memantau tingkah laku atau pergerakan *quadcopter* ketika terbang diudara. Sistem *monitoring* navigasi *quadcopter* yang akan direalisasikan menggunakan beberapa sensor, diantaranya adalah akseleromoter, giroskop, magnetometer, barometer, sensor tegangan dan arus baterai dan 4 buah sensor RPM motor. Kemudian menggunakan 2 mikrokontroler arduino nano, yaitu arduino nano sebagai *transmitter* (arduino pemancar) yang dipasang atau dihubungkan pada *quadcopter* untuk memproses data yang dihasilkan dari sensor-sensor dan arduino nano sebagai *receiver* (arduino penerima) yang dihubungkan ke komputer dan processing IDE untuk menerima data yang dikirim oleh sistem pemancar pada *quadcopter*. Dan selanjutnya sistem *monitoring* navigasi *quadcopter* menggunakan *software* processing IDE sebagai penampil hasil pengukuran parameter atau antarmuka pergerakan *quadcopter* yang disajikan dalam bentuk data visual 2 dimensi dan 3 dimensi agar dapat dipahami oleh pengguna.

2.1.5 Spesifikasi tiap Parameter *Quadcopter* yang akan di-*monitoring*

Terdapat 10 (sepuluh) spesifikasi parameter *quadcopter* yang akan di-*monitoring* dalam penelitian ini yang memiliki kriteria sebagai berikut :

1) Tegangan baterai : merupakan parameter yang akan diubah menjadi kapasitas baterai *quadcopter* yang dinyatakan dalam satuan persen dengan spesifikasi antara 0% hingga 100% dan tingkat ketelitian 1%.
2) Arus baterai : merupakan parameter yang dinyatakan dalam satuan ampere dengan spesifikasi antara 0A hingga kurang lebih 15A.
3) *Yaw* / *Heading* (geleng) : merupakan parameter arah hadap *quadcopter* ketika berputar pada porosnya ke kiri dan ke kanan yang dinyatakan dalam satuan

derajat dengan spesifikasi antara 0 derajat hingga 360 derajat dan tingkat ketelitian 1 derajat.

4) *Pitch* (angguk) : merupakan parameter kemiringan *quadcopter* ketika bergerak ke belakang dan ke depan yang dinyatakan dalam satuan derajat dengan spesifikasi antara -90 derajat hingga 90 derajat dan tingkat ketelitian 1 derajat.
5) *Roll* (putar) : merupakan parameter kemiringan *quadcopter* ketika bergerak ke kiri dan ke kanan yang dinyatakan dalam satuan derajat dengan spesifikasi antara -180 derajat hingga 180 derajat dan tingkat ketelitian 1 derajat.
6) *Altimeter* (ketinggian) : merupakan parameter ketinggian *quadcopter* yang dinyatakan dalam satuan meter dari permukaan laut (mdpl) dengan spesifikasi antara 0 mdpl hingga 50 mdpl dan tingkat ketelitian 1 mdpl.
7) *RPM Meter* 1 : merupakan parameter putaran motor *quadcopter* pada motor *brushless* 1 yang dinyatakan dalam satuan rotasi per menit (rpm) dengan spesifikasi antara 0 rpm hingga 13500 rpm dan tingkat ketelitian 300 rpm.
8) *RPM Meter* 2 : merupakan parameter putaran motor *quadcopter* pada motor *brushless* 2 yang dinyatakan dalam satuan rotasi per menit (rpm) dengan spesifikasi antara 0 rpm hingga 13500 rpm dan tingkat ketelitian 300 rpm.
9) *RPM Meter* 3 : merupakan parameter putaran motor *quadcopter* pada motor *brushless* 3 yang dinyatakan dalam satuan rotasi per menit (rpm) dengan spesifikasi antara 0 rpm hingga 13500 rpm dan tingkat ketelitian 300 rpm.
10) *RPM Meter* 4 : merupakan parameter putaran motor *quadcopter* pada motor *brushless* 4 yang dinyatakan dalam satuan rotasi per menit (rpm) dengan spesifikasi antara 0 rpm hingga 13500 rpm dan tingkat ketelitian 300 rpm.

2.1.6 Ardupilot APM 2.8

Ardupilot adalah proyek *autopilot* berbasis *platform open source* Arduino. Ardupilot Mega (APM) adalah produk yang dikembangkan oleh Chris Anderson dan Jordi Munoz dari *DIY Drones*. Modul APM berbasis *open source* paling berkembang untuk modul *autopilot*, baik *autopilot* untuk pesawat (Arduplane), *Multicopter* (Arducopter) dan juga kendaraan darat (Ardurover). Mudul ini menggunakan mikrokontroler Arduino yang sangat populer di bidang instrumentasi. Perangkat keras APM akan tetap sama untuk semua tipe pesawat dan dibedakan berdasarkan *firmware* yang ditanamkan sesuai dengan tipe pesawat yang digunakan menggunakan via Mission Planner. Untuk meningkatkan performa pesawat secara spesifik, dapat ditambahkan sensor lain seperti sensor *airspeed* atau magnetometer untuk *multicopter* (Arifin dkk., 2015: 9-10).

Ardupilot yang digunakan dalam penelitian ini adalah Ardupilot APM 2.8 yang merupakan versi terbaru. Sensor yang persis sama dengan Ardupilot APM 2.6 namun ini memiliki opsi untuk menggunakan kompas internal yang sudah ada pada modul, atau kompas eksternal melalui *jumper*. Hal ini membuat APM 2.8 ideal untuk digunakan dengan *multicopter* dan *rover*. APM 2.8 adalah sistem *autopilot open source* yang lengkap dan teknologi laris yang memenangkan kompetisi *UAV Outback Challenge* bergengsi. Kendaraan pesawat atau *multirotor* (bahkan mobil dan perahu) menjadi kendaraan sepenuhnya otomatis, mampu melakukan misi GPS diprogram dengan *waypoint*. Pada revisi ini, kompas ditempatkan jauh dari pengaruh gangguan magnetik motor. APM 2.8 ini dirancang untuk digunakan dengan 3DR GPS uBlox dengan kompas, sehingga GPS / kompas dapat dipasang

jauh dari sumber-sumber kebisingan dari APM sendiri (*Unmannedtechshop*, 2017). Bentuk fisik Ardupilot APM 2.8 ditunjukkan pada Gambar 2.6.

Gambar 2.6 Ardupilot APM 2.8
(Sumber: http://i.ebayimg.com/images/i/231982434726-0-1/s-l1000.jpg)

2.1.7 *Frame* F450Q

Frame F450Q (*Frame quad* 450) adalah jenis *frame multicopter* yang memiliki 4 lengan dan berukuran diameter 450 mm (Hardiansyah, 2014). *Frame* F450Q terbuat dari bahan plastik dan *fiberglass* yang terintegrasi dengan PDB (*Power Distribution Battery*). *Frame* F450Q sangat terjangkau, kuat dan cukup lentur, sehingga dapat mengurangi getaran atau goncangan pada saat *landing* atau terjadi *crash*. Selain itu, *frame* F450Q memiliki *landing gear* kecil pada bagian bawah masing-masing ujung lengan agar perangkat lain tidak terkena kontak langsung ke landasan. Hal tersebut berguna untuk melindungi *flight controller* agar tidak terkena benturan ke tanah jika terjadi *crash*. *Frame* F450Q memiliki berat sekitar 282g dan cocok untuk *propeller* berukuran 8x45 atau 10x45 (*Unmannedtechshop*, 2017). Bentuk dan spesifikasi *frame* F450Q *quadcopter* ditunjukkan pada Gambar 2.7 dan Tabel 2.5.

Gambar 2.7 *Frame* F450 *Quadcopter*
(Sumber : http://www.radioc.co.uk/v/vspfiles/photos/1000-2.jpg)

Tabel 2.5 Spesifikasi F450 *Quadcopter*

Parameter	Nilai
Frame Weight	282 g
Diagonal Wheelbase	450 mm
Takeoff Weight	800 g – 1200 g

2.1.8 Motor *Brushless*

Motor *Brushless* (BLDC) merupakan motor listrik *synchronous* AC (*Alternating Current*) 3 fasa. Walaupun merupakan motor listrik *synchronous* AC 3 fasa, motor ini tetap disebut dengan motor BLDC karena pada implementasinya motor BLDC menggunakan sumber tegangan DC sebagai sumber energi utama yang kemudian diubah menjadi tegangan AC dengan menggunakan *inverter* 3 fasa. Tujuan dari pemberian tegangan AC 3 fasa pada *stator* BLDC adalah menciptakan medan magnet putar *stator* untuk menarik magnet rotor (Dharmawan, 2009).

Motor BLDC secara umum terdiri dari *stator*, *rotor* dan komutator. *Stator* adalah bagian motor yang diam yang terpasang kumparan 3 fasa, sedangkan *rotor* adalah bagian motor yang berputar yang terpasang magnet neodinium serta komutator elektronik adalah sensor *hall* atau *encoder* yang berfungsi untuk menentukan *timing* komutasi yang tepat, sehingga didapatkan torsi dan kecepatan

dimana :

P = tekanan (Pa)

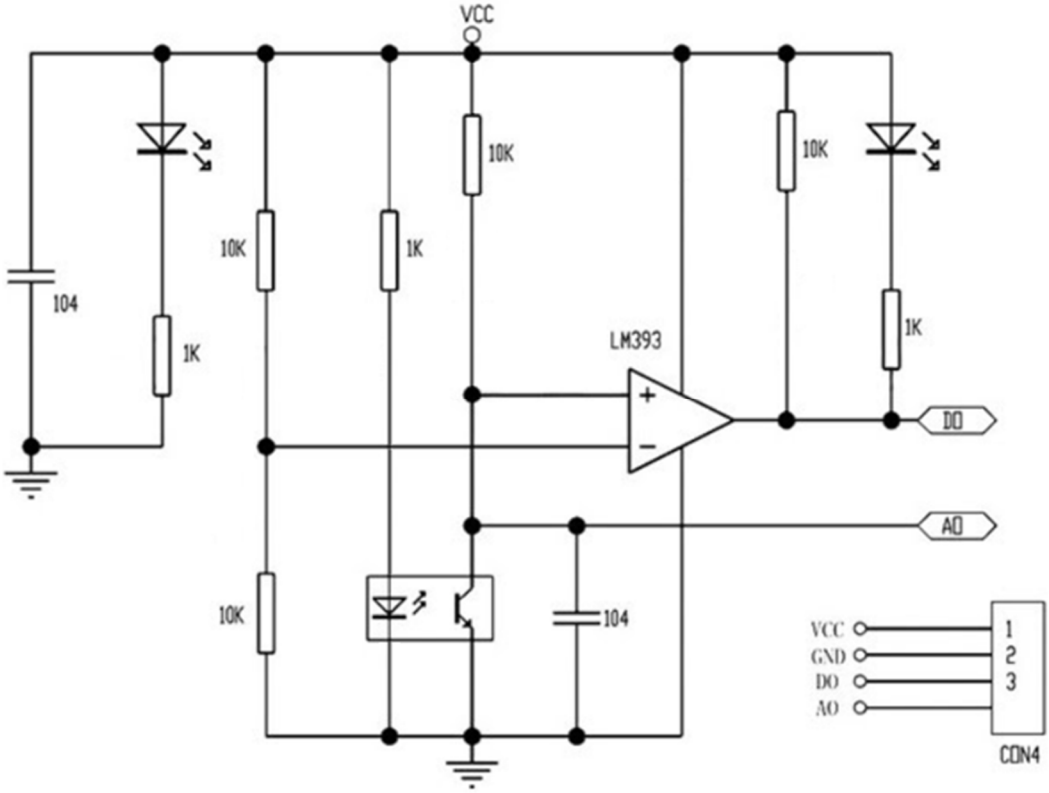
p0 = *sea level standard atmospheric pressure* (101325 Pa)

altitude = nilai ketinggian (mdpl).

2.1.13 Modul LM393 *Infrared Speed Sensor*

Speed sensor atau sensor kecepatan merupakan suatu sensor yang digunakan untuk mendeteksi kecepatan gerak benda untuk selanjutnya diubah kedalam bentuk sinyal elektrik (Zuroya, 2017). Dalam penelitian ini, *Speed sensor* yang diterapkan untuk mengukur RPM (*Rotation per Minutes*) motor *brushless* (BLDC) adalah Modul LM393 *Infrared Speed Sensor*. *Speed sensor* yang dibutuhkan berjumlah 4 buah dipasang pada tiap-tiap motor, agar dapat diketahui perbedaan banyaknya rotasi pada tiap-tiap motor ketika *quadcopter* bernavigasi *yaw*, *pitch* dan *roll*. Modul dan skema rangkaian *LM393 infrared speed sensor* yang dapat digunakan sebagai RPM meter ditunjukkan pada Gambar 2.14 dan 2.15.

Gambar 2.14 Modul *LM393 Infrared Speed Sensor*
(https://www.tokopedia.com)



Gambar 2.15 Skema Rangkaian *LM393 Infrared Speed Sensor*
(https://forum.arduino.cc)

2.1.14 3DR *Power Module*

3DR *Power Module* merupakan cara sederhana untuk menyediakan papan kontrol pengguna dengan data pengukuran tegangan dan konsumsi arus dari baterai lipo sehingga papan kontrol dapat memperkirakan lama waktu penerbangan yang tersisa sesuai dengan kapasitas daya dan konsumsi arus *real time* dihitung dengan data yang diterima. 3DR *Power Modul* ini memiliki fungsi handal yang mana dapat memberikan catu daya yang stabil ke papan kontrol (Geeetech, 2015).

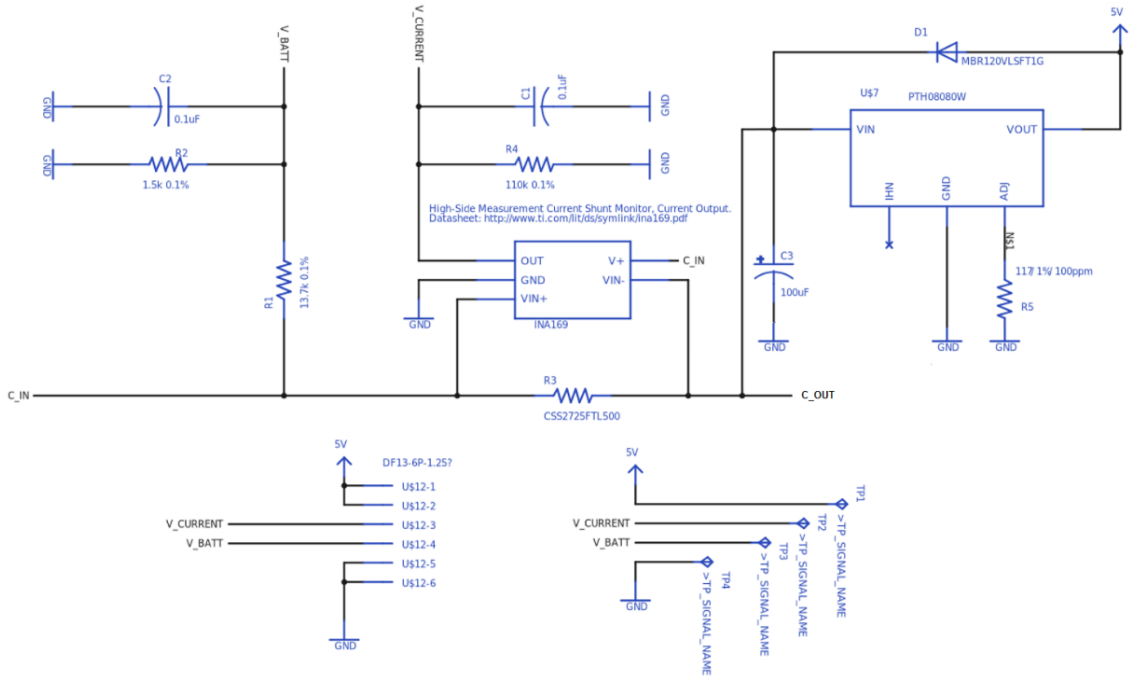
3DR *Power Module* dapat beroperasi pada rentang tegangan 4.5 Volt hingga 18 Volt dan dapat mengukur tegangan hingga 30 Volt DC dan mengukur arus hingga 90 Ampere serta konfigurasi pengukuran sebesar 5 Volt ADC pada pin keluaran arus dan tegangan. Pada penelitian ini, 3DR *Power Module* dikontrol bukan menggunakan Ardupilot APM, melainkan dikontrol menggunakan Arduino

Nano. Modul dan spesifikasi 3DR *Power Module* dapat dilihat pada Gambar 2.16 dan Tabel 2.6 serta skema rangkaiannya dapat dilihat pada Gambar 2.17 :

Gambar 2.16 3DR *Power Module*
(www.geetech.com)

Tabel 2.6 Spesifikasi 3DR *Power Module*

No.	Nama Spesifikasi	Keterangan Spesifikasi
1.	Tegangan input	4,5 – 18 Volt
2.	Maks. pengukuran tegangan	30 Volt
3.	Maks. pengukuran arus	90 Ampere
4.	Konfigurasi pengukuran	5 Volt ADC
5.	Dimensi	23mm*22mm*10mm
6.	Berat	19,2g
7.	Konektor kabel	6-pos DF13



Gambar 2.17 Skema Rangkaian 3DR *Power Module*
(https://upverter.com)

2.1.15 Telemetri

Telemetri adalah pengukuran besaran tertentu secara jarak jauh (LAPAN, 2016). Sedangkan pengertian lainnya, Telemetri (sejenis dengan telematika) adalah sebuah teknologi yang memungkinkan pengukuran jarak jauh dan pelaporan informasi kepada perancang atau operator sistem. Kata telemetri berasal dari akar bahasa Yunani *tele* yang berarti jarak jauh, dan *metron* yang berarti pengukuran. Telemetri merujuk pada komunikasi nirkabel (komunikasi tanpa kabel atau melalui udara) contohnya menggunakan sistem radio untuk mengimplementasikan hubungan data, tetapi juga merujuk pada data yang dikirimkan melalui media lain, seperti telepon, jaringan komputer, melalui kabel serat optik atau ketika membuat robot kita dapat menggunakan satu kabel. Secara umum sistem telemetri terdiri atas enam bagian pendukung, yaitu objek ukur, sensor, pemancar, saluran transmisi, penerima dan *display* (Wikipedia, 2017).

Telemetri adalah proses dimana karakteristik obyek diukur (seperti kecepatan pesawat terbang), dan hasilnya dikirimkan ke stasiun yang jauh di mana hasil pengukuran ditampilkan, dicatat, dan dianalisis. Standar internasional yang ditetapkan untuk peralatan dan piranti lunak telemetri adalah CCSDS (*Consultative Comittee for Standard Data Services*) dan IRIG (*The Standard for Digital Flight Data Recording)*. Proses telemetri melibatkan pengukuran pengelompokan (seperti tekanan, kecepatan, dan temperatur) ke dalam format yang dapat ditransmisikan sebagai aliran data tunggal. Setelah diterima, aliran data dipisahkan menjadi komponen pengukuran asli untuk analisis. Telemetri memungkinkan pengguna tetap di lokasi yang aman (atau nyaman) saat memantau apa yang terjadi di lokasi yang tidak aman (atau tidak nyaman).

2.1.16 *Radio Control*

Radio 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. Frekuensi yang digunakan sebagai media transmisi adalah gelombang radio pada frekuensi 2,4 GHz. Pada kondisi *outdoor* penggunaan frekuensi jika dibandingkan dengan menggunakan sinyal *infrared* sinar matahari sering memberikan gangguan terhadap sinyal *infrared* yang sangat mempengaruhi proses kendali, maka gelombang radio merupakan pilihan tepat. Selain itu, penggunaan gelombang radio mempunyai keunggulan dimana data yang dipancarkan dapat dikirim pada jarak yang cukup jauh bahkan dapat menembus halangan (Sirajuddin, 2013). Bentuk fisik *radio control* dapat dilihat pada Gambar 2.18.

Gambar 2.18 *Radio Control*
(Sumber: https://pixhawk.org/_media/modules/peripherals/futaba_t7c.png?w=400&tok=d78678)

2.1.17 XBee *Pro* S1

Modul XBee *Pro* S1 merupakan perangkat telemetri nirkabel yang beroperasi dalam ISM 2,4 GHz dan dapat berkomunikasi dua arah (mengirim dan menerima data). XBee Pro S1 dapat berkomunikasi di dalam ruangan (*Indoor*) sejauh kisaran 90 meter dan di luar ruangan (*RF line-of-sight Range*) sejauh 1600 meter. Perangkat nirkabel XBee ini dapat langsung dihubungkan ke *port serial* (pada tegangan 3.3 Volt) dari mikrokontroler. Dengan menggunakan penerjemah tingkat logika juga dapat dihubungkan ke perangkat logika 5 Volt (TTL) yang memiliki *interface serial*. Modul XBee biasanya diaplikasikan untuk pengendalian dan pemantauan mesin di Industri, pengendalian dan pengambilan data (*data logger*) pada Turbin Angin dari jarak jauh, pengambilan data parameter udara di tempat yang berbahaya, pengambilan gambar diudara menggunakan *drone*, penggunaan pada pesawat *Aeromodelling*, penerapan pada robot berbasis jaringan *wireless* dan banyak aplikasi lainnya. Modul XBee mendukung kecepatan data hingga 250 kbps. Modul XBee seri 1 dan Seri 2 memiliki *pin-out* yang sama. Namun, modul seri 1 tidak dapat berkomunikasi dengan modul seri 2 (Digi, 2012). Modul XBee dapat dilihat

pada Gambar 2.19 dan blok sistem pemancar dan penerima XBee ditunjukkan pada Gambar 2.20. Sedangkan untuk spesifikasi, fitur dan konfigurasi pin XBee dijelaskan pada Tabel 2.7, 2.8 dan 2.9.

Gambar 2.19 Modul XBee *Pro* S1
(www.digi.com)

Tabel 2.7 Spesifikasi Modul XBee *Pro* S1

No	Spesifikasi
1.	Frekuensi operasi: ISM 2,4 GHz
2.	Jenis antena: Antena Kawat
3.	*Indoor* / jangkauan hingga 300 ft. (90 m)
4.	Luar ruangan RF *Line-of-Sight* (LoS) jangkauan hingga 1 *miles* (1600 m)
5.	Antarmuka: *Serial* (UART) pada 1200 bps - 250 kbps
6.	Tegangan kerja: 2.8 - 3.4V
7.	Arus Pengiriman 250 mA (@ 3,3 V)
8.	Arus Penerimaan 55 mA (@ 3,3 V)
9.	Dimensi: 0,960 "x 1,297" (2.438 cm x 3.294 cm)
10.	Suhu operasi: -40 sampai 85º C (industri)

Tabel 2.8 Fitur Modul XBee *Pro* S1

No	Fitur
1.	Jumlah Kanal: (*software* dipilih) 12 Urutan *channel* Langsung
2.	Mengatasi Pilihan: PAN ID, *channel* dan Alamat
3.	AT *Command* & API *operation Mode*
4.	Transfer Daya: 63 mw (18 dBm)
5.	Sensitivitas Penerima: 100 dBm

Tabel 2.9 Konfigurasi Pin XBee

No.	Name	Direction	Description
1	VCC	-	Power Supply
2	DOUT	Output	UART data out
3	DIN/ $\overline{CONFIG}$	Input	UART data in
4	DO8[1]	Either	Digital output 8
5	$\overline{RESET}$	Input/Open drain output	Device reset (reset pulse must be at least 200 ns). This must be driven as an open drain/collector. The device drives this line low when a reset occurs. Never drive this line high.
6	PWM0/RSSI	Either	PWM output 0 / RX signal strength indicator
7	PWM1	Either	PWM output 1
8	[reserved]	-	Do not connect
9	$\overline{DTR}$/SLEEP_RQ/DI8	Either	Pin sleep control line or digital input 8
10	GND	-	Ground
11	AD4/DIO4	Either	Analog input 4 or digital I/O 4
12	$\overline{CTS}$/DIO7	Either	Clear to send flow control or digital I/O 7
13	ON/$\overline{SLEEP}$	Output	Device status indicator
14	VREF	Input	Voltage reference for A/D inputs
15	Associate/AD5/DIO5	Either	Associated indicator, analog input 5 or digital I/O5
16	$\overline{RTS}$/DIO6	Either	Request to send flow control, or digital I/O 6
17	AD3/DIO3	Either	Analog input 3 or digital I/O 3
18	AD2/DIO2	Either	Analog input 2 or digital I/O 2
19	AD1/DIO1	Either	Analog input 1 or digital I/O 1
20	AD0/DIO0	Either	Analog input 0 or digital I/O 0

(Sumber: Digi, 2017:14-15)

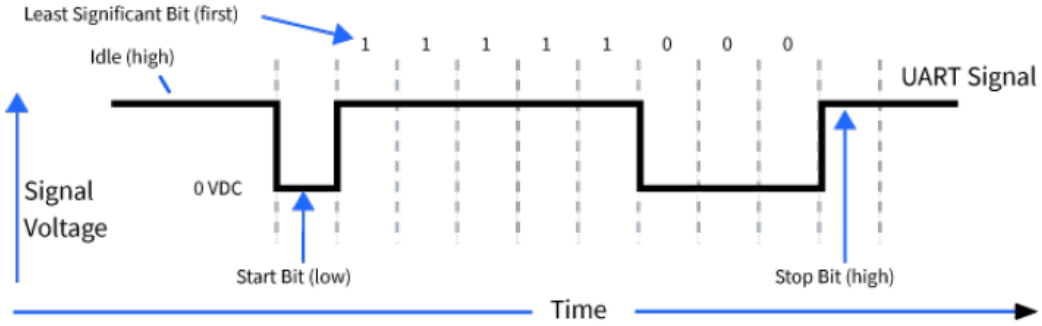
Desain Catatan:

1. Koneksi Minimum: VCC, GND, DOUT & DIN
2. Koneksi minimum untuk memperbarui *firmware*: VCC, GND, DIN, DOUT, RTS & DTR
3. Sinyal *Direction* ditentukan sehubungan dengan modul
4. Modul meliputi 50k Ω *pull-up* resistor pada *RESET*
5. Beberapa masukan *pull-up* dapat dikonfigurasi menggunakan perintah PR
6. Pin yang tidak digunakan harus dibiarkan terputus.

Gambar 2.20 Blok Sistem Pemancar dan Penerima XBee
(Sumber: Digi, 2017:22)

Serial Data:

Data memasuki modul UART melalui pin DI (pin 3) sebagai sinyal serial *asynchronous*. Sinyal harus siaga tinggi ketika ada data yang sedang dikirim. Setiap *byte* data yang terdiri dari *start* bit (rendah), 8 bit data (setidaknya bit signifikan pertama) dan *stop* bit (tinggi). Gambar 2.21 mengilustrasikan pola bit serial data yang melewati perangkat dan menunjukkan paket data UART 0x1F (angka desimal 31) seperti yang ditransmisikan melalui perangkat.



Gambar 2.21 Ilustrasi Pola Bit Serial Data XBee

Komunikasi serial tergantung pada dua UART (mikrokontroler dan modul RF) untuk dikonfigurasi dengan pengaturan yang kompatibel (*baud rate*, paritas, *start* bit, *stop* bit, data bit). UART *baud rate* dan paritas pengaturan pada modul *XBee* dapat dikonfigurasi dengan perintah BD dan SB masing-masing.

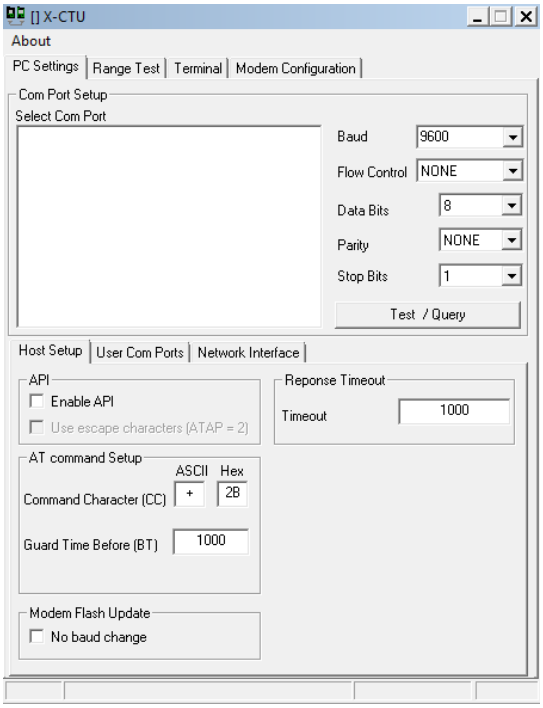
Keunggulan X Bee:

1. Kemampuan mengirim data secara jarak jauh
2. Konsumsi daya yang rendah
3. Dapat mendukung ADC dan jalur I/O
4. Mudah digunakan.

2.1.18 XBee *Adapter*

Xbee *adapter* adalah sebuah *shield* yang menghubungkan modul XBee ke komputer melalui Port USB (Gravitech, 2008). *XBee adapter board* menggunakan IC FT232RL. Untuk menghubungkan *XBee adapter board* ke komputer, diperlukan sebuah *driver*, yaitu driver FTDI yang sudah terinstal di komputer. Selain itu, XBee *adapter* juga dapat berfungsi untuk menyetting alamat frekuensi antar XBee (*transmitter – receiver*) menggunakan *software* X-CTU. Tujuannya adalah untuk menghindari terjadinya interferensi dengan frekuensi perangkat lain jika terjadi kesaman frekuensi. Modul XBee *adapter board* dan *software* X-CTU ditunjukkan pada Gambar 2.22 dan 2.23.

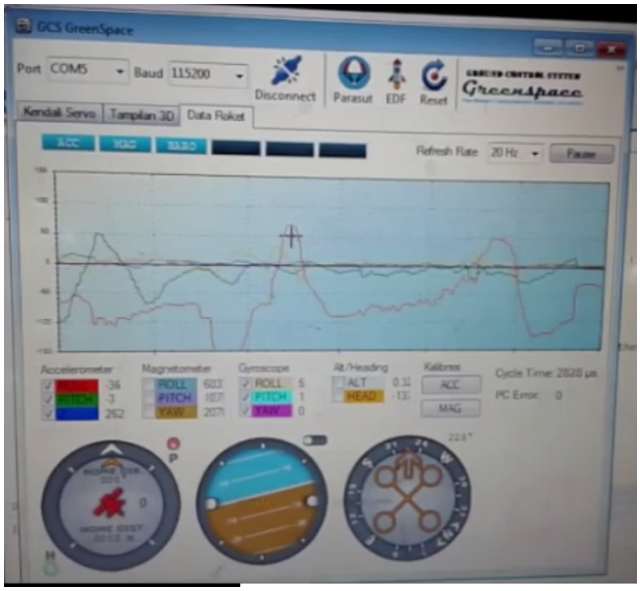
Gambar 2.22 XBee *Adapter Board*
(https://www.tokopedia.com/endashoper-tokop/arduino-xbeebluetooth-bee-adapter-with-data-cable)



Gambar 2.23 *Software* X-CTU

2.1.19 *Ground Control Station*

Ground Control Station merupakan alat pantau / stasiun pengendali untuk mengontrol kendali armada (*vehicle* yang akan di kendalikan). Baik itu *vehicle* darat, laut dan udara (rovindonesia, 2017). GCS dalam penelitian ini dibuat untuk mengetahui pergerakan *quadcopter* melalui komputer (*laptop*), baik arah, ketinggian, keseimbangan, posisi *quadcopter* dan lain-lain. GCS dalam penelitian ini terdiri dari perangkat keras dan perangkat lunak. Perangkat keras yaitu terdiri dari *radio control*, *laptop*, arduino nano, XBee, dan XBee *adapter board*. Sedangkan perangkat lunaknya adalah antarmuka (tampilan 2D dan 3D) yang telah dibuat menggunakan *software* Processing IDE. Contoh GCS ditunjukkan pada Gambar 2.24.



Gambar 2.24 Contoh Tampilan GCS pada *Laptop*

2.2 Konsep Pengembangan Produk

2.2.1 Pengertian Pengembangan Produk

Pengembangan produk adalah merupakan penelitian terhadap produk yang sudah ada untuk dikembangkan lebih lanjut agar mempunyai tingkat kegunaan yang lebih tinggi dan lebih disukai konsumen. Penelitian ini dapat bersifat penelitian lapangan (survey konsumen) serta dapat pula sebagai penelitian laboratoris (di dalam laboratorium perusahaan) atau dapat pula kedua-duanya. Di dalam penelitian lapangan akan dicari data-data mengenai produk yang akan dikembangkan. Pengembangan disini dapat meliputi pengembangan kualitasnya, kegunaannya, dan sebagainya, sesuai dengan selera konsumen. Sedangkan penelitian laboratorium menyangkut masalah penerapan pengembangan produk tersebut, terutama untuk produk-produk obat-obatan dan sebagainya. Adanya penelitian dan pengembangan produk ini diharapkan perusahaan selalu dapat menyesuaikan diri dengan produk produk yang disenangi konsumen (Nasution, 2003).

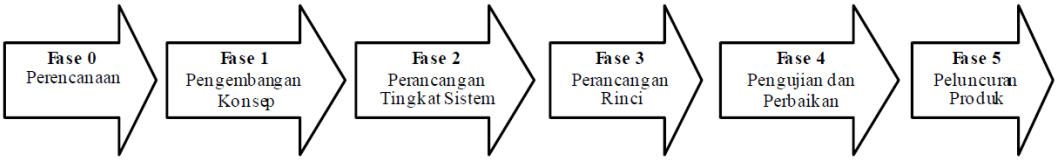
Tujuan dari penelitian dan pengembangan produk adalah agar barang atau jasa yang dihasilkan selalu sesuai dengan kebutuhan dan perkembangan selera masyarakat. Dengan demikian barang yang dihasilkan akan selalu dapat diminati dan dibutuhkan masyarakat. Tujuannya adalah agar barang atau jasa yang dihasilkan dapat selalu meningkat penjualannya, sehingga keuntungan perusahaan dapat selalu berkembang dan meningkat. Keuntungan yang meningkat akan dapat menjamin kelangsungan hidup perusahaan (Nasution, 2003).

2.2.2 Langkah-langkah Penelitian dan Pengembangan Produk

Sebelum melakukan penelitian pengembangan, peneliti terlebih dahulu menyusun langkah-langkah penelitian dan pengembangan produk yang akan dilakukan. Dalam menyusun langkah-langkah tersebut, peneliti melakukan studi literasi terkait teori model penelitian dan pengembangan produk menurut beberapa para ahli yang kemudian disintesiskan menjadi model penelitian baru yang akan digunakan dalam penelitian ini. Terdapat 3 (tiga) teori model peneltian dan pengembangan yang dibahas, diantaranya adalah (1) model Karl T. Ulrich dan Steven D Eppinger, (2) model Borg dan Gall, dan (3) model Sugiyono. Model penelitian dan pengembangan menurut para ahli dijelaskan sebagai berikut.

2.2.2.1 Model Karl T. Ulrich dan Steven D. Eppinger

Proses pengembangan produk menurut Karl T. Ulrich dan Steven D. Eppinger dalam buku mereka yang berjudul "*Perancangan dan Pengembangan Produk*" (2001: 15-17) ada 6 fase dalam proses pengembangan produk, yaitu ditunjukkan pada Gambar 2.25.



Gambar 2.25 Langkah-langkah Penelitian dan Pengembangan Produk menurut Ulrich dan Eppinger
(Sumber : Perancangan dan Pengembangan Produk, Ulrich-Eppinger)

1. **Fase 0 Perencanaan Produk** : Kegiatan perencanaan disebut sebagai "*zero fase*" karena kegiatan ini mendahului persetujuan proyek dan proses penelusuran pengembangan produk aktual yang akan dikembangkan.
2. **Fase 1 Pengembangan Konsep** : Pada fase pengembangan konsep, kebutuhan pasar target diidentifikasi, alternatif konsep-konsep produk dibangkitkan dan dievaluasi, dan satu atau lebih konsep dipilih untuk pengembangan dan percobaan lebih jauh. Dimana yang dimaksud dengan konsep disini adalah uraian dari bentuk, fungsi dan tampilan suatu produk dan biasanya disertai dengan sekumpulan spesifikasi, analisis produk-produk pesaing serta pertimbangan ekonomis proyek.
3. **Fase 2 Perancangan Tingkat Sistem** : Fase perancangan tingkat sistem mencakup definisi arsitektur produk dan uraian produk menjadi subsistem-subsistem serta komponen-komponen. Gambaran rakitan akhir untuk sistem produksi biasanya didefinisikan selama fase ini. *Output* pada fase ini biasanya mencakup tata letak bentuk produk, spesifikasi secara fungsional dari tiap subsistem produk, serta diagram aliran proses pendahuluan untuk proses rakitan akhir.
4. **Fase 3 Perancangan Rinci** : Fase perancangan rinci mencakup spesifikasi lengkap dari bentuk, material, dan toleransi-toleransi dari seluruh komponen unit

pada produk dan identifikasi seluruh komponen standar yang dibeli dari pemasok. Rencana proses dinyatakan dan peralatan dirancang untuk tiap komponen yang dibuat dalam sistem produksi. *Output* dari fase ini adalah pencatatan pengendalian untuk produk, gambar untuk tiap komponen produk dan peralatan produksinya, spesifikasi komponen-komponen yang dapat dibeli, rencana untuk proses pabrikasi dan perakitan produk.

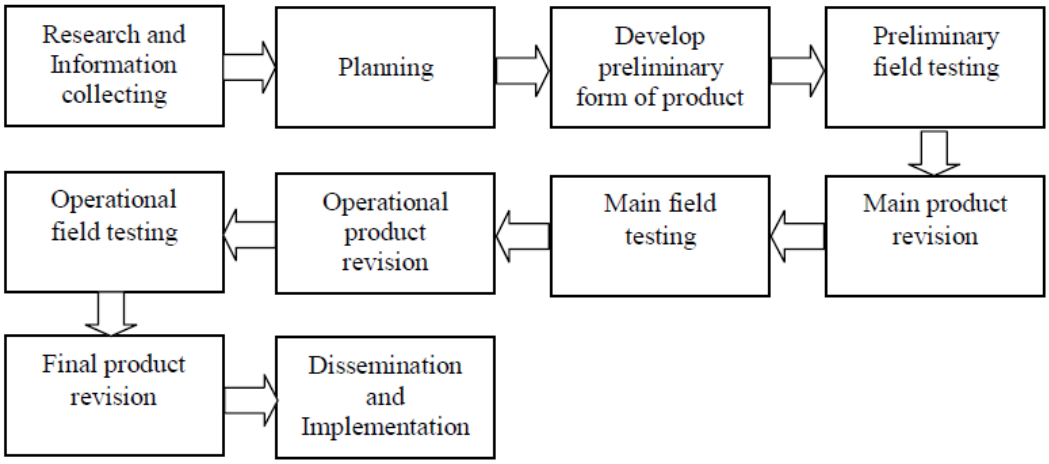
5. **Fase 4 Pengujian dan Perbaikan** : Fase pengujian dan perbaikan melibatkan konstruksi dan evaluasi dari bermacam-macam versi produksi awal produk. Prototipe awal (*alpha*) biasanya dibuat dengan menggunakan komponen-komponen dengan bentuk dan jenis material pada produksi sesungguhnya, namun tidak memerlukan proses pabrikasi dengan proses yang sama dengan yang dilakukan pada proses pabrikasi sesungguhnya. Prototipe *alpha* diuji untuk menetukan apakah produk yang akan bekerja sesuai dengan apa yang direncanakan dan apakah produk memuaskan kebutuhan utama konsumen. Prototipe berikutnya (*beta*) biasanya dibuat dengan komponen-komponen yang dibutuhkan pada produksi tidak dirakit dengan menggunakan proses perakitan akhir seperti pada perakitan sesungguhnya. Prototipe *beta* dievaluasi secara internal dan juga diuji oleh konsumen dengan menggunakannya secara langsung. Sasaran dari prototipe *beta* biasanya adalah untuk menjawab pertanyaan mengenai kinerja dan keandalan dalam rangka mengidentifikasi kebutuhan perubahan-perubahan secara teknik untuk produk akhir.
6. **Fase 5 Pelucuran Produk** : Pada fase peluncuran produk, yang terlebih dahulu dilakukan adalah produksi awal, dimana produk dibuat dengan menggunakan sistem produksi yang sesungguhnya. Tujuan dari produksi awal ini adalah untuk

melatih tenaga kerja dalam memecahkan permasalahan yang mungkin timbul pada proses produksi sesungguhnya. Produk-produk yang dihasilkan selama produksi awal kadang-kadang disesuaikan dengan keinginan pelanggan dan secara hati-hati dievaluasi untuk mengidentifikasi kekurangan-kekurangan yang timbul. Peralihan dari produksi awal menjadi produksi sesungguhnya harus melewati tahap demi tahap. Pada beberapa titik pada masa peralihan ini, produk diluncurkan dan mulai disediakan untuk didistribusikan.

Total keseluruhan fase adalah 6 fase, yakni: dari fase 0 sampai dengan fase 5, dan pemahaman dari tiap tahapan dapat dimengerti dan diterapkan secara terpisah (Ulrich-Eppinger, 2001).

2.2.2.2 Model Borg dan Gall

Menurut Borg dan Gall (1989:782), yang dimaksud dengan model penelitian dan pengembangan adalah *"a process used develop and validate educational product"*. Kadang-kadang penelitian ini juga disebut "*research based development*", yang muncul sebagai strategi dan bertujuan untuk meningkatkan kualitas pendidikan. Selain untuk mengembangkan dan memvalidasi hasil-hasil pendidikan, *research and development* juga bertujuan untuk menemukan pengetahuan-pengetahuan baru melalui "*basic research*", atau untuk menjawab pertanyaan-pertanyaan khusus tentang masalah-masalah yang bersifat praktis melalui "*applied research*", yang digunakan untuk meningkatkan praktik-praktik pendidikan. Menurut Borg dan Gall (1989:783-795), pendekatan *research and development* (R & D) dalam pendidikan meliputi sepuluh langkah. Adapun bagan langkah-langkah penelitiannya seperti ditunjukkan pada Gambar 2.26.



Gambar 2.26 Langkah-langkah Penelitian dan Pengembangan Produk menurut Borg dan Gall

Selanjutnya, untuk dapat memahami tiap langkah tersebut dapat dijelaskan sebagai berikut.

1. **Studi Pendahuluan (*Research and Information Collecting*)** : Langkah pertama ini meliputi analisis kebutuhan, studi literatur, penelitian skala kecil dan standar laporan yang dibutuhkan.
   a) **Analisis kebutuhan** : Untuk melakukan analisis kebutuhan ada beberapa kriteria, yaitu 1) Apakah produk yang akan dikembangkan merupakan hal yang penting bagi pendidikan? 2) Apakah produknya mempunyai kemungkinan untuk dikembangkan? 3) Apakah SDM yang memiliki keterampilan, pengetahuan dan pengalaman yang akan mengembangkan produk tersebut ada? 4) Apakah waktu untuk mengembangkan produk tersebut cukup?
   b) **Studi literatur** : Studi literatur dilakukan untuk pengenalan sementara terhadap produk yang akan dikembangkan. Studi literatur ini dikerjakan

untuk mengumpulkan temuan riset dan informasi lain yang bersangkutan dengan pengembangan produk yang direncanakan.

c) **Riset skala kecil** : Pengembang sering mempunyai pertanyaan yang tidak bisa dijawab dengan mengacu pada reseach belajar atau teks professional. Oleh karenanya pengembang perlu melakukan riset skala kecil untuk mengetahui beberapa hal tentang produk yang akan dikembangkan.

2. **Merencanakan Penelitian (*Planning*)** : Setelah melakukan studi pendahuluan, pengembang dapat melanjutkan langkah kedua, yaitu merencanakan penelitian. Perencaaan penelitian R & D meliputi: 1) merumuskan tujuan penelitian; 2) memperkirakan dana, tenaga dan waktu; 3) merumuskan kualifikasi peneliti dan bentuk-bentuk partisipasinya dalam penelitian.
3. **Pengembangan Desain (*Develop Preliminary of Product*)** : Langkah ini meliputi: 1) Menentukan desain produk yang akan dikembangkan (desain hipotetik); 2) menentukan sarana dan prasarana penelitian yang dibutuhkan selama proses penelitian dan pengembangan; 3) menentukan tahap-tahap pelaksanaan uji desain di lapangan; 4) menentukan deskripsi tugas pihak-pihak yang terlibat dalam penelitian.
4. ***Preliminary Field Testing*** : Langkah ini merupakan uji produk secara terbatas. Langkah ini meliputi: 1) melakukan uji lapangan awal terhadap desain produk; 2) bersifat terbatas, baik substansi desain maupun pihak-pihak yang terlibat; 3) uji lapangan awal dilakukan secara berulang-ulang sehingga diperoleh desain layak, baik substansi maupun metodologi.
5. **Revisi Hasil Uji Lapangan Terbatas (*Main Product Revision*)** : Langkah ini merupakan perbaikan model atau desain berdasarakan uji lapangan terbatas.

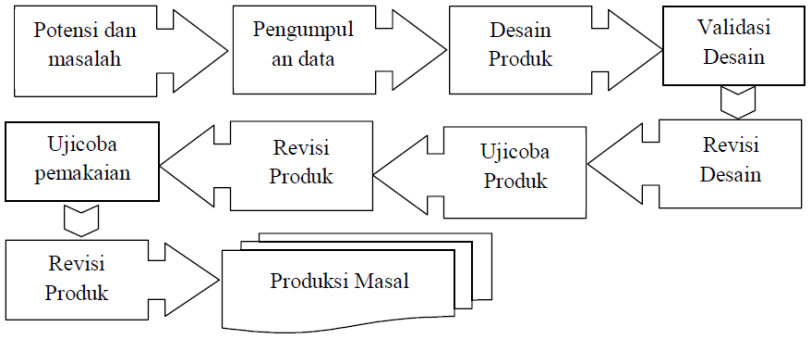
Penyempurnaan produk awal akan dilakukan setelah dilakukan uji coba lapangan secara terbatas. Pada tahap penyempurnaan produk awal ini, lebih banyak dilakukan dengan pendekatan kualitatif. Evaluasi yang dilakukan lebih pada evaluasi terhadap proses, sehingga perbaikan yang dilakukan bersifat perbaikan internal.

6. ***Main Field Test*** : Langkah merupakan uji produk secara lebih luas. Langkah ini meliputi 1) melakukan uji efektivitas desain produk; 2) uji efektivitas desain, pada umumnya, menggunakan teknik eksperimen model penggulangan; 3) Hasil uji lapangan adalah diperoleh desain yang efektif, baik dari sisi substansi maupun metodologi.
7. **Revisi Hasi Uji Lapangan Lebih Luas (*Operational Product Revision*)** : Langkah ini merupakan perbaikan kedua setelah dilakukan uji lapangan yang lebih luas dari uji lapangan yang pertama. Penyempurnaan produk dari hasil uji lapangan lebih luas ini akan lebih memantapkan produk yang kita kembangkan, karena pada tahap uji coba lapangan sebelumnya dilaksanakan dengan adanya kelompok kontrol. Desain yang digunakan adalah pretest dan posttest. Selain perbaikan yang bersifat internal. Penyempurnaan produk ini didasarkan pada evaluasi hasil sehingga pendekatan yang digunakan adalah pendekatan kuantitatif.
8. **Uji Kelayakan (*Operational Field Testing*)** : Langkah ini meliputi sebaiknya dilakukan dengan skala besar: 1) melakukan uji efektivitas dan adaptabilitas desain produk; 2) uji efektivitas dan adabtabilitas desain melibatkan para calon pemakai produk; 3) hasil uji lapangan adalah diperoleh model desain yang siap diterapkan, baik dari sisi substansi maupun metodologi.

9. **Revisi Final Hasil Uji Kelayakan (*Final Product Revision*)** : Langkah ini akan lebih menyempurnakan produk yang sedang dikembangkan. Penyempurnaan produk akhir dipandang perlu untuk lebih akuratnya produk yang dikembangkan. Pada tahap ini sudah didapatkan suatu produk yang tingkat efektivitasnya dapat dipertanggungjawabkan. Hasil penyempurnaan produk akhir memiliki nilai "generalisasi" yang dapat diandalkan.
10. **Desiminasi dan Implementasi Produk Akhir (*Dissemination and Implementation*)** : Laporan hasil dari R & D melalui forum-forum ilmiah, ataupun melalui media massa. Distribusi produk harus dilakukan setelah melalui *quality control*. Teknik analisis data, langkah-langkah dalam proses penelitian dan pengembangan dikenal dengan istilah lingkaran *research dan development* menurut Borg and Gall terdiri atas: (a) meneliti hasil penelitian yang berkaitan dengan produk yang akan dikembangkan, (b) mengembangkan produk berdasarkan hasil penelitian, (c) uji lapangan (d) mengurangi devisiensi yang ditemukan dalam tahap ujicoba lapangan.

2.2.2.3 Model Sugiyono

Menurut Sugiyono (2011:298), langkah-langkah penelitian dan pengembangan ada sepuluh langkah, yaitu sebagai berikut: (1) Potensi dan masalah, (2) Pengumpulan data, (3) Desain produk, (4) Validasi desain, (5) Revisi desain, (6) Ujicoba produk, (7) Revisi produk, (8) Ujicoba pemakaian, (9) Revisi produk, dan (10) Produksi masal. Adapun bagan langkah-langkah penelitiannya seperti ditunjukkan pada Gambar 2.27.



Gambar 2.27 Langkah-langkah Penelitian dan Pengembangan Produk menurut Sugiono
(Sumber : Metode Penelitian Kuantitatif, Kualitatif dan R&D, Sugiono)

Selanjutnya, untuk dapat memahami tiap langkah tersebut dapat dijelaskan sebagai berikut.

1. **Potensi dan Masalah** : Penelitian berawal dari adanya potensi atau masalah. Potensi adalah segala sesuatu yang bila didayagunakan akan memiliki nilai tambah.
2. **Mengumpulkan Informasi** : Setelah potensi dan masalah dapat ditunjukkan secara factual dan *up to date*, selanjutnya perlu dikumpulkan berbagai informasi yag dapat digunakan sebagai bahan untuk perencanaan produk tertentu yang diharapkan dapat mengatasi masalah tersebut. Metode yang akan digunakan untuk penelitian tergantung permasalahan dan ketelitian tujuan yang ingin dicapai.
3. **Desain Produk** : Produk yang dihasilkan dalam penelitian *research and development* bermacam-macam. Untuk menghasilkan sistem kerja baru maka peneliti harus membuat rancangan kerja baru yang dibuat berdasarkan penilaian terhadap sistem kerja lama, sehingga dapat ditemukan kelemahan-kelemahan terhadap sistem tersebut. Selain itu, peneliti harus mengadakan penelitian terhadap unit lain yang dipandang sistem kerjanya bagus. Selain itu harus

mengkaji referensi mutakhir yang terkait dengan sistem kerja yang modern berikut indikator sistem kerja yang baik. Hasil akhir dari kegiatan tersebut berupa desain produk baru yang lengkap dengan spesifikasinya. Desain ini masih bersifat hipotetik, karena efektivitasnya belum terbukti, dan akan dapat diketahui setelah melalui pengujian-pengujian. Desain produk harus diwujudkan dengan gambar atau bagan, sehingga akan memudahkan pihak lain untuk memahaminya.

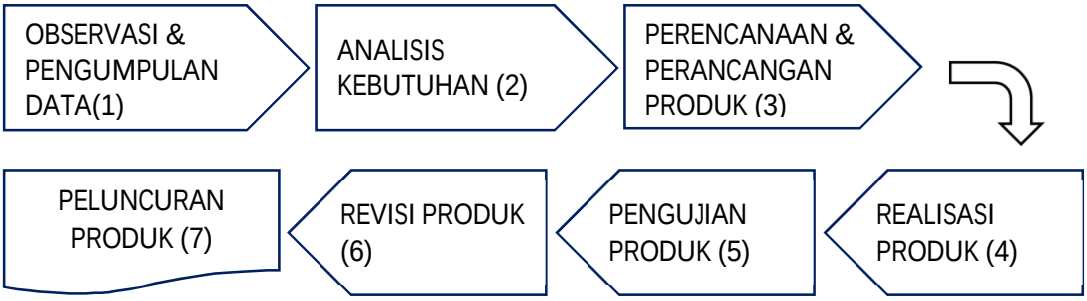
4. **Validasi Desain** : Validasi desain merupakan proses kegiatan untuk menilai apakah rancangan produk, dalam hal ini system kerja baru secara rasional akan lebih efektif dari yang lama. Dikatakan secara rasional karena validasi disini masih bersifat penilaian berdasarkan pemikiran rasional, belum merupakan fakta di lapangan. Validasi produk dapat dilakukan dengan cara menghadirkan beberapa pakar atau tenaga ahli yang sudah berpengalaman untuk menilai produk baru yang dirancang tersebut. Setiap pakar diminta untuk menilai desain tersebut, sehingga selanjutnya dapat diketahui kelemahan dan kekuatannya. Validasi desain dapat dilakukan dalam forum diskusi. Sebelum diskusi peneliti mempresentasikan proses penelitian sampai ditemukan desain tersebut, sekaligus keunggulannya.
5. **Perbaikan Desain** : Setelah desain produk divalidasi melalui diskusi dengan para pakar dan ahli lainnya, selanjutnya dapat diketahui kelemahannya. Kelemahan tersebut selanjutnya dicoba untuk dikurangi dengan cara memperbaiki desain. Yang bertugas memperbaiki desain adalah peneliti yang hendak menghasilkan produk tersebut.

6. **Uji Coba Produk** : Uji coba produk dapat dilakukan melalui eksperimen, yaitu membandingkan efektifitas dan efisiensi keadaan sebelum dan sesudah memakai sistem baru atau dengan membandingkan dengan kelompok yang tetap menggunakan sistem lama.
7. **Revisi Produk** : Pengujian produk pada sampel yang terbatas menunjukkan bahwa kinerja tindakan baru tersebut lebih baik dari tindakan lama.
8. **Ujicoba Pemakaian** : Setelah pengujian terhadap produk berhasil dan mungkin ada revisi yang tidak terlalu penting.
9. **Revisi Produk** : Revisi produk ini dilakukan apabila dalam pemakaian kondisi nyata terdapat kekurangan dan kelemahan. Dalam uji pemakaian, sebaiknya pembuat produk selalu mengevaluasi bagaimana kinerja produk dalam hal ini adalah sistem kerja atau tindakan.
10. **Pembuatan Produk Masal** : Pembuatan produk masal ini dilakukan apabila produk yang telah diujicoba dinyatakan efektif dan layak untuk diproduksi masal.

2.2.2.4 Sintesis Langkah-langkah Penelitian dan Pengembangan Produk

Berdasarkan model pengembangan produk yang dikemukakan oleh Karl T. Ulrich dan Steven D. Eppinger, Borg dan Gall, dan Sugiyono, masing-masing teori tersebut memiliki penamaan langkah-langkah yang berbeda, namun maksud dan tujuannya adalah sama. Dari ketiga model tersebut, model yang lebih sederhana adalah model Karl T. Ulrich dan Steven D. Eppinger, sedangkan model Borg dan Gall dan Sugiyono lebih menekankan kepada tahap pengujian dan revisi produk yang dilakukan lebih dari satu kali pengujian. Dari ketiga model yang telah dibahas, peneliti menyimpulkan menjadi langkah-langkah penelitian pengembangan baru

dengan dilakukan beberapa penyerderhanaan dan modifikasi menyesuaikan dengan keperluan penelitian. Penelitian diawali dengan observasi, yaitu dengan mengamati permasalahan yang terjadi di lapangan dan melakukan pengumpulan data untuk menunjang selama penelitian berlangsung, berikutnya peneliti melakukan analisis kebutuhan terhadap produk yang akan dibuat, kemudian pembuatan perencanaan dan perancangan sistem penelitian agar penelitian berjalan sesuai yang diharapkan dan terarah, lalu merealisasikan penelitian dan pengembangan produk sesuai rencana, kemudian melakukan pengujian terhadap produk yang telah terealisasi, setelah itu produk dievaluasi dan diperbaiki (revisi), hingga langkah penelitian terakhir produk dapat digunakan sesuai tujuan penelitian. Untuk lebih jelasnya, berikut 7 langkah penelitian pengembangan produk digambarkan dan dijelaskan pada Gambar 2.28.



Gambar 2.28 Langkah-langkah Penelitian dan Pengembangan Produk

Berikut adalah keterangan 7 (tujuh) langkah penelitian pengembangan produk :

1. **Observasi dan Pengumpulan Data** : Melakukan observasi dengan pengamatan permasalahan yang terjadi di lapangan hingga ditentukannya objek yang akan diteliti dan mengumpulkan berbagai informasi yang dapat digunakan sebagai bahan untuk perencanaan.

2. **Analisis Kebutuhan** : Melakukan analisa kebutuhan dengan memperhatikan keefektifan dan fungsi komponen sistem dan bahan-bahan yang akan digunakan untuk pembuatan produk.
3. **Perencanaan dan Perancangan Sistem** : Membuat perencanaan dan rancangan produk seperti perancangan diagram blok, perencanaan *input output*, perancangan rangkaian sistem, perancangan *flowhart* program, dan perancangan alat serta melakukan pengadaan komponen sistem dan bahan-bahan yang akan digunakan untuk pembuatan produk.
4. **Realisasi Produk** : Mulai membuat produk berdasarkan perencanaan dan perancangan yang telah dibuat.
5. **Pengujian Produk** : Melakukan uji coba sistem dan penggunaan produk.
6. **Revisi Produk** : Apabila ada kekurangan dalam penggunaan, maka produk diperbaiki.
7. **Peluncuran Produk** : Produk dapat digunakan sebagai pe-*monitoring* navigasi *quadcopter* dan selanjutnya dapat dikembangkan.

2.3 Konsep Produk yang Dikembangkan

Menurut Keller, K. L. (2003:19), Konsep produk menyatakan bahwa konsumen menyukai produk yang menawarkan kualitas, kinerja atau fitur inovatif terbaik. Manajer dari organisasi ini berfokus untuk membuat produk yang unggul dan senantiasa memutakhirkannya. Namun para manajer ini kadang-kadang terlibat persaingan dengan produk mereka. Mereka mungkin melakukan kesalahan yang diistilahkan “jebakan tikus yang lebih baik”, yaitu mempercayai bahwa jebakan tikus yang lebih baik akan mengarahkan orang ke pintu mereka, Suatu produk baru

tidak akan sukses jika tidak didukung oleh harga, distribusi, iklan dan penjualan yang tepat.

Inti dari perencanaan desain adalah terletak pada pengembangan konsep. Crawford (1994), mengemukakan bahwa konsep desain adalah kombinasi antara lisan, tulisan, dan atau bentuk prototipe yang akan dilakukan perbaikan dan bagaimana pelanggan menunjukkan keuntungan/kerugiannya. Tiga bagian penting yang diangkat untuk ide/perencanaan produk yang akan dikembangkan dalam bentuk konsep adalah :

1. Bentuk, Hal ini merupakan bentuk fisik suatu produk itu sendiri, material penyusun produk, dan sebagainya.
2. Teknologi, Termasuk di dalamnya antara lain: prinsip, teknik, perlengkapan, mekanika, kebijakan, dan seterusnya yang dapat digunakan untuk menciptakan/mencapai produk yang dimaksud.
3. Keuntungan, Nilai lebih/keuntungan yang diharapkan pelanggan dari produk tersebut.

Dengan memperhatikan teori tentang konsep produk yang telah dibahas, produk yang dibangun pada penelitian ini yaitu *Ground Control Station* (GCS) atau *Human Machine Interface* (HMI). GCS yang dibuat hanya untuk me-*monitoring quadcopter* melalui komputer, tidak untuk mengedalikan. GCS berfungsi untuk mengolah atau mengubah data *integer* dan *float* yang didapat dari hasil pengukuran parameter *quadcopter* menggunakan sensor-sensor yang dikirim melalui *wireless* menjadi data grafik 2D dan 3D yang dapat dipahami oleh pengguna. Para pengembang di belahan dunia sudah banyak yang membuat GCS *quadcopter* yang sangat berkualitas dan *user friendly*, yang produknya baik dikomersilkan maupun

open source. Berikut akan dibahas beberapa GCS yang telah ada dan populer namun *open source*.

2.3.1 Mission Planner

Mission Planner adalah *Ground Control Station open source* untuk Pesawat, helikopter dan *Rover*. Mission Planner merupakan aplikasi gratis yang dikembangkan oleh Michael Oborne dan didukung oleh komunitas pengembang UAV lainnya. Mission Planner kompatibel dengan Windows saja. Mission Planner dapat digunakan sebagai utilitas konfigurasi atau sebagai suplemen kontrol dinamis untuk kendaraan otonom pengguna (Ardupilot, 2017). Tampilan GCS Mission Planner ditunjukkan pada Gambar 2.29 dan berikut adalah beberapa hal yang dapat dilakukan dengan Mision Planner :

1. Memuat *firmware* (perangkat lunak) ke dalam autopilot (APM, PX4 ...) yang mengontrol kendaraan.
2. Setup, mengkonfigurasi, dan tune kendaraan untuk kinerja optimal.
3. Merencanakan, menyimpan dan memuat misi otonom ke autopilot pengguna dengan *point-and-click way-point* sederhana masuk di Google atau peta lainnya.
4. *Download* dan menganalisis log misi yang dibuat oleh autopilot pengguna.
5. Antarmuka dengan simulator penerbangan PC untuk membuat *hardware* penuh di lingkaran simulator UAV.
6. Dengan *hardware* telemetri yang tepat pengguna dapat: a) Memonitor status kendaraan saat beroperasi, b) Rekam log telemetri yang berisi informasi lebih pada papan log autopilot, c) Melihat dan menganalisis log telemetri, d) Mengoperasikan kendaraan di FPV (*first person view*).

Gambar 2.29 Mission Planner
(Sumber : http://ardupilot.org/planner/docs/mission-planner-overview.html)

2.3.2 OpenPilot GCS

OpenPilot GCS adalah *Ground Control Station* dan aplikasi konfigurasi untuk keluarga OpenPilot *open source* dari kontrol penerbangan, modem telemetri dan papan autopilot. Aplikasi OpenPilot digunakan untuk *firmware upload*, konfigurasi, kontrol dan pemantauan telemetri. Aplikasi OpenPilot mendukung semua papan OpenPilot yang tersedia (Showroom, 2017). OpenPilot GCS dapat beroperasi di Windows, Linux x32 / x64 dan MacOSX. OpenPilot benar-benar *open source*, yang dikembangkan oleh sekelompok non profit dari *hardware* dan *software* pengembang individu. OpenPilot GCS dikembangkan dalam C++ menggunakan Qt 5.2, QtGUI dan QtQuick2 untuk menerapkan berbagai solusi GUI digunakan untuk mengontrol, berkomunikasi dengan dan mengkonfigurasi *hardware* OpenPilot. OpenPilot GCS dibangun dengan desain yang sama seperti QtCreator dan sepenuhnya mendukung *plug-in*. OpenPilot GCS menggunakan beberapa perpustakaan *open source* lainnya untuk menyediakan fungsi yang

dibutuhkan untuk GCS. Sampai saat ini pengembangan OpenPilot masih terus berlanjut (https://showroom.qt.io/openpilot-gcs/). Tampilan GCS OpenPilot ditunjukkan pada Gambar 2.30 berikut ini.

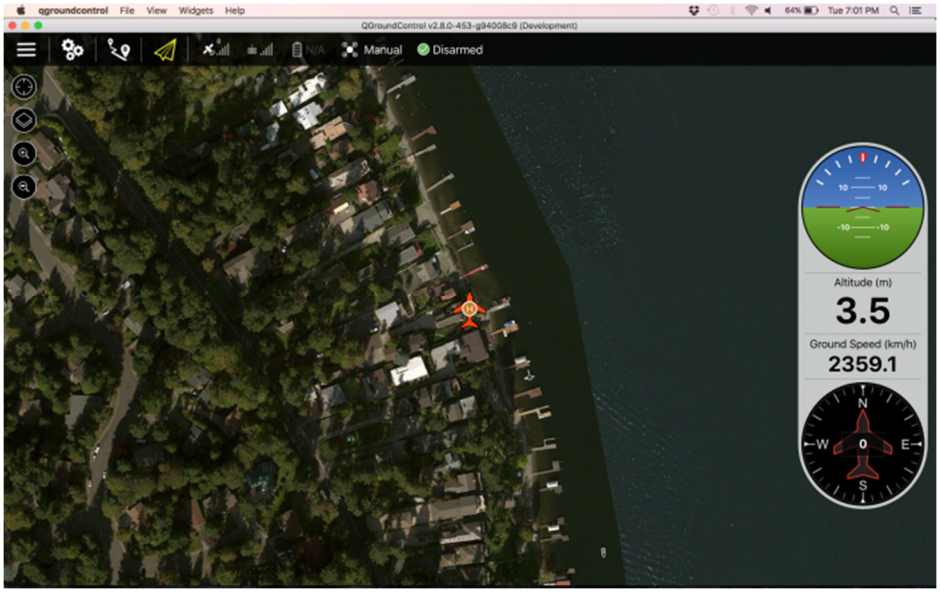
Gambar 2.30 OpenPilot GCS
(Sumber: http://www.hit-karlsruhe.de/hit-info/info-ws13/EFIS/EFIS%20Team1/standder technik.html)

2.3.3 QGroundControl

QGroundControl adalah sebuah unit kontrol operator / perangkat lunak *ground control* untuk kendaraan udara mikro. Hal ini memungkinkan untuk memvisualisasikan dan mengendalikan kendaraan udara mikro selama pengembangan dan operasi, baik *indoor* maupun *outdoor*. Dengan arsitektur perangkat lunak yang fleksibel mendukung beberapa jenis MAV / proyek *autopilot*. QGroundControl terutama dikembangkan oleh Lorenz Meier di ETH Zurich sebagai perangkat lunak GCS untuk Proyek PIXHAWK. Sekarang itu berkembang menjadi sebuah aplikasi *open-source* yang digunakan di beberapa universitas di seluruh dunia dan dengan banyak kontributor aktif (QGroundcontrol, 2017).

QGroundControl adalah aplikasi kaya fitur dengan pemandangan disesuaikan, masing-masing berisi koleksi komponen antarmuka yang paling berguna untuk tujuan tertentu. QgroundControl dikembangkan menggunakan bahasa pemrograman C++ berbasis Qt 4.7 dan dapat berjalan di Windows, Linux, MacOS dan Maemo. GCS QgroundControl dapat dilihat pada Gambar 2.31 dan berikut adalah fitur dari QGroundControl :

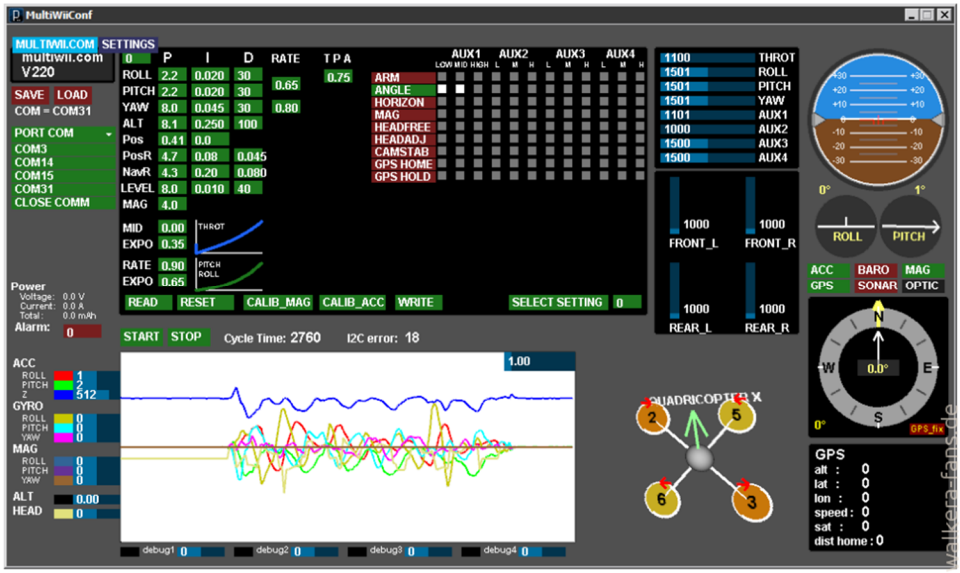
1. Dukungan multi-MAV
2. Dukungan multi-sistem (*multiple procotols*, beberapa pilot otomatis/proyek)
3. 2D peta bergerak (Google satelit/*Maps*, *OpenStreetMap*, Yahoo satelit/*Maps*)
4. perencanaan misi (*waypoints*, status sistem)
5. Tuning kontroler dan kalibrasi
6. Dukungan untuk singgle rotor, multi rotor dan sayap tetap (Pesawat, helikopter, *coaxial* dan desain quadrotor)
7. kontrol *joystick*
8. Output suara/audio diuji pada Linux, Mac.



Gambar 2.31 Qground Control
(Sumber : http://qgroundcontrol.org/about)

2.3.4 Multiwii GUI

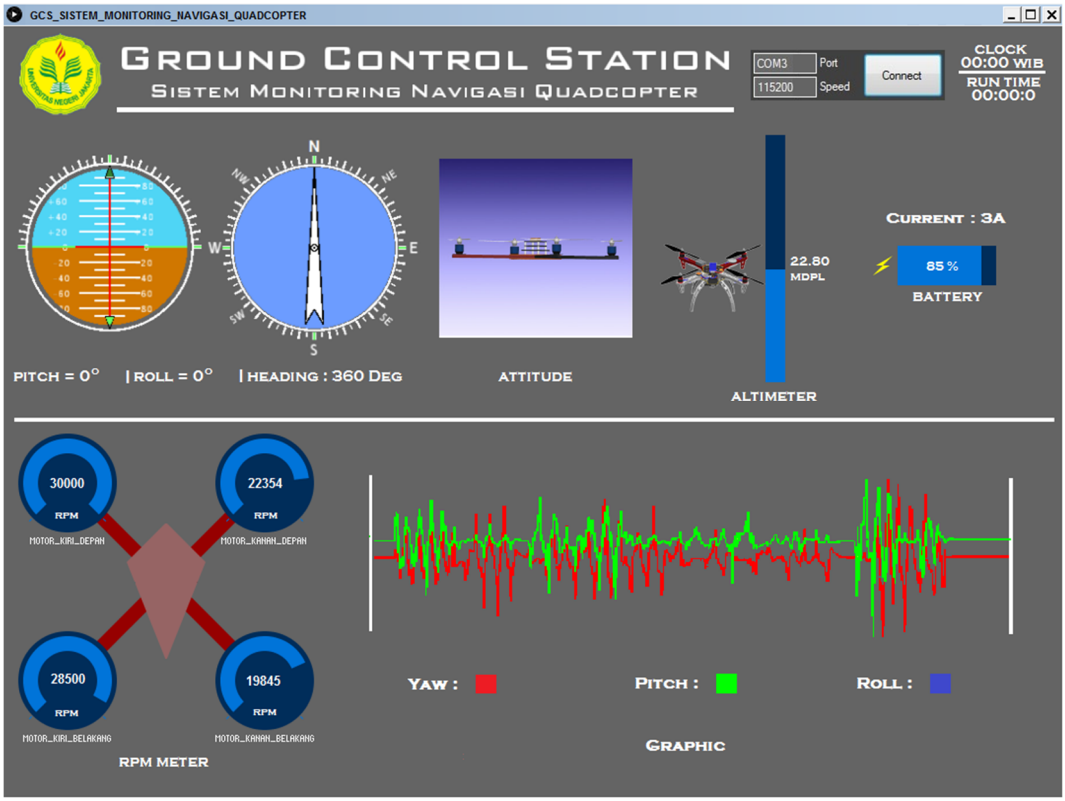
Multiwii GUI adalah sebuah antaramuka grafis yang dikeluarkan untuk mendukung perangkat keras Multiwii yang merupakan sebuah *platform opensource* yang dapat berjalan di Arduino untuk mengendalikan *multirotor* atau salah satunya adalah *quadcopter*. *Firmware* Multiwii dimulai oleh Prancis Alexandre Dubus sedangkan perangkat lunaknya didirikan oleh Alexin Paris di RC grup. Multiwii GUI dapat berjalan pada sistem operasi komputer seperti mac, linux dan lainnya. Beberapa kelebihan *platform* Multiwii diantaranya adalah *flight controller* begitu minimalis, murah dan mudah untuk digunakan, GUI sederhana dan *opensource* serta fitur dapat ditambahkan sesuai dengan keinginan pengguna. Sudah banyak fitur-fitur yang dikembangkan, seperti fitur gimbal kamera yang dapat mempertahan objek, *multirotor* dapat dikendalikan menggunakan Wii Nunchuck, termasuk navigasi GPS bahkan sampai sekarang masih dalam pengembangan yang dikembangkan oleh komunitas pengembang Multiwii (Multiwii, 2017). Tampilan MultiWiii GUI ditunjukkan pada Gambar 2.32.



Gambar 2.32 Multiwii GUI
(Sumber : https://walkera-fans.de/runner-250-basiert-auf-multiwii-2-2/)

2.3.5 Rancangan *Ground Control Station Penelitian*

Rancangan *Ground Control Station* yang akan dibuat yaitu dapat menampilkan data visual berupa *Heading/Yaw*, *Pitch*, *Roll*, *attitude* / 3D *quadcopter*, grafik kestabilan *Yaw Pitch Roll*, RPM motor 1 - 4, *Altimeter*, dan *Battery sensor*. Tampilan rancangan GCS penelitian dan fitur-fiturnya ditunjukkan dan dijelaskan pada Gambar 2.33.



Gambar 2.33 Rancangan *Ground Control Station*

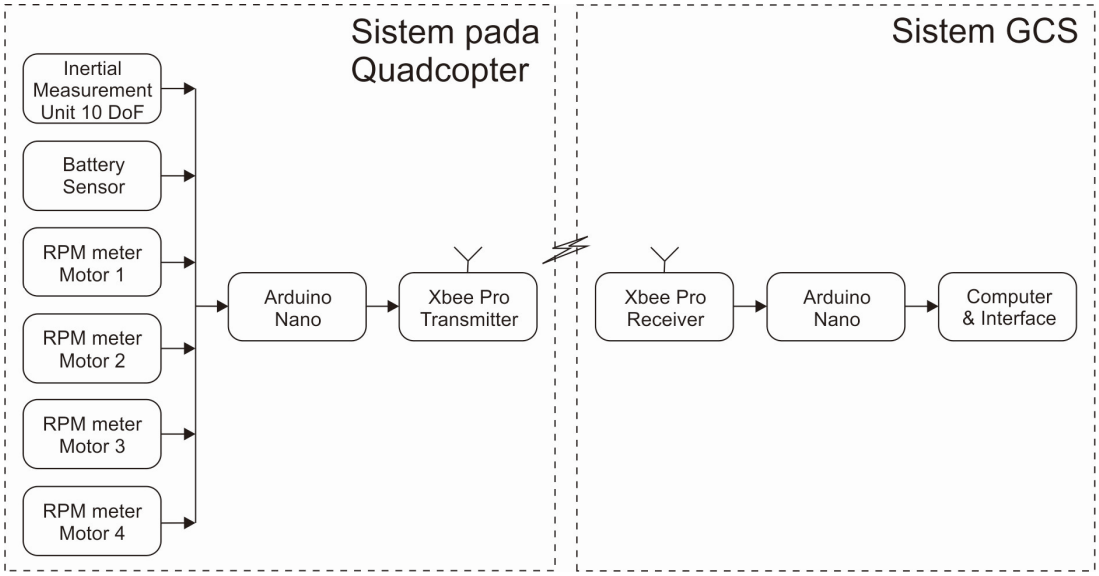
1. ***COM Port*** : untuk menampilkan pemberitahuan *port* serial berapa yang terhubung ke komputer, yang menandakan bahwa perangkat arduino telah terhubung dengan komputer atau belum.
2. ***Speed (Baud rate)*** : untuk menampilkan pemberitahuan *baud rate* berapa yang terbaca oleh komputer untuk berkomunikasi.

3. **Tombol *Connect*** : untuk memulai menerima data serial secara *real time* dari perangkat sistem *monitoring* pada *quadcopter*.
4. ***Manual Intruction*** : digunakan untuk membantu pengguna dalam mengoperasikan *Ground Control Station*.
5. ***Clock*** : untuk menampilkan waktu Indonesia bagian Barat.
6. ***Run Time*** : untuk menampilkan berapa lama pengguna menampilkan *Ground Control Station.*
7. ***Rotary Pitch-Roll*** : untuk menampilkan nilai sudut kemiringan *quadcopter* (atas-bawah dan putar kiri dan putar kanan) yang dinyatakan dalam satuan derajat.
8. ***Rotary Heading*** : untuk menunjukkan arah hadap *quadcopter* (kompas).
9. **3D *shape Attitude*** : untuk memperlihatkan sikap *quadcopter* ketika terbang diudara yang dinyatakan dalam bentuk gambar 3 dimensi.
10. ***Graphic*** : untuk menampilkan data kestabilan *yaw, pitch* dan *roll* dalam bentuk grafik.
11. **RPM meter *Slider*** : untuk menampilkan data kecepatan rotasi motor dalam satuan menit.
12. ***Altimeter Slider*** : untuk mengetahui berapa ketinggian *quadcopter* yang dinyatakan dalam satuan MDPL (meter dari permukaan laut).
13. ***Battery Slider*** : untuk mengetahui keadaan kapasitas baterai yang dinyatakan dalam satuan persen.

2.4 Rancangan Produk

2.4.1 Diagram Blok Sistem *Monitoring* Navigasi *Quadcopter*

Sebelum merealisasikan produk, dilakukan perancangan sistem kendali, dimana sebuah sistem terdapat *input*, proses dan *output*. Berikut pada Gambar 2.34 ditunjukkan diagram blok untuk mengetahui komponen yang saling terhubung dan bekerja pada sistem *monitoring* navigasi *quadcopter*.



Gambar 2.34 Diagram Blok Sistem *Monitoring* Navigasi *Quadcopter*

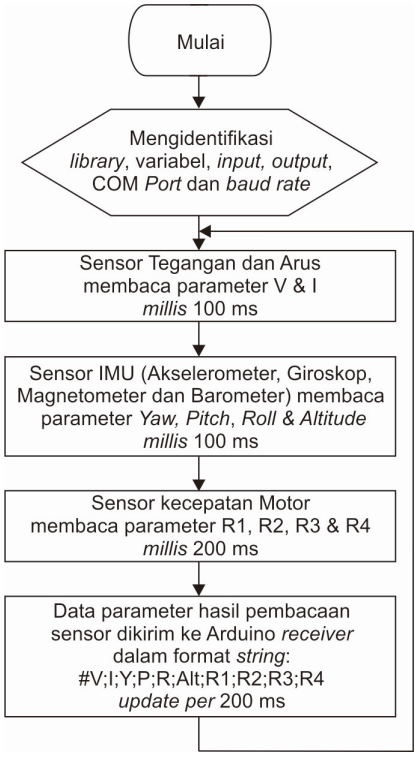
2.4.2 Rancangan *Flowchart* Program

Dalam pembuatan alat, setelah menetukan bagian *input*, proses dan *output*, maka langkah selanjutnya adalah merancang *flowchart* untuk membuat program yang akan digunakan pada alat tersebut. Berikut ini penjelasan *flowchart* program yang akan dibuat menggunakan *software* Arduino IDE dan Processing IDE.

2.4.2.1 *Flowchart* Arduino Nano *Transmitter*

Proses kerja program pada arduino *transmitter* merupakan tahap untuk memperoleh data dari sensor-sensor yang selanjutnya dikirim ke sistem arduino

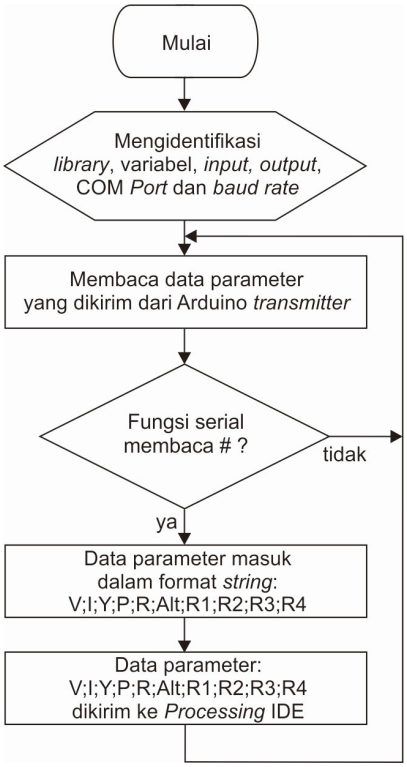
receiver secara *real time* dengan *update* per 200 ms. Kerja program arduino nano *transmitter* diawali dengan persiapan mengidentifikasi *library*, variabel, *input*, *output*, COM *port* dan *baud rate*. Selanjutnya melakukan proses pembacaan parameter tegangan dan arus (variabel V dan I) tiap 100 ms. Kemudian melakukan proses pembacaan parameter sudut *Yaw*, *Pitch*, *Roll* dan *Altitude* (variabel Y, P, R dan Alt) tiap 100 ms. Lalu melakukan proses pembacaan parameter kecepatan motor 1 - 4 tiap 200 ms. Dari hasil pembacaan sensor-sensor, kemudian dicetak atau di *serial print* dengan format string # V; I; Y; P; R; Alt; R1; R2; R3; R4 yang di *update* per 200 ms. Untuk alur lebih jelasnya ditunjukkan *Flowchart* Arduino Nano *Transmitter* pada Gambar 2.35.



Gambar 2.35 *Flowchart* Arduino Nano *Transmitter*

2.4.2.2 *Flowchart* Ardunio Nano *Receiver*

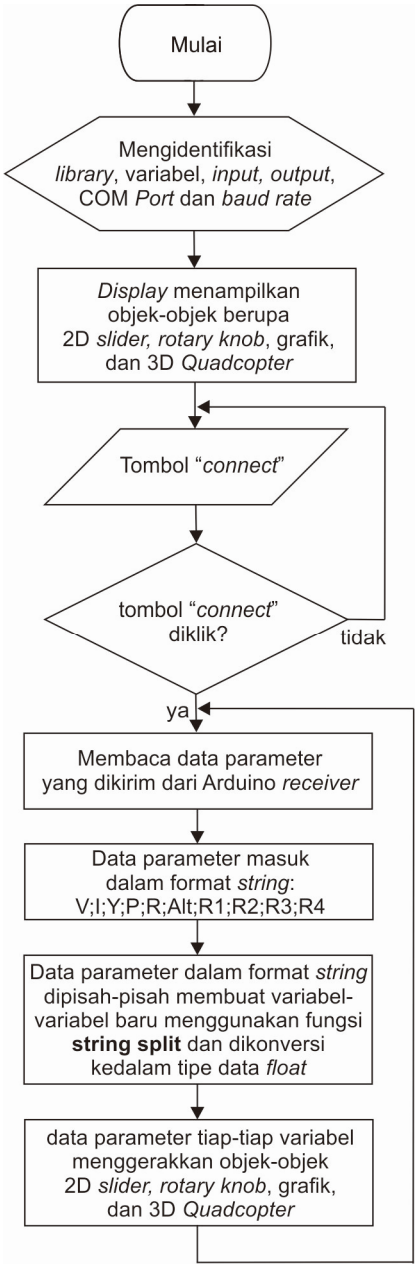
Proses kerja program pada arduino *receiver* merupakan tahap untuk membaca data yang dikirim dari arduino *transmitter*. Kerja program arduino *receiver* diawali dengan persiapan mengidentifikasi *library*, variabel, *input*, *output*, COM *port* dan *baud rate*. Selanjutnya melakukan proses pembacaan parameter yang dikirim oleh arduino *transmitter* yang dikirim dalam format string # V; I; Y; P; R; Alt; R1; R2; R3; R4. Metode yang digunakan dalam proses pembacaan string, dalam hal ini fungsi serial harus membaca tanda # (*hastag*), sehingga data string yang masuk menjadi V; I; Y; P; R; Alt; R1; R2; R3; R4. Kemudian data dalam format string V; I; Y; P; R; Alt; R1; R2; R3; R4 dikirim ke aplikasi processing. Dengan demikian, arduino nano *receiver* digunakan sebagai perantara antara XBee dengan komputer dan processing. Untuk alur lebih jelasnya ditunjukkan *flowchart* arduino nano *receiver* pada Gambar 2.36.



Gambar 2.36 *Flowchart* Arduino Nano *Receiver*

2.4.2.3 *Flowchart* Processing IDE

Proses kerja program pada processing merupakan tahap untuk membaca data yang dikirim dari arduino *receiver*. Kerja program processing diawali dengan mengklik tombol *running* pada aplikasi processing (GCS), lalu processing melakukan persiapan mengidentifikasi *library*, variabel, *input*, *output*, COM *port* dan *baud rate*. Setelah itu, aplikasi processing menampilkan objek-objek berupa *slider*, *rotary knob*, grafik, dan 3D *quadcopter*, namun objek-objek masih belum dapat bergerak. Pada aplikasi processing (GCS), terdapat tombol koneksi untuk memulai penerimaan data. Jika tombol digeser ke kiri, maka selanjutnya processing melakukan proses pembacaan parameter yang dikirim dari arduino *receiver* berupa susunan atau format string V; I; Y; P; R; Alt; R1; R2; R3; R4. Dari format string V; I; Y; P; R; Alt; R1; R2; R3; R4 yang diterima, kemudian data tersebut dipisah-pisah kembali menjadi masing-masing satu paremeter dan ditempatkan ke variabel baru dari processing. Kemudian variabel tersebut dimasukkan kedalam fungsi objek-objek, sehingga objek-objek dapat bergerak dengan fungsinya masing-masing. Untuk alur lebih jelasnya ditunjukkan *flowchart* processing pada Gambar 2.37.



Gambar 2.37 *Flowchart* Processing IDE

BAB III

METODOLOGI PENELITIAN

3.1 Tempat dan Waktu Penelitian

Penelitian dilaksanakan di Laboratorium Mekatronika dan Robotika dan Laboratorium Bengkel Mekanik di Fakultas Teknik Universitas Negeri Jakarta. Waktu penelitian dilaksanakan pada bulan September 2016 s.d. Juni 2017.

3.2 Metode Pengembangan Produk

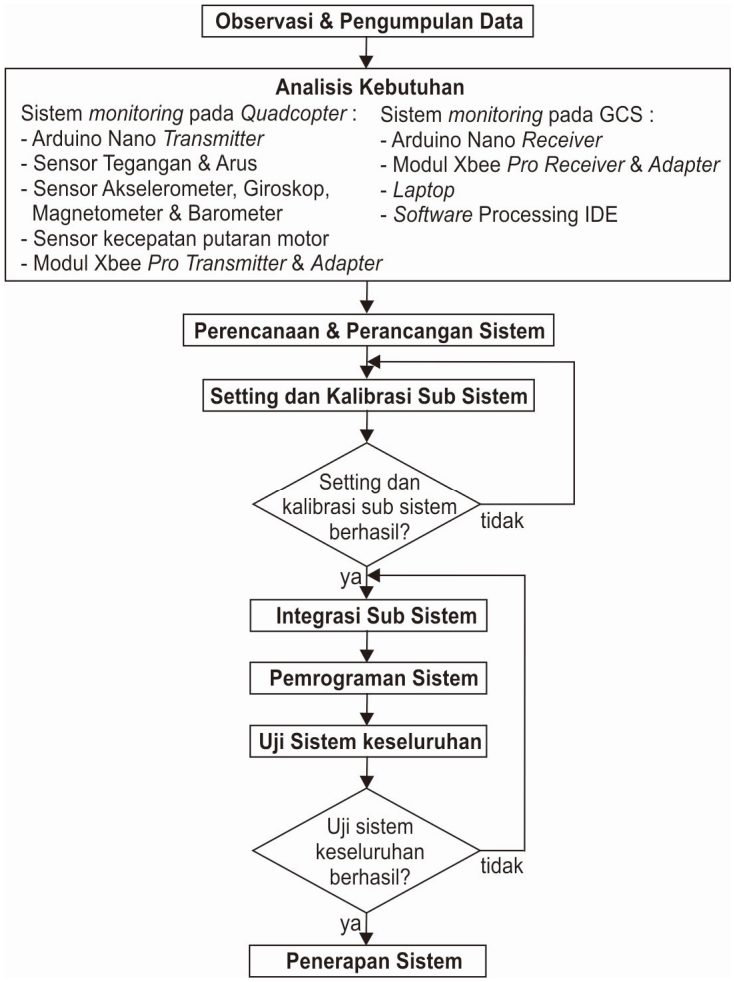
3.2.1 Tujuan Pengembangan

Tujuan penelitian dan pengembangan ini adalah merancang, merealisasikan, dan menguji Sistem *Monitoring* Navigasi *Quadcopter* berbasis Arduino Nano dengan pengiriman data secara *wireless* menggunakan modul XBee *Pro* S1 yang kemudian akan ditampilkan ke dalam antarmuka yang telah dibuat menggunakan *software* Processing IDE untuk mempermudah pemantauan pergerakan *quadcopter* agar dapat dinavigasikan dengan baik.

3.2.2 Metode Pengembangan

Metode pengembangan yang digunakan dalam penelitian ini ada 8 tahap. Tahap 1, Peneliti melakukan observasi dengan pengamatan permasalahan yang terjadi di lapangan hingga ditentukannya objek yang akan diteliti dan mengumpulkan berbagai informasi yang dapat digunakan sebagai bahan untuk perencanaan. Tahap 2, Melakukan analisa kebutuhan dengan memperhatikan keefektifan dan fungsi komponen sistem dan bahan-bahan yang akan digunakan untuk pembuatan produk. Tahap 3, Membuat perencanaan dan rancangan produk seperti perancangan diagram blok, perencanaan *input output*, perancangan

rangkaian sistem, perancangan *flowhart* program, dan perancangan alat serta melakukan pengadaan komponen sistem dan bahan-bahan yang akan digunakan untuk pembuatan produk. Tahap 4, Melakukan penyettingan dan kalibrasi terhadap tiap-tiap sub sistem agar sistem keseluruhan dapat bekerja dengan baik dan akurat. Tahap 5, Melakukan integrasi (pengawatan) tiap-tiap sub sistem agar menjadi satu kesatuan sistem. Tahap 6, Melakukan pemrograman terhadap sistem keseluruhan. Tahap 7, Melakukan pengujian terhadap sistem untuk membuktikan bahwa sistem berhasil atau dilakukan perbaikan. Terakhir tahap 8, Melakukan penerapan terhadap sistem, kemudian produk dapat digunakan. Alur metode pengembangan yang dilakukan, ditunjukkan pada Gambar 3.1.



Gambar 3.1 Metode Pengembangan

3.2.3 Sasaran Produk

Sasaran utama produk ini adalah untuk pengguna atau para penghobi *quadcopter* agar dapat mengetahui pergerakan dan performa *quadcopter* ketika terbang diudara. Kemudian sasaran lain adalah untuk mahasiswa dalam mempelajari pemrograman berorientasi objek melalui wahana *quadcopter*.

3.2.4 Instrumen

3.2.4.1 Instrumen Penelitian

Instrumen penelitian adalah alat ukur yang digunakan dalam penelitian. Tabel 3.1 adalah alat ukur yang digunakan dalam penelitian ini.

Tabel 3.1 Instrumen Penelitian

No.	Instrumen	Merk / Model
1.	*Multimeter* digital	Sanwa
2.	*Clamp Multimeter* (Tang Ampere)	APPA A16R (*True* RMS)
3.	*Tachometer* laser digital	DT-2234C+
4.	Busur	*Butterfly*
5.	Meteran	-

Fungsi Instrumen penelitian :

1. Multimeter digital, digunakan untuk mengukur tegangan baterai *lithium polymer quadcopter*.
2. *Clamp multimeter* (tang ampere), digunakan untuk mengukur arus baterai *lithium polymer quadcopter*.
3. *Tachometer* laser digital, digunakan untuk mengukur jumlah putaran motor BLDC dalam satuan rpm (*rotation per minutes*).
4. Busur, digunakan untuk mengukur sudut *yaw*, *pitch* dan *roll* pada *quadcopter*.
5. Meteran dan tali, digunakan untuk mengukur ketinggian *quadcopter* ketika terbang diudara.

3.2.4.2 *Software* Penelitian

Software penelitian adalah perangkat lunak yang digunakan untuk membantu proses penelitian, yaitu baik untuk proses penulisan, membuat desain, simulasi dan membuat GCS (*Ground Control Station*). Tabel 3.2 adalah *software-software* yang digunakan dalam penelitian ini.

Tabel 3.2 *Software* Penelitian

No.	*Software*
1.	Arduino IDE 1.6.9
2.	Processing IDE 3.2.1
3.	Microsoft Word 2013
4.	Corel Draw X7
5.	Photoshop CS3
6.	Eagle Professional 6.4.0
7.	Google SketchUp 2016
8.	Paint
9.	X-CTU

3.2.4.3 Alat Penelitian

Alat penelitian adalah alat yang digunakan untuk menunjang proses penelitian. Tabel 3.3 adalah alat yang digunakan dalam penelitian ini.

Tabel 3.3 Alat Penelitian

No.	Alat
1.	Solder
2.	Atraktor
3.	Tang Kombinasi
4.	Tang Potong
5.	Obeng +
6.	Obeng Heksagon
7.	Gunting
8.	*Cutter* (pisau biasa)
9.	*Cutter* PCB
10.	Mesin Bor
11.	Mesin Gerinda
12.	Kikir
13.	Li-Po *Charger*
14.	Li-Po *Checker*

3.2.4.4 Bahan Penelitian

Bahan penelitian adalah bahan-bahan atau komponen yang digunakan untuk diteliti. Tabel 3.4 adalah bahan yang digunakan dalam penelitian ini.

Tabel 3.4 Bahan Penelitian

No.	Bahan
1.	*Laptop*
2.	Motor BLDC
3.	*Propeller*
4.	ESC (*Electronic Speed Controller*)
5.	*Frame Quadcopter*
6.	*Radio Control* (TX & RX)
7.	Ardupilot APM 2.8
8.	Baterai Lipo 3000 mAH
9.	Arduino Nano (TX & RX)
10.	3DR *Power Module*
11.	Sensor RPM
12.	IMU (*Inertial Measurement Unit*) 10 DOF
13.	XBee Pro S1 (TX & RX)
14.	Kabel USB
15.	PCB
16.	Tulang Ikan
17.	Soket Tulang Ikan
18.	Kabel *Jumper*
19.	Spiser
20.	Timah
21.	Pasta Solder
22.	*Spoons* Solder
23.	Lakban hitam
24.	*Polyfoam*
25.	*Cable Ties*
26.	*Double Tip*
27.	*Box* hitam

3.3 Prosedur Pengembangan

3.3.1 Tahap Penelitian dan Pengumpulan Informasi

Teknologi UAV, khususnya *quadcopter* belakangan ini dalam kehidupan sehari-hari sering digunakan untuk mendukung dalam melakukan suatu misi dan kegiatan-kegiatan tertentu. Contoh penggunaan UAV untuk pencarian korban

bencana pada kondisi ekstrim, penginderaaan jarak jauh seperti sistem *monitoring* serta bermanfaat sebagai alat pemetaan dan pengawasan pada suatu wilayah dan lain-lain.

Dalam melakukan suatu misi dan kegiatan-kegiatan tertentu menggunakan *quadcopter*, agar suatu misi dapat berjalan dengan baik, diperlukan sistem pengawasan atau *monitoring* terhadap pergerakan dan performa *quadcopter* untuk mengurangi kesalahan-kesalahan yang memungkinkan dapat terjadi dilapangan. Sebagai contoh, ketika mengoperasikan *quadcopter* secara jarak jauh dan tidak terpantau secara langsung oleh penginderaan manusia atau pengguna (*pilot*), hal ini pengguna sulit untuk menentukan dan memperkirakan navigasi (arah) *quadcopter* itu sendiri, serta contoh lain adalah untuk mengawasi performa kapasitas baterai pada *quadcopter*.

Untuk membuat sistem *monitoring* navigasi *quadcopter*, terlebih dahulu peneliti menentukan parameter-parameter *quadcopter* yang akan di-*monitoring* sesuai dengan tujuan penelitian dan kebutuhan. Parameter-parameter *quadcopter* yang akan di-*monitoring* antara lain tegangan dan arus baterai, sudut *yaw-pitch-roll*, *altimeter*, dan kecepatan putaran tiap-tiap motor dalam satuan menit (RPM). Setelah parameter-paremeter yang akan diukur telah ditentukan, selanjutnya peneliti menetukan modul-modul yang digunakan, antara lain adalah mikrokontroler Arduino Nano yang berfungsi sebagai sub sistem pengolah data, 3DR *power module* berfungsi untuk membaca tegangan dan arus baterai, modul akselero, giroskop, magnetometer dan barometer berfungsi untuk membaca sudut kemiringan *yaw-pitch-roll* dan *altimeter quadcopter*, dan modul *speed sensor* berfungsi untuk membaca kecepatan putaran tiap-tiap motor dalam satuan menit

(RPM), serta untuk pengiriman data secara nirkabel menggunakan modul XBee *pro* S1 (TX dan RX).

Untuk memvalidasi hasil pengukuran, peneliti melakukan penyettingan atau kalibrasi pada tiap-tiap modul yang digunakan untuk mengukur parameter-parameter *quadcopter*, yaitu dengan mengkondisikan modul atau sensor sesuai dengan spesifikasi dan cara kerja yang tetera pada *datasheet* modul. Penyettingan dan kalibrasi yang dimaksud diantaranya melakukan pengkalibrasian 3DR *power module*, pengkalibrasian modul IMU (akselero giroskop), pengkalibrasian modul GPS, pengkalibrasian modul *speed sensor* dan penyettingan modul XBee *pro* S1.

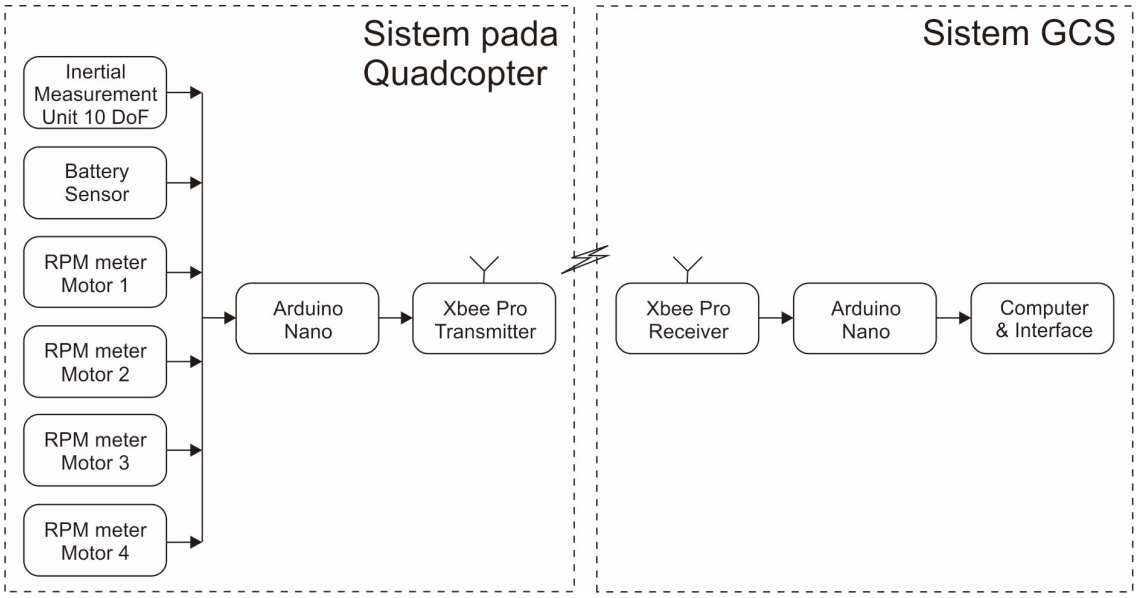
3.3.2 Tahap Perencanaan dan Perancangan Sistem

Perencanaan dan perancangan sistem adalah mendesain suatu sistem dengan langkah–langkah yang tepat sehingga menghasilkan sistem yang baik. Perancangan alat digunakan untuk menentukan komponen penyusun dari suatu alat yang dibuat, sehingga hasil akhirnya sesuai dengan yang diinginkan. Pada tahap perencanaan dan perancangan sistem terdapat 5 tahap, diantaranya adalah: 1) Perancangan diagram blok, 2) Perencanaan *input* dan *output*, 3) Perancangan rangkaian sistem *monitoring* pada *quadcopter*, 4) Perancangan rangkaian sistem *Ground Control Station*, dan 5) Perancangan *flowchart* program. Ke 5 (lima) tahap perencanaan dan perancangan sistem dibahas pada rincian berikut.

3.3.2.1 Perancangan Diagram Blok

Dalam pembuatan alat, diagram blok merupakan termasuk tahap awal perancangan yang sangat penting untuk menentukan susunan blok-blok pada suatu sistem, agar bila terjadi kegagalan pada sistem dapat segera diidentifikasi

kerusakannya perbagian blok. Pada penelitian ini, sistem digolongkan menjadi 11 bagian blok. Berikut diagram blok sistem *monitoring* navigasi *quadcopter* ditunjukkan pada Gambar 3.2 beserta penjelasannya.



Gambar 3.2 Diagram Blok Sistem *Monitoring* Navigasi *Quadcopter*

1. **IMU** : terdiri dari sensor Akselerometer, Girometer, Magnetometer dan Barometer berfungsi untuk mengukur sudut kemiringan (*pitch* dan *roll*), untuk mengetahui arah/kompas (*yaw*), dan untuk mengetahui ketinggian *quadcopter*.
2. ***Battery* Sensor** : berfungsi untuk mengetahui pemakaian kapasitas baterai.
3. **RPM meter Motor 1 – 4** : berfungsi untuk mengukur RPM pada tiap-tiap motor.
4. **Arduino Nano** : untuk memproses data dari sensor-sensor yang selanjutnya data dikirim melalui XBee menggunakan komunikasi *wireless* serial.
5. **XBee *Pro Transmitter*** : berfungsi untuk mengirim data yang dihasilkan dari Arduino Nano untuk dikirim ke sistem GCS.
6. **XBee *Pro Receiver*** : berfungsi untuk menerima data yang dikirim dari XBee *Transmitter*.

7. **Arduino Nano** : berfungsi untuk mengolah data agar dapat terhubung dengan komputer dan data diproses oleh *software processing* IDE.
8. ***Computer* dan *Interface*** : berfungsi untuk mengubah data *integer*/*float* menjadi data berorientasi objek berupa 2 dimensi dan 3 dimensi.

3.3.2.2 Perencanaan *Input* dan *Output*

Setelah melakukan perancangan diagram blok, selanjutnya dilakukan perancangan *input* dan *output* sistem, dimana untuk menghubungkan *pin-pin input* dan *output* agar memudahkan dalam melakukan perancangan rangkaian sistem. perancangan *input* dan *output* sistem ditunjukkan pada Tabel 3.5 dan 3.6.

Tabel 3.5 *Input Output* Sistem *Monitoring* pada *Quadcopter*

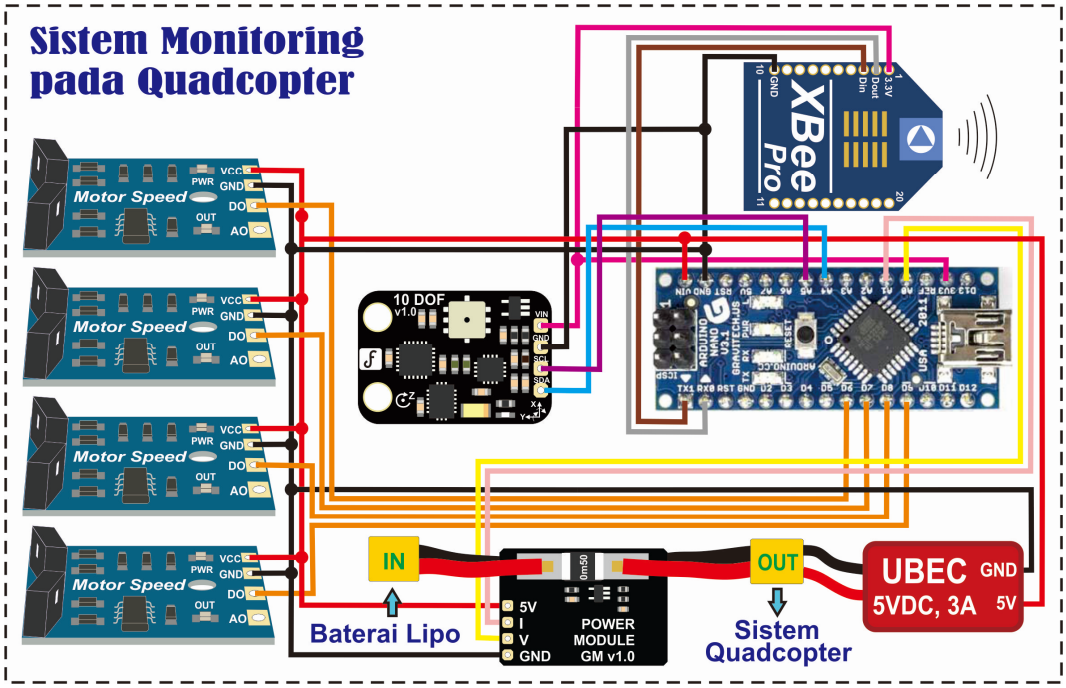
Pin Arduino Nano *Transmitter*	Modul Sensor	*Input / Output*
A0	3DR *Power Module*	Masukan sensor Arus
A1		Masukan sensor Tegangan
A4	Sensor IMU (Akselero-meter, Giroskop, Magne-tometer dan Barometer)	Masukan SDA (I2C) IMU
A5		Masukan SCL (I2C) IMU
D6	Sensor RPM Motor (Kecepatan putaran Motor)	Masukan sensor RPM Motor 1
D7		Masukan sensor RPM Motor 2
D8		Masukan sensor RPM Motor 3
D9		Masukan sensor RPM Motor 4
RX0	Modul XBee Pro S1 (*Transmitter*)	Masukan dari *transmitter* (TX) XBee
TX1		Keluaran ke *receiver* (RX) XBee

Tabel 3.6 *Input Output* Sistem *Monitoring* pada GCS

Pin Arduino Nano *Receiver*	Modul Sensor	*Input / Output*
RX0	Modul XBee Pro S1 (*Receiver*)	Masukan dari *transmitter* (TX) XBee
TX1		Keluaran ke *receiver* (RX) XBee

3.3.2.3 Perancangan Rangkaian Sistem *Monitoring* pada *Quadcopter*

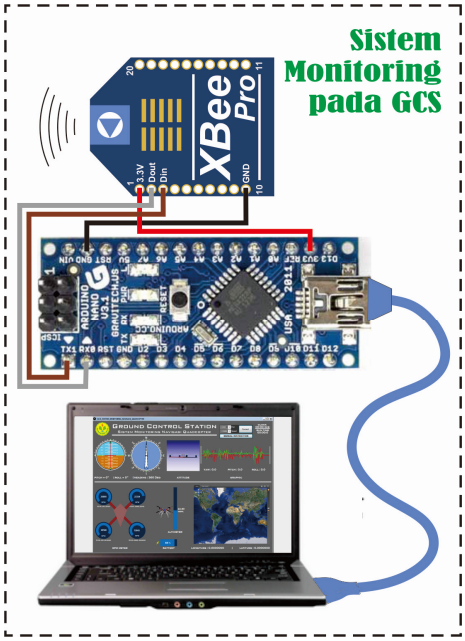
Setelah melakukan perancangan diagram blok dan penentuan *input* dan *output*, selanjutnya dilakukan perancangan rangkaian sistem *monitoring* pada *quadcopter*, dimana untuk menghubungkan *pin-pin input output* agar memudahkan dalam pembuatan jalur *layout shield* pada papan PCB. Rangkaian sistem dirancang menggunakan Corel Draw X7. Rangkaian sistem *monitoring* pada *quadcopter* ditunjukkan pada Gambar 3.3.



Gambar 3.3 Perancangan Rangkaian Sistem *Monitoring* pada *Quadcopter*

3.3.2.4 Perancangan Rangkaian Sistem *Ground Control Station*

Perancangan rangkaian sistem *Ground Control Station* sama halnya dengan perancangan rangkaian sistem pada *quadcopter*, hanya saja pada rancangan rangkain sistem ini tidak dibuatkan *shield*, melainkan di pasang pada *box* hitam dan menghubungkannya dengan cara di *jumper*. Rancangan rangkaian sistem pada *quadcopter* ditunjukkan pada Gambar 3.4.



Gambar 3.4 Perancangan Rangkaian Sistem *Ground Control Station*

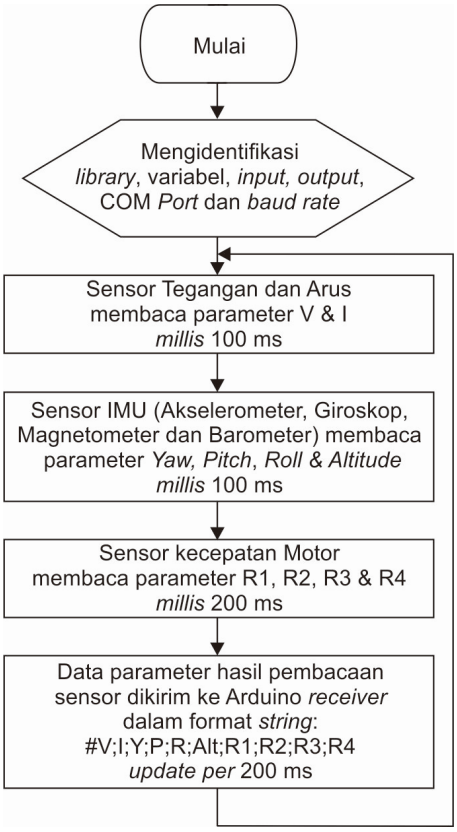
3.3.2.5 Perancangan *Flowchart* Program

Perancangan perangkat lunak adalah salah satu tahap utama dalam rancang bangun sistem *monitoring* navigasi *quadcopter*. Perancangan yang dimaksud adalah pembuatan *flowchart* program untuk arduino nano *transmitter*, arduino nano *receiver* dan processing IDE yang alur programnya dijelaskan sebagai berikut.

a. Flowchart Arduino Nano *Transmitter*

Proses kerja program pada arduino *transmitter* merupakan tahap untuk memperoleh data dari sensor-sensor yang selanjutnya dikirim ke sistem arduino *receiver* secara *real time* dengan *update* per 200 ms. Kerja program arduino nano *transmitter* diawali dengan persiapan mengidentifikasi *library*, variabel, *input*, *output*, COM *port* dan *baud rate*. Selanjutnya melakukan proses pembacaan parameter tegangan dan arus (variabel V dan I) tiap 100 ms. Kemudian melakukan proses pembacaan parameter sudut *Yaw*, *Pitch*, *Roll* dan *Altitude* (variabel Y, P, R

dan Alt) tiap 100 ms. Dan melakukan proses pembacaan parameter kecepatan motor 1 - 4 tiap 200 ms. Dari hasil pembacaan sensor-sensor, kemudian dicetak atau di *serial print* dengan format string # V; I; Y; P; R; Alt; R1; R2; R3; R4 yang di *update* per 200 ms. Untuk alur lebih jelasnya ditunjukkan *flowchart* arduino nano *transmitter* pada Gambar 3.5.

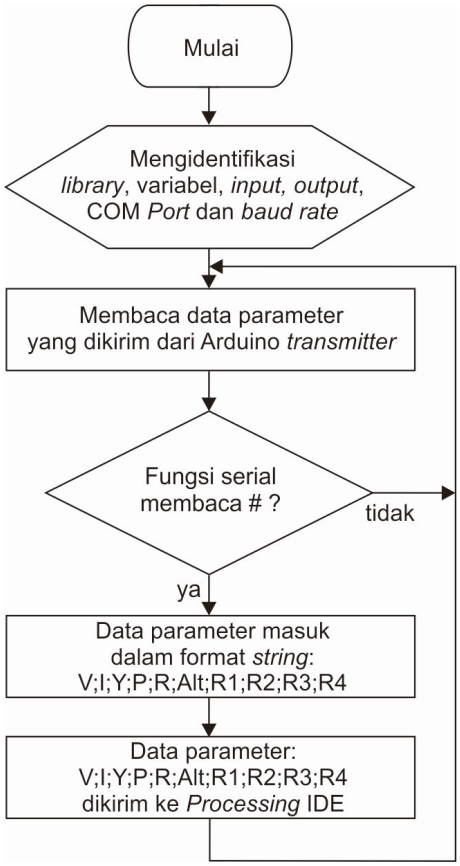


Gambar 3.5 *Flowchart* Arduino Nano *Transmitter*

b. Flowchart Ardunio Nano *Receiver*

Proses kerja program pada arduino *receiver* merupakan tahap untuk membaca data yang dikirim dari arduino *transmitter*. Kerja program arduino *receiver* diawali dengan persiapan mengidentifikasi *library*, variabel, *input*, *output*, COM *port* dan *baud rate*. Selanjutnya melakukan proses pembacaan parameter yang dikirim oleh arduino *transmitter* yang dikirim dalam format string # V; I; Y; P; R; Alt; R1; R2;

R3; R4. Metode yang digunakan dalam proses pembacaan string, dalam hal ini fungsi serial harus membaca tanda # (*hastag*), sehingga data string yang masuk menjadi V; I; Y; P; R; Alt; R1; R2; R3; R4. Kemudian data dalam format string V; I; Y; P; R; Alt; R1; R2; R3; R4 dikirim ke aplikasi processing. Dengan demikian, arduino nano *receiver* digunakan sebagai perantara antara XBee dengan komputer dan processing. Untuk alur lebih jelasnya ditunjukkan *flowchart* arduino nano *receiver* pada Gambar 3.6.

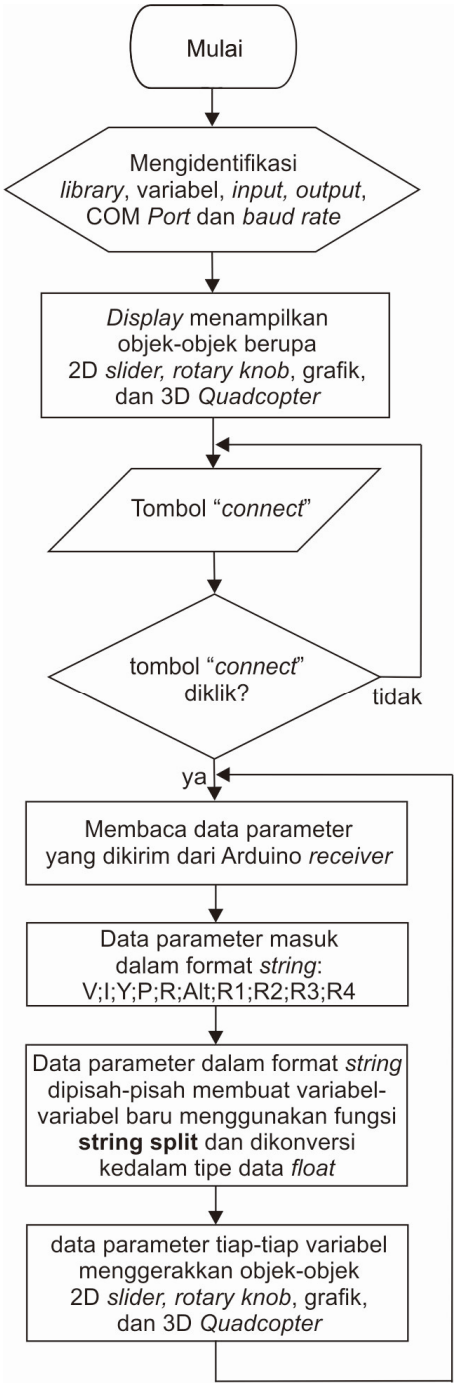


Gambar 3.6 *Flowchart* Arduino Nano *Receiver*

c. Flowchart Processing IDE

Proses kerja program pada processing merupakan tahap untuk membaca data yang dikirim dari arduino *receiver*. Kerja program processing diawali dengan mengklik tombol *running* pada aplikasi processing (GCS), lalu processing

melakukan persiapan mengidentifikasi *library*, variabel, *input*, *output*, COM *port* dan *baud rate*. Setelah itu, aplikasi processing menampilkan objek-objek berupa *slider*, *rotary knob*, grafik, dan 3D *quadcopter*, namun objek-objek masih belum dapat bergerak. Pada aplikasi processing (GCS), terdapat tombol koneksi untuk memulai penerimaan data. Jika tombol digeser ke kiri, maka selanjutnya processing melakukan proses pembacaan parameter yang dikirim dari arduino *receiver* berupa format string V; I; Y; P; R; Alt; R1; R2; R3; R4. Dari format string V; I; Y; P; R; Alt; R1; R2; R3; R4 yang diterima, kemudian data tersebut dipisah-pisah kembali menjadi masing-masing satu parameter dan ditempatkan ke variabel baru dari processing. Kemudian variabel tersebut dimasukkan kedalam fungsi objek-objek, sehingga objek-objek dapat bergerak dengan fungsinya masing-masing. Untuk alur lebih jelasnya ditunjukkan *flowchart* processing pada Gambar 3.7.



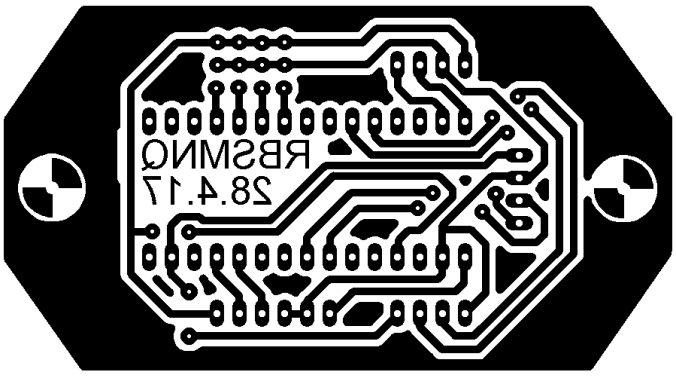
Gambar 3.7 *Flowchart* Processing IDE

3.3.3 Tahap Desain Produk

3.3.3.1 Desain *Layout Shield* Modul Sensor pada *Quadcopter*

Layout shield modul sensor berfungsi untuk menghubungkan *pin input output* antara arduino nano dengan modul-modul sensor yang berada pada *quadcopter*.

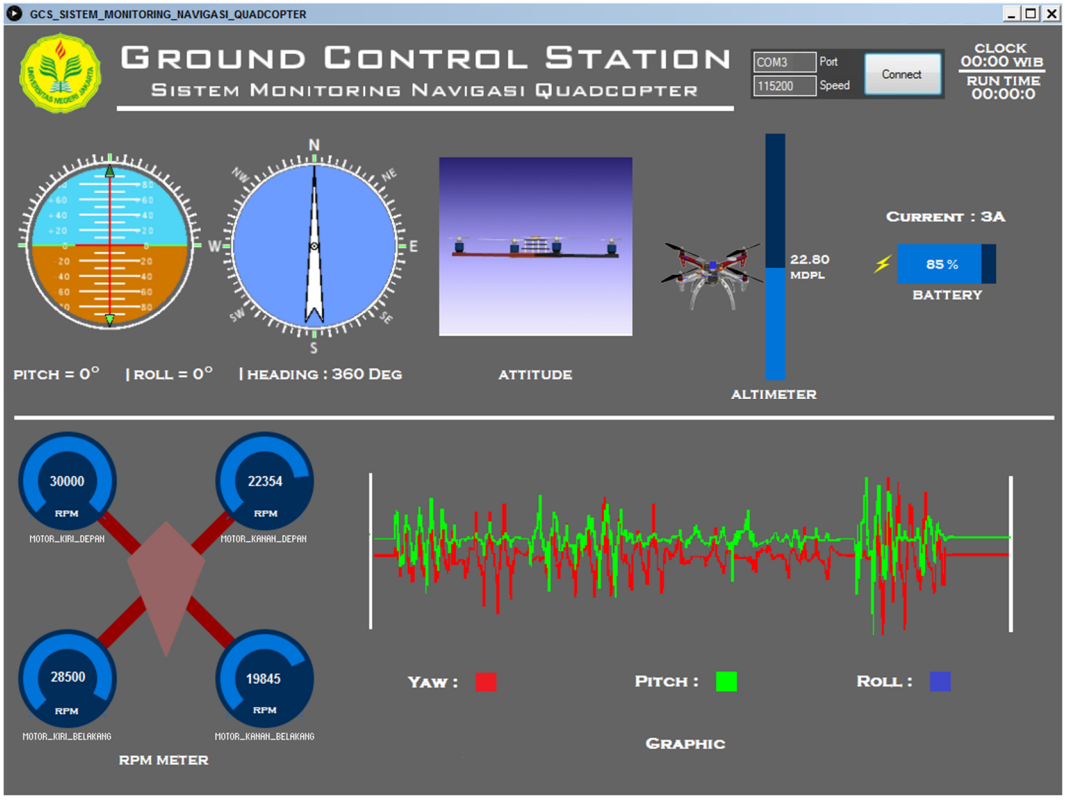
Dari modul-modul sensor ke *shield* dihubungkan menggunakan kabel *jumper* yang sudah disesuaikan panjangnya. *Layout shield* ini di desain menggunakan aplikasi Eagle 6.4.0. Gambar 3.8 adalah *layout shield* yang telah dibuat.



Gambar 3.8 Desain *Layout Shield* Modul-modul Sensor

3.3.3.2 Desain Tampilan Antarmuka *Ground Control Station*

Dalam pembuatan sistem *monitoring* navigasi *quadcopter*, diperlukan *Ground Control Station* (GCS) atau *Human Machine Interface* (HMI), yaitu sebuah tampilan antarmuka visual untuk memudahkan pengguna dalam membaca data pergerakan *quadcopter* ketika terbang diudara. Berikut gambar rancangan atau desain *Ground Control Station* yang ditunjukkan pada Gambar 3.9.



Gambar 3.9 Desain *Ground Control Station*

3.4 Teknik Pengumpulan Data

Teknik pengumpulan data yang digunakan dalam penelitian ini adalah observasi dan pengukuran, meliputi pencarian literatur atau *datasheet*, melakukan percobaan lapangan, melakukan pengukuran disetiap objek yang akan diuji, mencatat setiap hasil pengukuran objek yang diamati yang kemudian hasil daripada data tersebut diimplementasikan dan disajikan dalam bentuk tabel dan grafik, serta menjelaskannya secara naratif.

3.5 Teknik Analisi Data

Teknik analisis data merupakan kriteria pengujian yang dilakukan untuk mendapatkan data yang diperlukan pada keseluruhan sistem yang dibuat. Kriteria pengujian dilakukan untuk menyatakan bahwa sistem yang telah dibuat dapat

dinyatakan berhasil atau gagal. Berikut tabel-tabel pengujian pada penelitian rancang bangun sistem *monitoring* navigasi *quadcopter*.

3.5.1 Pengujian Koneksi GCS dengan *Quadcopter*

Pengujian koneksi antarmuka GCS (*Ground Control Station*) dengan *quadcopter* adalah bertujuan untuk menjelaskan cara pengkoneksian antarmuka GCS dengan *quadcopter* dan membuktikan bahwa koneksi antara antarmuka GCS dengan *quadcopter* berjalan dengan lancar. Hasil pengujian ditandai dengan "Koneksi terhubung atau Koneksi tidak terhubung". Pengujian koneksi GCS dengan *quadcopter* ditunjukkan pada Tabel 3.7.

Tabel 3.7 Pengujian Koneksi GCS dengan *Quadcopter*

Percobaan Ke-	Kondisi *Toggle* pada GCS	Kriteria Pengujian	Hasil Pengujian
1.	*Toggle* pada GCS digeser ke kanan	Tidak Menampilkan pemberitahuan COM *Port* dan *Baud rate* yang digunakan	...
2.	*Toggle* pada GCS digeser ke kiri	Menampilkan pemberitahuan COM *Port* dan *Baud rate* yang digunakan, dan menampilkan data-data dari *Quadcopter* serta menggerakkan objek-objek GCS	...

3.5.2 Pengujian Jarak Jangkauan Komunikasi

Pengujian jarak jangkauan komunikasi adalah proses untuk mengukur sejauh berapakah jarak antara sistem *monitoring quadcopter* dengan sistem GCS dapat berkomunikasi searah. Yang dimaksud berkomunikasi searah dalam hal ini adalah sebuah perangkat *transmitter* dapat terhubung dan mengirim data ke perangkat *receiver*. Proses pengukuran dilakukan dengan bertahap, yaitu dengan jarak ukur

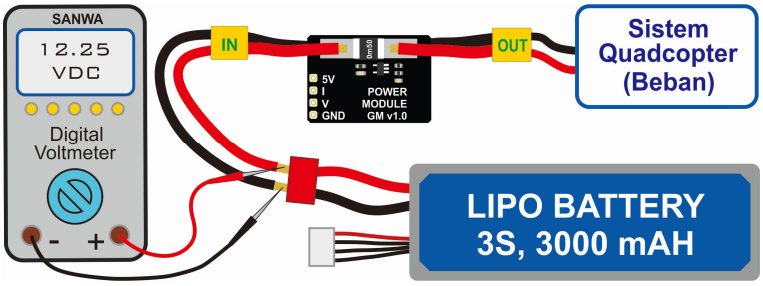
10 meter, 20 meter, 30 meter, 40 meter, dan 50 meter. Pada kolom hasil pengujian diisi dengan “Terhubung” atau “Tidak Terhubung”. Kemudian indikator pengujian ditandai dengan bergerak atau tidaknya objek-objek pada antarmuka GCS. Keberhasilan pengujian jarak jangkauan adalah jika sistem dapat berkomunikasi minimal 20 meter. Pengujian jarak jangkauan komunikasi ditunjukkan pada Tabel 3.8.

Tabel 3.8 Pengujian Jarak Jangkauan Komunikasi

No.	Jarak pengujian	Hasil Pengujian	Indikator Pengujian
1.	10 meter	...	...
2.	20 meter	...	...
3.	30 meter	...	...
4.	40 meter	...	...
5.	50 meter	...	...

3.5.3 Pengujian Tegangan Baterai *Quadcopter*

Pengujian Tegangan baterai adalah proses untuk mengukur tegangan baterai *lithium polymer* pada sistem *quadcopter* yang dinyatakan dalam satuan volt menggunakan alat ukur multitester digital merk Sanwa dengan 3DR *power module*. Pengukuran tegangan baterai dilakukan ketika baterai sedang terhubung ke beban atau sistem *quadcopter*. Proses pengukuran dilakukan masing-masing 6 kali pengukuran dan tingkat keakurasian dinyatakan dalam bentuk persentase *error* antara hasil pengukuran menggunakan multitester digital dengan mengggunakan 3DR *power module*. Metode pengukuran tegangan diperlihatkan pada Gambar 3.10 dan tabel pengujian tegangan baterai ditunjukkan pada Tabel 3.9.



Gambar 3.10 Metode Pengujian Tegangan Baterai *Quadcopter*

Kemudian tegangan yang dihasilkan dari proses pengukuran, peneliti mengkonversi menjadi kapasitas baterai yang dinyatakan dalam persen untuk pemrograman di mikrokontroler arduino yang ditampilkan oleh antarmuka GCS yang telah dibuat dengan menggunakan rumus :

$$Kapasitas\ baterai = \frac{tegangan\ terbaca - tegangan\ minimal}{tegangan\ maksimal - tegangan\ minimal}\ x\ 100\%$$

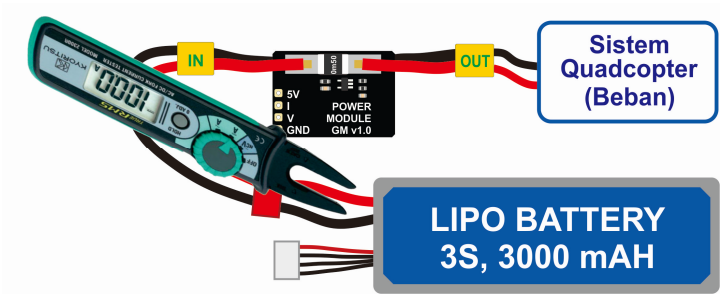
Keterangan :
Kapasitas baterai : nilai akhir (dinyatakan dalam persen)
Tegangan terbaca : nilai tegangan yang terbaca dari hasil pengukuran sensor tegangan (V)
Tegangan minimal : nilai tegangan baterai *lithium polymer* ketika habis (V).
Tegangan maksimal : nilai tegangan baterai *lithium polymer* ketika penuh (V).

Tabel 3.9 Pengujian Tegangan Baterai *Quadcopter*

No.	Pengukuran Tegangan menggunakan Volt-meter (Sanwa)	Pembacaan Tegangan menggunakan 3DR *Power Modul* pada *Serial monitor*	*Error* (%) Pembacaan antara Volt-meter dengan 3DR *Power Modul*	Pembacaan kapasitas baterai pada GCS dalam persentase (%)
1.	...	...	...	...
2.	...	...	...	...
3.	...	...	...	...
4.	...	...	...	...
5.	...	...	...	...
6.	...	...	...	...

3.5.4 Pengujian Arus Baterai *Quadcopter*

Pengujian Arus baterai adalah proses untuk mengukur arus baterai *lithium polymer* pada sistem *quadcopter* yang dinyatakan dalam satuan ampere menggunakan alat ukur *Clamp Multimeter* (Tang Ampere) model APPA A16R (*True* RMS) dengan 3DR *power module* yang ditampilkan oleh antarmuka GCS yang telah dibuat. Pengukuran arus baterai dilakukan ketika baterai sedang terhubung ke beban atau sistem *quadcopter*. Proses pengukuran dilakukan masing-masing 6 kali pengukuran dan tingkat keakurasian dinyatakan dalam bentuk persentase *error* antara hasil pengukuran menggunakan Tang Ampere dengan mengggunakan 3DR *power module* yang ditampilkan menggunakan GCS. Metode pengukuran menggunakan *Clamp multimeter* atau Tang Ampere diperlihatkan pada Gambar 3.11 dan tabel pengujian arus baterai ditunjukkan pada Tabel 3.10.



Gambar 3.11 Metode Pengukuran Arus Baterai *Quadcopter*

Tabel 3.10 Pengujian Arus Baterai *Quadcopter*

No.	Pengukuran Arus Menggunakan Tang Ampere (APPA A16R)	Pembacaan Arus menggunakan 3DR *Power Modul* dan GCS	*Error* (%)
1.	...	...	...
2.	...	...	...
3.	...	...	...
4.	...	...	...
5.	...	...	...
6.	...	...	...

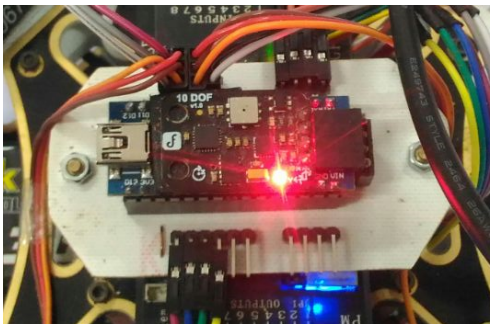
berukuran 2,5 mm untuk spiser penyangga *shield*. Hasil rancangan *shield* untuk modul-modul sensor diperlihatkan pada Gambar 4.1 dan 4.2.

Gambar 4.1 *Layout Shield* untuk Modul-modul Sensor

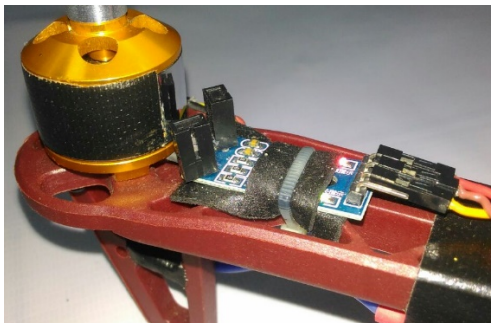
Gambar 4.2 *Shield* Tampak Atas

4.1.1.2 Hasil Rancangan Penempatan Modul-modul *Quadcopter*

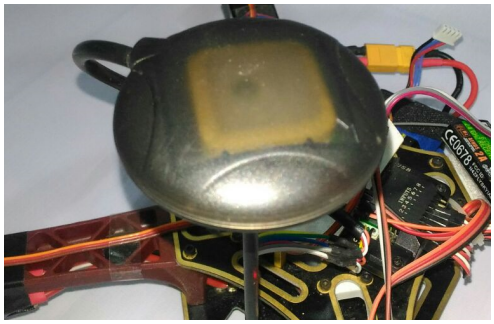
Modul-modul dan sensor-sensor ditempatkan di *frame quadcopter* dengan mempertimbangkan keseimbangan, agar *quadcopter* tetap stabil ketika terbang diudara. Gambar 4.3 – 4.9 menampilkan penempatan masing-masing modul dan sensor.



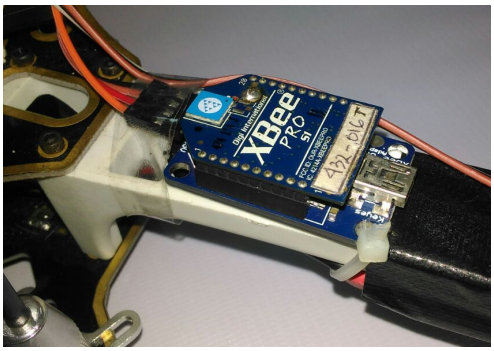
Gambar 4.3 Penempatan *Shield,* Arduino Nano dan IMU



Gambar 4.4 Penempatan Sensor RPM Motor

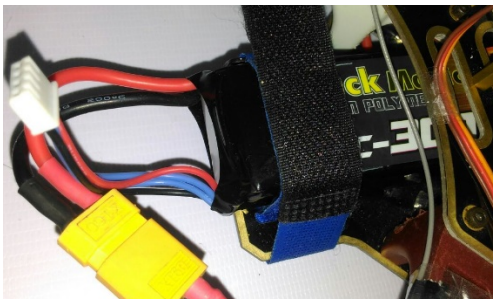


Gambar 4.5 Penempatan GPS

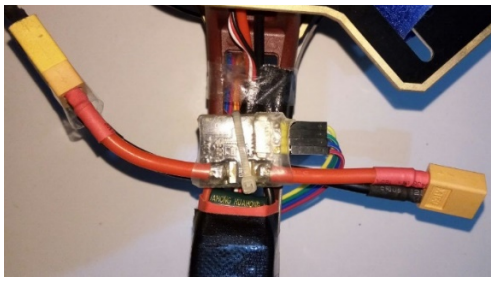


Gambar 4.6 Penempatan XBee *Pro* S1

Gambar 4.7 Penempatan *Radio Receiver*



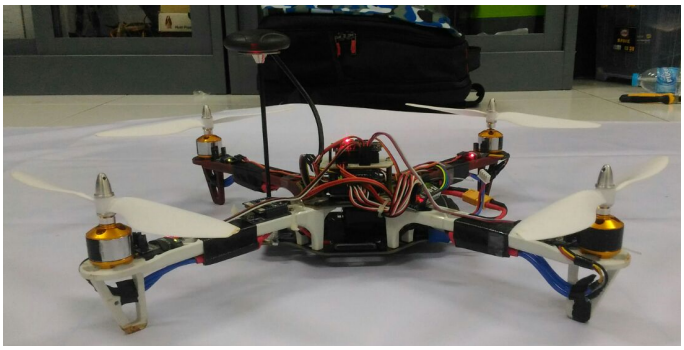
Gambar 4.8 Penempatan Baterai *Lithium Polymer*



Gambar 4.9 Penempatan *Power Module*

4.1.1.3 Hasil Rakitan dan *Wiring Quadcopter*

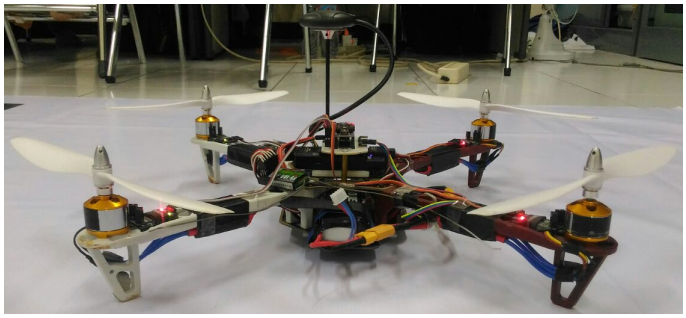
Dalam pembuatan alat, *wiring* (pengkabelan) harus diperhatikan untuk menghindari kegagalan sistem, baik dari jenis kabel yang digunakan, kesesuaian antara luas penampang kabel dengan arus yang akan dilewati, penempatan jalur kabel agar tidak mengganggu kerja sistem dan sebagainya. Gambar *wiring* (pengkabelan) sistem diperlihatkan pada Gambar 4.10 – 4.14.



Gambar 4.10 *Quadcopter* Tampak Depan

Gambar 4.11 *Quadcopter* Tampak Belakang

Gambar 4.12 *Quadcopter* Tampak Sisi Kanan

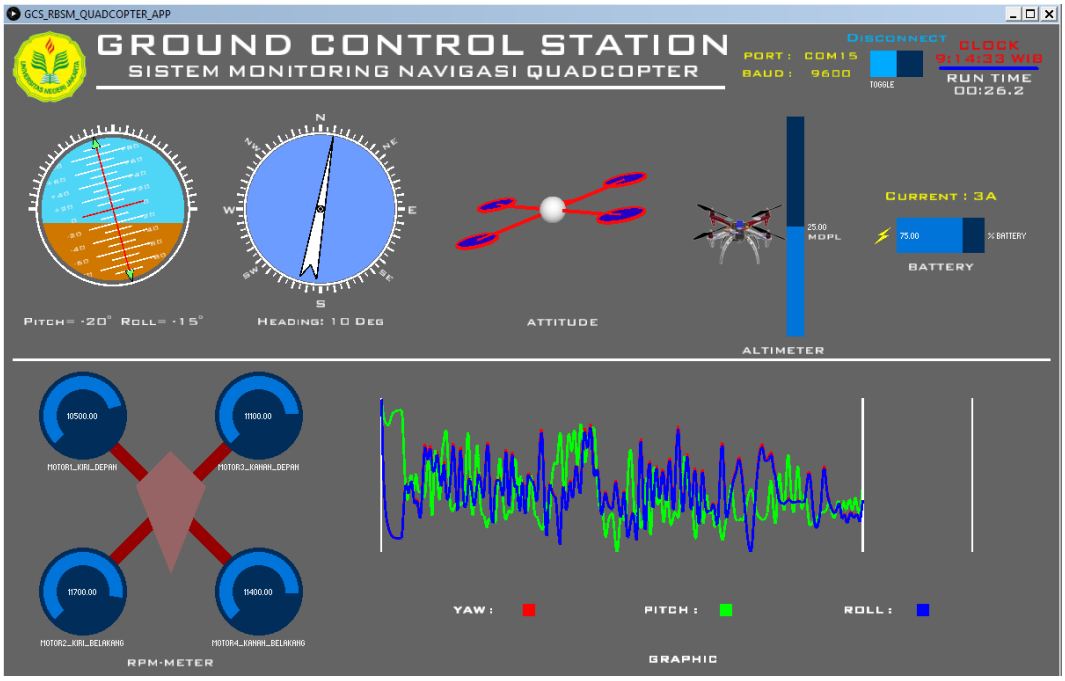


Gambar 4.13 *Quadcopter* Tampak Sisi Kiri

Gambar 4.14 *Quadcopter* Tampak Atas

4.1.2 Hasil Rancangan *Software* Sistem *Monitoring* (GCS)

Antarmuka GCS yang telah dibuat memiliki panjang dan lebar 1200 x 750 *pixel*. Pada tampilan antarmuka tersebut berisi objek-objek visual berupa 2 dimensi dan 3 dimensi, diantaranya adalah logo institusi, judul penelitian, pemberitahuan COM *port* dan *Baud rate* yang digunakan, *Toggle Connect* atau *Disconnect*, *Clock* atau jam, *Run time* (pemberitahuan telah berapa lama aplikasi sedang berjalan), objek *Pitch* dan *Roll*, objek *Heading*, *Attitude* (sikap roket 3D), grafik pergerakan *quadcopter*, *rotary knob* RPM Motor, *slider Altitude* (menampilkan ketinggian *quadcopter*), *slider Battery* dan pemberitahuan arus. Gambar 4.15 memperlihatkan tampilan GCS yang digunakan untuk me-*monitoring* pergerakan *quadcopter*.



Gambar 4.15 Tampilan Antarmuka GCS

Keterangan Objek-objek Antarmuka :

14. ***COM Port*** : untuk menampilkan pemberitahuan *port* serial berapa yang terhubung oleh komputer, yang menandakan bahwa perangkat arduino telah terhubung dengan komputer atau belum.

15. ***Speed (Baud rate)*** : untuk menampilkan pemberitahuan *baud rate* berapa yang terbaca oleh komputer untuk berkomunikasi.
16. **Tombol *Connect*** : untuk memulai menerima data serial secara *real time* dari perangkat sistem *monitoring* pada *quadcopter*.
17. ***Manual Intruction*** : digunakan untuk membantu pengguna dalam mengoperasikan *Ground Control Station*.
18. ***Clock*** : untuk menampilkan waktu Indonesia bagian Barat.
19. ***Run Time*** : untuk menampilkan berapa lama pengguna menampilkan *Ground Control Station.*
20. ***Rotary Pitch-Roll*** : untuk menampilkan nilai kemiringan *quadcopter* yang dinyatakan dalam satuan derajat.
21. ***Rotary Heading*** : untuk menunjukkan arah hadap *quadcopter* (kompas).
22. **3D *shape Attitude*** : untuk memperlihatkan sikap *quadcopter* ketika terbang diudara yang dinyatakan dalam bentuk gambar 3 dimensi.
23. ***Graphic*** : untuk menampilkan data kestabilan *yaw, pitch* dan *roll* dalam bentuk grafik.
24. **RPM meter *Slider*** : untuk menampilkan data kecepatan rotasi motor dalam satuan menit.
25. ***Altimeter Slider*** : untuk mengetahui berapa ketinggian *quadcopter* yang dinyatakan dalam satuan mdpl (meter dari permukaan laut).
26. ***Battery Slider*** : untuk mengetahui keadaan kapasitas baterai yang dinyatakan dalam satuan persen.
27. ***Current Text*** : untuk menampilkan nilai arus baterai *lithium polymer* yang terukur.

4.2 Kelayakan Produk (Teoritik dan Empiris)

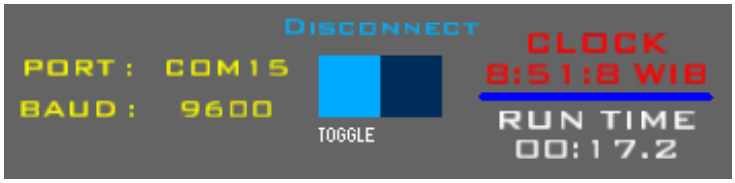
Pada penelitian ini kelayakan dari produk yang dikembangkan dapat diketahui dari proses dan hasil rancangan yang di ajukan. Dalam proses pengujian produk, metode yang digunakan adalah melakukan perbandingan nilai parameter yang didapat oleh hasil pengukuran produk yang dibuat oleh peneliti dengan hasil pengukuran menggunakan alat ukur yang telah berstandar. Proses pengujian diawali dengan mempersiapkan produk yang siap untuk diuji, melakukan analisis kebutuhan dan pengadaan alat ukur yang telah berstandar, dan melakukan proses pengujian serta hasil dari pengujian di catat dan disimpulkan.

4.3 Efektifitas Produk (Melalui Uji Coba)

4.3.1 Hasil Pengujian Koneksi GCS dengan *Quadcopter*

Pengujian koneksi antarmuka GCS (*Ground Control Station*) dengan *quadcopter* adalah bertujuan untuk menjelaskan cara pengkoneksian antarmuka GCS dengan *quadcopter* dan membuktikan bahwa koneksi antara antarmuka GCS dengan *quadcopter* terhubung atau tidak. Hasil terkoneksi atau tidak terkoneksinya GCS dengan *quadcopter*, diperlihatkan pada Gambar 4.16 dan 4.17 dan dicatat pada Tabel 4.1.

Gambar 4.16 Tampilan GCS Tidak Terkoneksi dengan *Quadcopter*



Gambar 4.17 Tampilan GCS Terkoneksi dengan *Quadcopter*

Tabel 4.1 Hasil Pengujian Koneksi GCS dengan *Quadcopter*

Percobaan Ke-	Kondisi *Toggle* pada GCS	Kriteria Pengujian	Hasil Pengujian
1.	*Toggle* pada GCS digeser ke kanan	Tidak Menampilkan pemberitahuan COM *Port* dan *Baud rate* yang digunakan	Koneksi tidak terhubung
2.	*Toggle* pada GCS digeser ke kiri	Menampilkan pemberitahuan COM *Port* dan *Baud rate* yang digunakan, dan menampilkan data-data dari *Quadcopter* serta menggerakkan objek-objek GCS	Koneksi terhubung

Dari tabel 4.1 dapat disimpulkan bahwa pengkoneksian antara GCS dengan *quadcopter* memiliki indikator keberhasilan yang ditampilkan oleh antarmuka GCS. Jika pengkoneksian berhasil, maka antarmuka GCS akan menampilkan data-data dan menggerakkan objek-objek GCS. Berikut ini adalah tahap-tahap pengkoneksian antarmuka GCS dengan *quadcopter* :

1. Sambungkan konektor *male* XT60 *quadcopter* ke konektor *female* XT60 baterai *lithium polymer*.
2. Aktifkan *switch power radio control* dengan menggeser *switch* keatas dan tarik *throttle radio control* ke pojok kanan bawah dan tahan selama 5 detik, lalu posisikan *throttle* ke bawah tengah, kemudian gerakkan ke atas perlahan-lahan hingga motor *brushless* berputar.
3. Hubungkan perangkat penerima (arduino nano dan XBee penerima) ke komputer melalui kabel USB dan pastikan bahwa *port* arduino terbaca oleh komputer.
4. Pastikan *baud rate* arduino nano, XBee dan program processing semuanya sama, yaitu 9600 bps.

5. Buka program antarmuka GCS pada komputer.
6. Aktifkan *toggle* dengan cara menggeser *toggle* ke kiri, maka akan tampil nilai COM *port* dan *baud rate* yang menandakan bahwa antarmuka GCS telah terhubung dengan *quadcopter* serta menerima data-data dan menggerakkan objek-objek GCS.

4.3.2 Hasil Pengujian Jarak Jangkauan Komunikasi

Pengujian jarak jangkauan komunikasi dilakukan dengan mengaktifkan sistem *monitoring* yang ada pada *quadcopter* dan yang ada pada GCS. Kemudian sistem GCS tetap berada di tempat, dan sistem *monitoring* pada *quadcopter* dibawa berjalan secara bertahap sesuai jarak ukur yang telah ditentukan hingga komunikasi diantara keduanya terputus. Berdasarkan pengujian jarak jangkauan komunikasi yang telah dilakukan, pada jarak ukur 10 meter, 20 meter, 30 meter dan 40 meter, komunikasi data masih tetap baik, namun pada jarak 50 meter komunikasi antar keduanya terputus. Hasil pengujian dicatat pada Tabel 4.2.

Tabel 4.2 Hasil Pengujian Jarak Jangkauan Komunikasi

No.	Jarak pengujian	Hasil Pengujian	Indikator Pengujian
1.	10 meter	Terhubung	Objek-objek GCS bergerak
2.	20 meter	Terhubung	Objek-objek GCS bergerak
3.	30 meter	Terhubung	Objek-objek GCS bergerak
4.	40 meter	Terhubung	Objek-objek GCS bergerak
5.	50 meter	Tidak Terhubung	Objek-objek GCS tidak bergerak

4.3.3 Hasil Pengujian Tegangan Baterai *Quadcopter*

Pengujian tegangan baterai dilakukan dengan membandingkan pengukuran secara bersamaan antara menggunakan alat ukur volt meter yang telah memenuhi standar dengan menggunakan GCS yang peneliti buat. Berdasarkan pengamatan

hasil pengukuran, nilai rata-rata *error* sebesar 0,215%, nilai rata-rata tersebut cukup baik dalam kestabilan sistem. Hasil pengukuran dicatat pada Tabel 4.3.

Tabel 4.3 Hasil Pengujian Tegangan Baterai *Quadcopter*

No.	Pengukuran Tegangan Menggunakan Volt-meter (Volt)	Pembacaan Tegangan menggunakan 3DR *Power Modul* dan GCS (Volt)	*Error* (%)
1.	12,52 V	12,54 V	0,15%
2.	12,43 V	12,41 V	0,16%
3.	12,27 V	12,30 V	0,24%
4.	12,04 V	12,01 V	0,24 %
5.	11,89 V	11,85 V	0,33%
6.	11,62 V	11,64 V	0,17%

4.3.4 Hasil Pengujian Arus Baterai *Quadcopter*

Pengujian arus baterai dilakukan dengan membandingkan pengukuran secara bersamaan antara menggunakan alat ukur tang ampere yang telah memenuhi standar dengan menggunakan GCS yang peneliti buat. Pengambilan data arus baterai mnggunakan tang ampere dan *power module* dilakukan dengan membandingkan data secara bersamaan, sehingga didapat waktu yang tepat dalam pengambilan datanya. Berdasarkan pengamatan hasil pengukuran, selisih nilai arus paling tinggi sebesar 0.08A, dan nilai rata-rata selisih arus sebesar 0,046. Kemudian nilai rata-rata *error* arus sebesar 2,65%. Dengan demikian, pembacaan nilai arus oleh *power module* cukup stabil. Hasil pengukuran dicatat pada Tabel 4.4.

Tabel 4.4 Hasil Pengujian Arus Baterai *Quadcopter*

No.	Pengukuran Arus Menggunakan Tang Ampere (APPA A16R)	Pembacaan Arus menggunakan 3DR *Power Modul* dan GCS	*Error* (%)
1.	0,53 A	0,57 A	7.54 %
2.	2,66 A	2,63 A	1,12 %
3.	4,50 A	4,42 A	1,77 %
4.	3,04 A	2,98 A	1,97 %

5.	2,73 A	2,69 A	1,46 %
6.	1,47 A	1,50 A	2,04 %

4.3.5 Hasil Pengujian Sudut *Yaw Quadcopter*

Pengujian sudut *yaw* dilakukan dengan memutar *quadcopter* pada porosnya diatas busur lingkaran, lalu dilakukan pengukuran dimulai dari sudut ukur derajat yang terendah hingga sudut ukur derajat yang tertinggi. Kemudian, hasil pengukuran menggunakan busur dibandingkan dengan nilai yang terbaca oleh sensor Magnetometer yang ditampilkan pada GCS. Berdasarkan pengamatan hasil pengukuran, pada sudut ukur 45 hingga 135 derajat mengalami peningkatan selisih yang cukup signifikan yaitu 10 dan 15 derajat, dan pada sudut ukur 180 hingga 225 derajat mengalami peningkatan selisih yang sangat signifikan yaitu sebesar 35 derajat, namun pada sudut ukur 225 hingga 360 derajat mengalami penurunan selisih 5 dan 10 derajat. Sehingga, nilai rata-rata selisih sebesar 16,87 derajat. Dapat disimpulkan, bahwa pembacaan sensor kompas tidak presisi terutama pada sudut 180 ke sudut 225 derajat. Namun demikian, data tetap terbaca dengan baik oleh antarmuka GCS yang telah dibuat. Hasil pengukuran dicatat pada Tabel 4.5.

Tabel 4.5 Hasil Pengujian Sudut *Yaw Quadcopter*

No.	Pengukuran sudut *Yaw* menggunakan Busur	Pembacaan sudut *Yaw* Menggunakan Magnetometer HMC5883L dan GCS	Selisih
1.	45 derajat	35	-10
2.	90 derajat	70	-20
3.	135 derajat	100	-35
4.	180 derajat	165	-15
5.	225 derajat	250	25
6.	270 derajat	290	20
7.	315 derajat	325	10
8.	0 atau 360 derajat	0	0

4.3.6 Hasil Pengujian Sudut *Pitch Quadcopter*

Pengujian sudut *pitch* dilakukan dengan mekanisme pengukuran menggunakan alat bantu aluminium siku sepanjang 50 cm yang terpasang sesuai garis sumbu Y (*pitch*) dan sama rata (lurus) dibawah PDB (*Power Distribution Battery*) *quadcopter*. Pada ujung aluminium siku di pasang sebuah as horizontal sebagai poros yang dipasang atau dimasukkan pada lubang titik tengah busur. Sedangkan, busur di letakkan dengan berdiri tegak pada alas yang rata (tidak miring). Selanjutnya dilakukan proses pengukuran secara bertahap dengan cara menggerakkan *quadcopter* dari poros busur ke nilai yang tertera pada tepi busur yang kemudian dibandingkan dengan hasil pengukuran menggunakan Akselerometer dan Giroskop yang ditampilkan pada GCS. Berdasarkan pengujian sudut *pitch* yang telah dilakukan, akurasi sudut *pitch* sangat baik dengan nilai rata-rata selisih sebesar 0,28 derajat. Hasil pengukuran dicatat pada Tabel 4.6.

Tabel 4.6 Hasil Pengujian Sudut *Pitch Quadcopter*

No.	Pengukuran sudut *Pitch* menggunakan Busur	Pembacaan sudut *Pitch* Menggunakan Akselerometer dan Girometer dan GCS	Selisih
1.	-75 derajat	-76	1
2.	-45 derajat	-45	0
3.	-15 derajat	-15	0
4.	0 derajat	0	0
5.	15 derajat	15	0
6.	45 derajat	45	0
7.	75 derajat	74	-1

4.3.7 Hasil Pengujian Sudut *Roll Quadcopter*

Pengujian sudut *roll* dilakukan dengan mekanisme pengukuran menggunakan alat bantu aluminium siku sepanjang 50 cm yang terpasang sesuai garis sumbu X (*roll*) dan sama rata (lurus) dibawah PDB (*Power Distribution Battery*) *quadcopter*.

Pada ujung aluminium siku di pasang sebuah as horizontal sebagai poros yang dipasang atau dimasukkan pada lubang titik tengah busur. Sedangkan, busur di letakkan dengan berdiri tegak pada alas yang rata (tidak miring). Selanjutnya dilakukan proses pengukuran secara bertahap dengan cara menggerakkan *quadcopter* dari poros busur ke nilai yang tertera pada tepi busur yang kemudian dibandingkan dengan hasil pengukuran menggunakan Akselerometer dan Giroskop yang ditampilkan pada GCS. Berdasarkan pengujian sudut *roll* yang telah dilakukan, akurasi sudut *roll* sangat baik dengan nilai rata-rata selisih sebesar 0,57 derajat. namun pada sudut ukur menggunakan busur ke 135 mengalami selisih minus 2 terhadap hasil pembacaan IMU. Hasil pengukuran dicatat pada Tabel 4.7.

Tabel 4.7 Hasil Pengujian Sudut *Roll Quadcopter*

No.	Pengukuran sudut *Roll* menggunakan Busur	Pembacaan sudut *Roll* Menggunakan Akselerometer dan Girometer dan GCS	Selisih
1.	-135 derajat	-136	1
2.	-90 derajat	-90	0
3.	-45 derajat	-45	0
4.	0 derajat	0	0
5.	45 derajat	45	0
6.	90 derajat	89	-1
7.	135 derajat	133	-2

4.3.8 Hasil Pengujian *Altitude Quadcopter*

Pengujian *Altitud*e dilakukan dengan mekanisme pengukuran menggunakan alat ukur berupa meteran dan tali sebagai alat bantu. Pengukuran *altitude* menggunakan ketinggian ukur 0, 3, 5, 10, 12 dan 15 meter dari bumi. Dikarenakan satuan hasil pengukuran menggunakan barometer berbeda dengan hasil pengukuran menggunakan meteran dan tali, maka untuk pengisian nilai pada kolom selisih dirumuskan sebagai berikut :

$$Selisih \ = p.m.meteran - (p.m. \ barometer - nilai\ ketinggian\ bumi)$$

Keterangan :
h.p.m meteran : hasil pengukuran menggunakan meteran
h.p.m barometer : hasil pengukuran menggunakan barometer
nilai ketinggian bumi : nilai rata-rata ketinggian bumi di DKI Jakarta dari permukaan laut = 7 meter.

Berdasarkan hasil pengujian yang telah dilakukan, data pengukuran yang dihasilkan kurang akurat, sehingga besarnya *error*, namun dapat memberikan perubahan nilai pembacaan. Tabel pengujian *altitude quadcopter* ditunjukkan pada Tabel 4.8.

Tabel 4.8 Pengujian *Altitude Quadcopter*

No.	Pengukuran *Altitude* menggunakan Meteran dan tali (meter)	Pembacaan *Altitude* menggunakan IMU (Barometer) dan GCS (mdpl)	Selisih (meter)
1.	0 meter	7,05 mdpl	0 - (7,05 – 7) = -0,05
2.	3 meter	9,76 mdpl	3 - (9,76 – 7) = 0,24
3.	5 meter	10,94 mdpl	5 - (10,94 – 7) = 1,06
4.	10 meter	14,08 mdpl	10 - (13,08 – 7) = 1,92
5.	12 meter	17,52 mdpl	12 - (18,52 – 7) = 2,48
6.	15 meter	20,23 mdpl	15 - (22,23 – 7) = 3,77

4.3.9 Hasil Pengujian RPM Motor *Brushless Quadcopter*

Sebelum melakukan pengukuran, pada sebagian sisi stator motor *brushless* ditempel lakban hitam untuk menandai perbedaan pantulan cahaya agar dapat dideteksi oleh laser tachometer dan modul LM393 *Infrared Speed sensor*. Selanjutnya dilakukan pengukuran menggunakan tachometer dengan mengarahkan sinar laser tachometer ke rotor motor *brushless*. Lalu menghidupkan motor *brushless* sambil menyesuaikan kecepatan putaran motor *brushless* hingga nilai RPM motor sesuai dengan nilai sampel yang telah ditentukan. Kemudian hasil pengukuran menggunakan tachometer dibandingkan dengan hasil pengukuran

menggunakan modul LM393 *Infrared Speed Sensor* yang nilainya ditampilkan pada GCS. Berdasarkan hasil pengujian, semua data yang didapat berada dibawah 3,33%, yang berarti masih berada di tingkat ketelitian hasil pembacaan RPM motor b*rushless* yaitu 300 RPM. Hasil pengukuran dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil Pengujian RPM Motor *Brushless Quadcopter*

No.	Pengukuran menggunakan *Tachometer* (DT-2234C+)				Pengukuran menggunakan Sensor RPM dan GCS				*Error* (%)	*Error* (%)	*Error* (%)	*Error* (%)
	RPM MBL 1	RPM MBL 2	RPM MBL 3	RPM MBL 4	RPM MBL 1	RPM MBL 2	RPM MBL 3	RPM MBL 4				
1.	9288	9372	9096	9624	9000	9300	9000	9600	3,10	0,76	1,05	0,24
2.	10628	11228	9852	9372	10500	11100	9600	9300	1,20	1,14	2,55	0,76
3.	11156	10220	10700	10256	11100	10200	10500	10200	0,50	0,19	1,86	0,54
4.	11628	10820	11120	10880	11400	10800	11100	10800	1,96	0,18	0,18	0,73
5.	11180	11592	11276	11580	11100	11400	11000	11400	0,71	1,65	2,44	1,55
6.	11436	11496	11724	11412	11400	11400	11700	11400	0,31	0,83	0,20	0,10

4.4 Pembahasan

Berdasarkan penelitian yang telah dilakukan dan hasil yang telah didapat, kinerja sistem *hardware* dan *software* dapat diketahui kelebihan dan kekurangannya. Berikut ini akan dibahas kinerja sistem *hardware* dan *software* sistem *monitoring* navigasi *quadcopter*.

4.4.1 Kinerja Sistem *Hardware*

Sistem *hardware* diantaranya terdiri dari arduino nano, *power module*, IMU, RPM motor, dan XBee. Penelitian pada prosesnya terdapat banyak kendala, seperti tidak stabilnya data analog yang dikeluarkan oleh *power module*, tidak presisinya data kompas HMC5883L yang terdapat pada modul IMU, dan menurunnya performa arduino nano jika penggunaan fungsi waktu millis tidak tepat untuk mengeksekusi keempat modul yang telah disebutkan. Dalam mengatasi kendala

tersebut, peneliti menggunakan fungsi program perulangan (*for*) untuk mendapatkan nilai rata-rata pada data analog *power module*. Kemudian untuk mengatasi kendala pada arduino nano, peneliti memperlambat *update real time* yang sebelumnya adalah *update per* 100 ms menjadi *update per* 200 ms.

4.4.2 Kinerja Sistem *Software* pada Arduino

Kinerja sistem *software* arduino pada prosesnya banyak masalah yang ditemukan, terutama masalah pewaktuan. Dalam pengiriman data secara *real time*, penggunaan fungsi delay(); pada program harus dihindari, karena jika menggunakan fungsi delay, ketika delay sedang dieksekusi oleh mikrokontroler arduino, maka seluruh aktivitas kerja sistem mikrokontroler akan terhenti selama delay yang ditentukan. Untuk mengatasi masalah yang telah dijelaskan, peneliti menggunakan fungsi millis(); pada program. Dalam penggunaan millis pun pada saat itu masih terdapat kendala, yaitu terkait tidak tepatnya penempatan sintaks pada program dan tidak sinkronnya antara pencetakan (*print*) string oleh millis yang satu dengan pencetakan (*print*) string oleh millis yang lainnya yang membuat pencetakan string keseluruhan tidak teratur seperti format string yang diharapkan. Karena jika dalam pencetakan format string tidak teratur oleh arduino *transmitter*, maka data string tidak dapat diterima oleh arduino receiver akibat tidak sesuainya susunan string. Setelah peneliti melakukan percobaan lebih lanjut, didapat cara penggunaan millis dengan cara mode *switching* atau menjalankan program secara bergantian menggunakan fungsi millis. Alhasil pencetakan string jadi teratur seperti yang diharapkan.

4.4.3 Kinerja Sistem *Software* pada GCS

Dalam pembuatan antarmuka GCS menggunakan *software* Processing IDE secara umum tidak terdapat banyak kendala, walaupun pada awalnya peneliti kesulitan dalam pemrograman komunikasi serial, terutama untuk membaca data yang dikirimkan oleh arduino *receiver* berupa data string yang merupakan kumpulan dari beberapa data atau variabel parameter dari sensor-sensor. Seiring waktu penelitian berjalan, tahap demi tahap peneliti menemukan cara yang tepat yaitu menggunakan metode string split dengan memanfaatkan fungsi array, sehingga dapat membaca data string dan memisahkan kembali data string menjadi variabel-variabel baru yang akan dimasukkan kedalam fungsi program untuk menggerakkan objek-objek antarmuka GCS. Dengan demikian, aplikasi antarmuka GCS dapat diselesaikan seperti yang diharapkan sesuai tujuan penelitian.

4.4.4 Pengujian Sistem Secara Keseluruhan

Pengujian sistem secara keseluruhan telah dilakukan di lapangan terbuka, dengan mengaktifkan sistem pada *quadcopter* dan menghubungkan perangkat penerima ke komputer kemudian membuka aplikasi GCS (*Ground Control Station*) dan mengaktifkan *toggle* koneksi pada GCS tersebut. Saat pengujian sistem keseluruhan ini, data yang dikirim dari *quadcopter* dapat diterima oleh antarmuka GCS dengan baik.

4.4.5 Kelebihan dan Kekurangan Produk

4.4.5.1 Kelebihan Produk

Dari hasil penelitian dan pembahasan, maka alat dan antarmuka GCS yang dibuat memiliki beberapa kelebihan, antara lain :

1. Dalam pengkoneksian antarmuka GCS dengan *quadcopter* tidak perlu melakukan pemilihan COM *port* dan *baud rate*, yakni hanya dengan mengaktifkan *toggle* dengan menggeser ke kiri maka data dapat diterima dan langsung menggerakkan objek-objek GCS.
2. Mempermudah dalam pengendalian *quadcopter* dengan melihat pergerakan *quadcopter* dari antarmuka GCS dalam bentuk visualisasi 2 dimensi dan 3 dimensi.
3. Antarmuka GCS dapat memantau kecepatan putaran Motor *Brushless* yang dinyatakan dalam satuan RPM (*Rotation Per Minutes*).
4. Antarmuka GCS dilengkapi dengan *Clock* (jam) dan *Run time* (lamanya aplikasi berjalan).
5. Antarmuka GCS memiliki tombol yang berisi Instruksi Manual sebagai panduan dalam mengoperasikan Sistem *Monitoring* Navigasi *Quadcopter*.
6. Antarmuka GCS didesain dengan sederhana, elegan dan mudah dioperasikan.

4.4.5.2 Kekurangan Produk

Dari beberapa kelebihan diatas, alat dan GCS yang telah dibuat masih memiliki beberapa kekurangan, antara lain :

1. Jarak maksimal pemantauan *quadcopter* menggunakan GCS hanya berjarak 40 meter, karena tidak menggunakan antena.
2. Tidak presisinya data parameter kompas yang dihasilkan.
3. Kurang stabil dan besarnya *error* data parameter *altitude* atau ketinggian yang dihasilkan.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan peneliti melalui tahap perencanaan, perancangan, realisasi dan pengujian, maka dapat diambil kesimpulan sebagai berikut :

1. Produk sistem *monitoring* navigasi *quadcopter* berbasis arduino nano menggunakan processing IDE telah dirancang dan dapat direalisasikan dan parameter yang dihasilkan oleh sensor-sensor pada *quadcopter* dapat diterima dengan baik oleh sistem GCS (*Ground Control Station*) dengan jarak jangkauan maksimal sejauh 40 meter.
2. Produk sistem *monitoring* navigasi *quadcopter* mampu memproses 10 (sepuluh) data parameter secara *realtime* dengan *update* tiap 200 ms (*milli second*), diantaranya adalah parameter berupa tegangan, arus, *heading*/*yaw*, *pitch*, *roll*, *altitude*, RPM motor 1, RPM motor 2, RPM motor 3 dan RPM motor 4 dengan rentang pengukuran dan hasil pengujian sebagai berikut :
   a) Tegangan : dapat mengukur tegangan hingga 12,6 volt (kapasitas baterai dalam kondisi penuh atau 100%) dan memiliki nilai rata-rata *error* tegangan sebesar 0,215%.
   b) Arus : dapat mengukur arus hingga 10 ampere dan memiliki nilai rata-rata *error* arus sebesar 2,65%.
   c) *heading*/*yaw* : dapat mengukur sudut *yaw* dengan rentang dari 0 hingga 360 derajat, dan memiliki nilai rata-rata selisih sebesar 16,87 derajat dengan selisih tertinggi sebesar 35 derajat.

d) *pitch* : dapat mengukur sudut *pitch* dengan rentang dari -87 derajat hingga 88 derajat dan memiliki nilai rata-rata selisih sebesar 0,28 derajat.

e) *roll* : dapat mengukur sudut *roll* dengan rentang dari -180 derajat hingga 180 derajat dan memiliki nilai rata-rata selisih sebesar 0,57 derajat.

f) *altitude* : dapat mengukur ketinggian dengan rentang dari 0 mdpl hingga 50 mdpl dan memiliki nilai rata-rata selisih sejauh 1,58 meter.

g) RPM motor 1 – 4 : dapat mengukur rpm motor *brushless* hingga kurang lebih 13440 rpm dan masing-masing motor *brushless* memiliki nilai rata-rata *error*, yaitu RPM motor 1 sebesar 1,29%, RPM motor 2 sebesar 0,79%, RPM motor 3 sebesar 1,38%, dan RPM motor 4 sebesar 0,65%.

3. Produk sistem *monitoring* navigasi *quadcopter* dapat membantu pengguna dalam proses pemantauan *quadcopter* ketika sedang dikendalikan, karena pengguna dapat melihat pergerakan *quadcopter* berupa data visual 2 dimensi dan 3 dimensi yang ditampilkan oleh aplikasi GCS (*Ground Control Station*).

5.2 Saran

Dalam penelitian rancang bangun sistem *monitoring* navigasi *quadcopter* berbasis arduino nano menggunakan *software* processing IDE, terdapat beberapa saran yang diharapkan dapat memperbaiki penelitian selanjutnya tentang sistem *monitoring* navigasi *quadcopter* atau UAV, antara lain :

1. Pada XBee perlu ditambahkan antena agar jarak jangkauan komunikasi lebih jauh, dan penggunaan antena yang memiliki kemampuan *Line of Sight* lebih dianjurkan.
2. Untuk menambah kehandalan *quadcopter*, perlu ditambahkan modul GPS dan peta pada antarmuka GCS untuk mengetahui dimana posisi *quadcopter* berada

dan perlu ditambahkan modul kamera untuk mengetahui keadaan disekitar dan dapat memantau *quadcopter* walaupun secara tidak langsung terpantau oleh pengguna.

3. Untuk penelitian selanjutnya, jika penelitian hanya berfokus kepada sistem *monitoring quadcopter* saja, diharapkan dapat mengetahui dan merealisasikan tentang bagaimana cara untuk mendapatkan data parameter yang dihasilkan dari Ardupilot, agar peneliti tidak perlu membuat sistem *hardware*.

DAFTAR PUSTAKA

Arduino, dari (https://www.arduino.cc), di akses pada tanggal 11 Maret 2017, pukul 11.50 WIB.

Arduino Nano, dari (https://www.arduino.cc/en/Main/ArduinoBoard Nano), di akses pada tanggal 11 Maret 2017, pukul 11.53 WIB.

Ardupilot, dari (https://www.unmannedtechshop.co.uk/ardupilot-apm-2-8-flight-controller-board/), diakses tanggal 15 Maret 2017, pukul 21.14 WIB

Ardupilot GCS, dari (http://ardupilot.org/planner/docs/mission-planner-overview. html), diakses tanggal 1 April 2017, pukul 10.40 WIB.

Arifin, Fatchul dkk. 2015. *Rancang Bangun Quadcopter dilengkapi dengan Automatic Navigation GPS Control dan Camera Stabilizer sebagai alat bantu Monitoring Lalu Lintas dengan Live Streaming System*. Fakultas Teknik, Universitas Negeri Yogyakarta.

Banzi, Massimo. 2008. *Getting Started with* Arduino. Ed ke-1. *United States of America*: *Make:Books*, O'Reilly Media, Inc..

Borg, W.R. & Gall, M.D. Gall. (1989). *Educational Research: An Introduction, Fifth Edition.* New York: Longman. https://faridanursyahidah. files.wordpress.com/2012/06/research-and-development-vs-development-research.pdf. Diakses 26 Juni 2017.

Crawford, C. Merle. 1994. *New Products Management. Fourth Edition*. Irwin Marketing. http://journal.lppmunindra.ac.id/index.php/Faktor_Exacta/article /viewFile/55/53. Diakses 26 Juni 2017.

Dharmawan, Abe. 2009. *Pengendalian Motor Brushless DC dengan metode PWM Sinusoidal menggunakan Atmega 16*. [Jurnal Skripsi]. Fakultas Teknik, Universitas Indonesia.

Digi. 2012. *XBee Zig Bee User Manual*. United States of America: Digi International.Inc..

Djuandi, Feri. 2011. *Pengenalan Arduino* (*Pemula*). Jakarta: Elexmedia.

Fakultas Teknik. 2015. *Panduan Penyusunan Skripsi dan Non Skripsi*. Jakarta: Fakultas Teknik, Universitas Negeri Jakarta.

Frame F450, dari (https://www.*unmannedtechshop*.co.uk/f450-quadcopter-*frame-with-integrated*-pdb/), diakses pada tanggal 11 Maret 2017, pukul 15.46 WIB.

GCS, dari (http://rovindonesia.com/ground-control-station.html), diakses pada tanggal 28 Juni 2017, pukul 15.40 WIB.

Geeetech. 2015. *3DR Power Modul User Manual*. Dari (www.geeetech.com), diakses pada tanggal 25 Maret 2017, pukul 16.48 WIB.

Gravitech. 2016. *Arduino Nano (V3.0) User Manual*. *Creative Commons*.

Hardiansyah, Ade. 2014. *Rancang Bangun Quadcopter sebagai pemantau keamanan kampus Politeknik Negeri Sriwijaya*. [Laporan Akhir]. Palembang: Politeknik Negeri Sriwijaya.

IDE, dari (https://en.wikipedia.org/wiki/*Integrated_development_ environment*), diakses pada tanggal 11 Maret 2017, pukul 10.52 WIB.

Kurniawan, Dendy G.A. 2015. *Perancangan Sistem Pemantauan Kestabilan Quadcopter Robot berbasis MultiWii SE V2.5*. [Skripsi]. Jakarta: Fakultas Teknik, Universitas Negeri Jakarta.

Keller, Kevin Lane. 2003. *Strategic Brand Manajemen*. Second Edition. Prentice Hall. http://thesis.binus.ac.id/doc/Bab2/Bab%20II_11-51.pdf. Diakses 26 Juni 2017.

LAPAN. 2016. *Panduan Komurindo Kombat 2016*. Jakarta: Panitia Penyelenggara Komurindo Kombat 2016, Lembaga Penerbangan dan Antariksa Nasional (LAPAN).

Monitoring (Pemantauan), dari (https://kbbi.web.id/pantau-2), diakses pada tanggal 11 Maret 2017, pukul 12.04 WIB.

Monitoring, dari (https://id.wikipedia.org/wiki/Monitoring), diakses pada tanggal 11 Maret 2017, pukul 12.22 WIB.

Mulyana & Al Faiz. 2015. *Perancangan On-Board Data Handling untuk Roket EDF (Electric Ducted Fan)*. [Jurnal Skripsi]. Bandung: Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia.

Nasution, Arman Hakim. (2003). *Perencanaan dan Pengendalian Produksi*. Ed ke-1. Surabaya: Guna Widya. http://repository.uin-suska.ac.id/3759/3/BAB%20II.pdf. Diakses 26 Juni 2017.

Navigasi, dari (http://kbbi.web.id/navigasi), diakses pada tanggal 11 Maret 2017, pukul 12.24 WIB.

Navigasi, dari (https://id.wikipedia.org/wiki/Navigasi), diakses pada tanggal 11 Maret 2017, pukul 12.27 WIB.

Openpilot GCS, dari (https://showroom.qt.io/openpilot-gcs/), diakses pada tanggal 1 April 2017, pukul 12.10 WIB.

Oxer, Blemings. 2009. *Practical Arduino. United States of America*: apress.

Processing, dari (https://en.wikipedia.org/wiki/Processing_(*programming_ language*)), diakses pada tanggal 11 Maret 2017, pukul 12.50 WIB.

Propeller, dari (https://id.wikipedia.org/wiki/Baling-baling), diakses pada tanggal 12 Maret 2017, pukul 21.57 WIB.

Qgrouncontrol, dari (http://qgroundcontrol.org/about), diakses pada tanggal 1 April 2017, pukul 11.17 WIB.

Reas & Fry. 2007. Processing : *A Programming Handbook for Visual Desaigners and Artist* / Casey Reas & Ben Fry. London: John Maeda.

Sirajuddin. 2013. *Rancang Bangun Robot Terbang Quadcopter Berbasis Mikrokontroler ATmega16*. [Jurnal Skripsi]. Tanjungpura: Teknik Elektro Universitas Tanjungpura.

Sistem, dari (https://kbbi.web.id/sistem), diakses pada tanggal 11 Maret 2017, pukul 9.36 WIB.

Sistem, dari (https://en.wikipedia.org/wiki/System), diakses pada tanggal 11 Maret 2017, pukul 11.15 WIB.

Software, dari (https://en.wikipedia.org/wiki/*Software*), diakses pada tanggal 11 Maret 2017, pukul 11.01 WIB.

Software, dari (https://softwaredetail.wordpress.com/software/), diakses pada tanggal 28 Juli 2017, pukul 14.21 WIB.

Speed Sensor, dari (http://www.zuroya.net/2014/04/sensor-kecepatan-rpm-rotation-per-minute.html), diakses pada tanggal 29 Juli 2017, pukul 14.08 WIB

Sugiyono. 2011. *Metode Penelitian Kuantitatif Kualitatif dan R&D*. Bandung: Alfabeta. https://faridanursyahidah.files.wordpress.com/2012/06/research-and-development-vs-development-research.pdf. Diakses 26 Juni 2017.

Swamardika, Ida Bagus Alit. 2014. *Hand Motion Control untuk Menggerakkan Quadcopter Robot Menggunakan Sensor Accelerometer ADXL335 dan Wireless XBEE-PRO Series 1 60mW Berbasis Mikrokontroler ATmega32*. [Jurnal Skripsi]. Bali: Universitas Udayana.

Telemetri, dari (https://id.wikipedia.org/wiki/Telemetri), diakses pada tanggal 11 Maret 2017, pukul 18.38 WIB.

Telemetri-*West*, L3. 2014. *Telemetry Tutorial*. San Diego: (L-3com.com/TW).

Ulrich, Karl T. & Steven D. Eppinger. 2001. Perancangan & Pengembangan Produk. Jakarta: Salemba Teknika. http://thesis.binus.ac.id/doc/Bab2/2007-3-00407-TI-Bab%202.pdf. Diakses 26 Juni 2017.

XBee *adapter*, dari (http://site.gravitech.us/MicroResearch/Others/XBee-USB/XBee-USB_Manual.pdf), diakses pada tanggal 11 Maret 2017, pukul 18.39 WIB.

Gambar Ardupilot, dari (http://i.ebayimg.com/images/i/231982434726-0-1/s-l1000.jpg), diakses pada tanggal 11 Maret 2017, pukul 16.03 WIB.

Gambar Dinamika *Quadcopter*, dari (https://*creativentechno.files*.word press.com/ 2012/06/quadcopter_x_*flight_dynamics_yaw_pitch_roll*.jpg), diakses pada tanggal 11 Maret 2017, pukul 19.09 WIB.

Gambar ESC, dari (https://hobbyking.com/en_us/turnigy-k-force-30a-brushless-esc-opto.html), tanggal 11 Maret 2017, pukul 19.39 WIB.

Gambar *Frame* F450, dari (http://www.radioc.co.uk/v/vspfiles/photos/ 1000-2.jpg), diakses pada tanggal 11 Maret 2017, pukul 15.40 WIB.

Gambar GPS, dari (https://*store.fut-electronics*.com/*products*/ublox-neo-6m-gps-module), diakses pada tanggal 11 Maret 2017, pukul 14.58 WIB.

Gambar Motor BLDC, dari (Sumber: https://hobbyking.com/en_us/ntm-prop-drive-series-28-30a-1000kv-370w-1.html), diakses pada tanggal 15 Maret 2017, pukul 20.47 WIB.

Gambar Multiwii GUI, dari (https://walkera-fans.de/runner-250-basiert-auf-multiwii-2-2/), diakses pada tanggal 5 April 2017, pukul 20.56 WIB.

Gambar Navigasi *Quadcopter*, dari (https://fahmizaleeits.word press.com/2013/02/ 15/dasar-dasar-quadcopter-dan-setup-kkboard-2-0-pada-quadcopter-mode-x/), diakses pada tanggal 11 Maret 2017, pukul 19.27 WIB.

Gambar Openpilot GCS, dari (http://www.hit-karlsruhe.de/hit-info/info-ws13/ EFIS/EFIS%20Team1/standder technik.html), diakses pada tanggal 1 April 2017, pukul 12.40 WIB.

Gambar *Propeller*, dari (http://*hobbyready*.com/pic/201508/1214131. jpg), diakses pada tanggal 12 Maret 2017, pukul 22.23 WIB

Gambar Radio Control, dari (https://pixhawk.org/_media/modules/peripherals/ futaba_t7c.png?w=400&tok=d78678), diakses pada tanggal 11 Maret 2017, pukul 18.29 WIB.

Lampiran 1. *Listing* Program pada Arduino *Transmitter*

```
//-----------------------------------Library & Membuat objek baru
#include <Wire.h>
#include <ADXL345.h>
#include <Adafruit_BMP085.h>
#include <HMC5883L.h>
ADXL345 accelerometer;
Adafruit_BMP085 bmp;
HMC5883L compass;

//----------------------------------------Deklarasi Variabel bebas
int headingDegrees;
int yaw;
int error=0;
int fpitch;
int froll;
int masuk;
int bldc1,bldc2,bldc3,bldc4;
int hitam1,hitam2,hitam3,hitam4;
int putih1,putih2,putih3,putih4;
int hasil1,hasil2,hasil3,hasil4;
int jumlah1,jumlah2,jumlah3,jumlah4;
float V_in=0.00, V_sampling=0.00, V_mean=0.00, V_out=0.00;
float I_in=0.00, I_sampling=0.00, I_mean=0.00, I_out=0.00;
int persentase;
boolean Blink = true;
unsigned long prevMillisdesk  = 0;
unsigned long prevMillisdesk1 = 0;
unsigned long prevMillisdesk2 = 0;
unsigned long desk  = 2;
long desk1 = 200;
long desk2 = 200;
int rpm1, rpm2, rpm3, rpm4;

//----------------------------------------------Pengaturan program
void setup() {

  pinMode(9,INPUT);
  pinMode(8,INPUT);
  pinMode(7,INPUT);
  pinMode(6,INPUT);
  jumlah1 = 0; jumlah2 = 0; jumlah3 = 0; jumlah4 = 0;
  hasil1  = 0; hasil2  = 0; hasil3  = 0; hasil4  = 0;
  hitam1  = 0; hitam2  = 0; hitam3  = 0; hitam4  = 0;
  putih1  = 0; putih2  = 0; putih3  = 0; putih4  = 0;
  Serial.begin(9600);
  Wire.begin();
  compass = HMC5883L();
  error = compass.SetScale(1.3);
  if(error != 0)
    Serial.println(compass.GetErrorText(error));
  error = compass.SetMeasurementMode(Measurement_Continuous);
  if(error != 0)
    Serial.println(compass.GetErrorText(error));
  accelerometer.begin();
  accelerometer.setRange(ADXL345_RANGE_8G);
  bmp.begin();
}
```

```
//-----------------------------------------------------Program Utama
void loop() {

  unsigned long Millisdesk  = millis();
  unsigned long Millisdesk1 = millis();
  unsigned long Millisdesk2 = millis();
  bldc1 = digitalRead(9);
  bldc2 = digitalRead(8);
  bldc3 = digitalRead(7);
  bldc4 = digitalRead(6);

//--------------------------------Sketch Power Module
if(Millisdesk - prevMillisdesk > desk){
  prevMillisdesk = Millisdesk;
    for(int i = 0; i <= 25; i++){
      V_in = analogRead(A1);
      I_in = analogRead(A0);
      V_sampling = V_sampling + V_in;
      I_sampling = I_sampling + I_in;
    }
  }
  V_mean = V_sampling/25.00;
  I_mean = I_sampling/25.00;
  V_out  = ((V_mean * 0.00520)*11.1);
  I_out  = (I_mean  * 0.235);
  persentase = ((V_out-11.1)/1.5)*100;

  //--------------------------------RPM Motor BLDC 1
  if (bldc1 == 1){
    if (putih1 == 0){
      putih1 = 1;
      hitam1 = 0;
    }
  }

  if (bldc1 == 0){
    if(hitam1 == 0){
      hasil1 = 1;
      putih1 = 0;
      hitam1 = 1;
    }
  }

  //--------------------------------RPM Motor BLDC 2
  if (bldc2 == 1){
    if (putih2 == 0){
      putih2 = 1;
      hitam2 = 0;
    }
  }

  if (bldc2 == 0){
    if(hitam2 == 0){
      hasil2 = 1;
      putih2 = 0;
      hitam2 = 1;
    }
  }
```

```
  //--------------------------------RPM Motor BLDC 3
  if (bldc3 == 1){
    if (putih3 == 0){
      putih3 = 1;
      hitam3 = 0;
    }
  }

  if (bldc3 == 0){
    if(hitam3 == 0){
      hasil3 = 1;
      putih3 = 0;
      hitam3 = 1;
    }
  }

 //--------------------------------RPM Motor BLDC 4
 if (bldc4 == 1){
    if (putih4 == 0){
      putih4 = 1;
      hitam4 = 0;
    }
  }

  if (bldc4 == 0){
    if(hitam4 == 0){
      hasil4 = 1;
      putih4 = 0;
      hitam4 = 1;
    }
  }

  jumlah1 = jumlah1 + hasil1;
  jumlah2 = jumlah2 + hasil2;
  jumlah3 = jumlah3 + hasil3;
  jumlah4 = jumlah4 + hasil4;

  rpm1 = (jumlah1/2);
  rpm2 = (jumlah2/2);
  rpm3 = (jumlah3/2);
  rpm4 = (jumlah4/2);

//--------------------------------Cetak Tegangan, Arus & RPM Motor
 if((Blink == true) && (Millisdesk1 - prevMillisdesk1 > desk1)){
   Blink = false;
   prevMillisdesk1 = Millisdesk1;
   Serial.print("#");
   Serial.print(persentase); // Kapasitas Tegangan
   Serial.print(";");
   Serial.print(I_out);
   Serial.print(";");
   Serial.print(rpm1);
   Serial.print(";");
   Serial.print(rpm2);
   Serial.print(";");
   Serial.print(rpm3);
   Serial.print(";");
   Serial.print(rpm4);
```

```
    jumlah1 = 0;
    jumlah2 = 0;
    jumlah3 = 0;
    jumlah4 = 0;
  }

  hasil1 = 0;
  hasil2 = 0;
  hasil3 = 0;
  hasil4 = 0;
  V_sampling=0.00;
  I_sampling=0.00;
  V_mean=0.00;
  I_mean=0.00;

//---------------------------Cetak Magneto, Akselero, Giro & Baro
if((Blink == false) && (Millisdesk2 - prevMillisdesk2 > desk2)){
  Blink = true;
  prevMillisdesk2 = Millisdesk2;
//----------------------------------------------Sketch Magnetometer
  MagnetometerRaw raw = compass.ReadRawAxis();
  MagnetometerScaled scaled = compass.ReadScaledAxis();
  int MilliGauss_OnThe_XAxis = scaled.XAxis;
  float heading = atan2(scaled.YAxis, scaled.XAxis);
  float declinationAngle = 0.0457;//
  heading += declinationAngle;
  if(heading < 0)//0
    heading += 2*PI;
  if(heading > 2*PI)
    heading -= 2*PI;
  headingDegrees = heading * 180/M_PI;
//----------------------------------------------Sketch Akselero Giro
  Vector norm = accelerometer.readNormalize();
  Vector filtered = accelerometer.lowPassFilter(norm, 0.5);
  fpitch =-(atan2(filtered.XAxis,sqrt(filtered.YAxis*filtered.
            Yaxis+filtered.ZAxis*filtered.ZAxis))*180.0)/M_PI;
  froll  = (atan2(filtered.YAxis, filtered.ZAxis)*180.0)/M_PI;
  Serial.print(";");
  Serial.print(headingDegrees);
  Serial.print(";");
  Serial.print(fpitch);//+90
  Serial.print(";");
  Serial.print(froll);//+180
  Serial.print(";");
  Serial.println(bmp.readAltitude(101500)-52);//7meter
  }
}
```

Lampiran 2. *Listing* Program pada Arduino *Receiver*

```
char chrDataMasuk;
String strDataMasuk = "";

void setup(){
  Serial.begin(9600);
}

void loop(){

  if (Serial.available()>0){

  chrDataMasuk = Serial.read();

    if (chrDataMasuk == '#')
      {
       Serial.println(strDataMasuk);
       strDataMasuk = "";
      }
    else
      {
       strDataMasuk = strDataMasuk + chrDataMasuk;
      }
  }
  Serial.flush();
}
```

Lampiran 3. *Listing* Program GCS pada Processing IDE

***Listing* Program Void Setup :**

```
import processing.serial.*;
import controlP5.*;
ControlP5 cp5;
DropdownList serialPortsList;
DropdownList baudRateList;
int lf = 10; //------------------------------Linefeed in ASCII
String myString = null;
Serial myPort;

float V_BATT, I_BATT, YAW, PITCH, ROLL, ATD;
float rpm1; float rpm2; float rpm3; float rpm4;

Knob myKnobA; Knob myKnobB; Knob myKnobC; Knob myKnobD;
Slider Altimeter;
Slider Battery;
//----------------------------------------Set Posisi Rpm Motor
int trX=40;
int trY=395;
//-----------------------------Set Posisi Altimeter Quadcopter
int txa=780;
int tya=105;
//-----------------------------------------Set Posisi Baterai
int txb=990;
int tyb=220;
//---------------------------------------------Buat Objek Baru
PImage Qcop;
PImage UNJ;
PFont tulisan;
PFont tulisan1;
int BaudRate = 9600;
String PortName;

void setup() {

  size(1200,750,P3D);
  smooth();

  PortName = Serial.list()[0];

  printArray(Serial.list());
  myPort = new Serial(this, PortName, BaudRate);
  myPort.clear();
  myString = myPort.readStringUntil(lf);
  myString = null;

  cp5 = new ControlP5(this);

            cp5.addToggle("toggle")
               .setPosition(985,30)
               .setSize(60,30)
               .setValue(false)
               .setMode(ControlP5.SWITCH)
               ;
```

```
  myKnobA = cp5.addKnob("motor1_kiri_depan")
                .setRange(0,13500)
                .setValue(0)
                .setPosition(0,0)
                .setRadius(50)
                ;
  myKnobB = cp5.addKnob("motor3_kanan_depan")
                .setRange(0,13500)
                .setValue(0)
                .setPosition(200,0)
                .setRadius(50)
                ;
  myKnobC = cp5.addKnob("motor2_kiri_belakang")
                .setRange(0,13500)
                .setValue(0)
                .setPosition(0,200)
                .setRadius(50)
                ;
 myKnobD =  cp5.addKnob("motor4_kanan_belakang")
                .setRange(0,13500)
                .setValue(0)
                .setPosition(200,200)
                .setRadius(50)
                ;
 Altimeter= cp5.addSlider("Altimeter")
                .setLabel(" ")
                .setRange(0,50)
                .setValue(0)
                .setPosition(400,35)
                .setSize(20,250)
                ;
 Battery =  cp5.addSlider("Battery")
                .setLabel("% Battery")
                .setRange(0,100)
                .setValue(0)
                .setPosition(25, 0)
                .setSize(100,40)
                ;
                tulisan   =loadFont("BankGothicBT-Medium-48.vlw");
                Qcop      = loadImage("Quadcopter.png");
                UNJ       = loadImage("Logo UNJ.png");
                textFont(tulisan);
//-------------------------------------------------------Grafik
  pushMatrix();
  lineGraphX= new LineGraph(width,-
30,2*height/4,width/1.25,height/3);
  lineGraphY= new LineGraph(width,-
30,2*height/4,width/1.25,height/3);
  lineGraphZ= new LineGraph(width,-
30,2*height/4,width/1.25,height/3);
  lineGraphX.lineColor = color(255, 0, 0);
  lineGraphY.lineColor = color(0, 255, 0);
  lineGraphZ.lineColor = color(0, 0, 255);
  popMatrix();
}
```

***Listing* Program Void Draw :**

```
boolean Koneksi;
void toggle(boolean theFlag){
  if(theFlag == true) {
    Koneksi = true;}
  else {
    Koneksi = false;}
  println("a toggle event.");
}

void draw() {

background(100);
pushMatrix();
fill(255,255,0);
textSize(15);
text("PORT : ", 870, 40);
text("BAUD : ", 870, 60);
if (Koneksi == true){
    fill(255,255,0);
    text(PortName,940,40);
    text(BaudRate,940,60);
    fill(0,180,250);
    textSize(18);
    text("Disconnect",1015,20);
}
else {
  fill(0,180,250);
  textSize(18);
  text("Connect",1015,20);
}
popMatrix();

clock();
runtime();

while (myPort.available() > 0) {
    myString = myPort.readStringUntil(lf);
  if (Koneksi == true){
    if (myString != null) {
              float [] list = float (split(myString, ';'));
print(myString); //println(list);
     if (list.length > 9) {
        V_BATT = (list[0]);
        I_BATT = (list[1]);
        rpm1   = (list[2]);
        rpm2   = (list[3]);
        rpm3   = (list[4]);
        rpm4   = (list[5]);
        YAW    = (list[6]);
        PITCH  = (list[7]);
        ROLL   = (list[8]);
        ATD    = (list[9]);
      }
    }
  }
}
```

```
//-------------------------------------Garis putih horizontal
pushMatrix();
noStroke();
translate(10,50);
fill(255);
rect(95, 20, 715, 4);
popMatrix();
pushMatrix();
translate(0,360);
fill(255);
rect(10, 20,1180, 3);
popMatrix();
//-------------------------------------------------------Judul
pushMatrix();
  fill(255);
  textSize(45);
  text("GROUND CONTROL STATION", 465, 35);
  textSize(25);
  text("SISTEM MONITORING NAVIGASI QUADCOPTER", 465, 60);
popMatrix();
//-----------------------------------------------------Logo UNJ
pushMatrix();
  scale(0.7);
  image(UNJ,15,10);
popMatrix();

RPMmeter();
altimeter();
battery();

   pushMatrix();
   translate(20,111);//(-100,-50)
   scale(0.8);
   pitchroll();
   popMatrix();

   pushMatrix();
   translate(240, 90);
   scale(0.8);
   yaw=YAW;
   yawcompass();
   popMatrix();

   pushMatrix();
   translate(600, 175);
   scale(0.2);
   attitude();
   popMatrix();

   pushMatrix();
   translate(450, 250);
   scale(0.7);
   graphic();
   popMatrix();
}
```

***Listing* Program Clock & Run Time :**

```
//-------------------------------------------------------CLOCK
void clock() {

int  s = second();
int  m = minute();
int  h = hour();

  String time = h + ":" + m + ":" + s + " WIB";
  textAlign(CENTER, CENTER);
  textSize(18);
  fill(255, 0, 0);
  text("CLOCK", 1120, 25);
  text(time, 1120, 40);
  pushMatrix();
   translate(0,0);
   fill(50);
   rect(1065, 50, 110, 0.5);
  popMatrix();
}

//-----------------------------------------------------RUN TIME
int millisecs;
int seconds;
int minutes;

void runtime(){

  if(Koneksi){
  if (int(millis()/100) % 10 != millisecs){
    millisecs++;

  }
  if (millisecs >= 10){
    millisecs -= 10;
    seconds++;
  }
  if (seconds >= 60){
    seconds -= 60;
    minutes++;
  }
  }

textAlign(CENTER);
fill(255);
textSize(18);
text("RUN TIME",1120,66);
text(nf(minutes,2)+":"+nf(seconds,2)+"."+nf(millisecs,1),1120,80);
}
```

***Listing* Program Battery** :

```
void battery(){

pushMatrix();
//-------------------------------Kilat merah & Tanda perhatian
float lipo = map(V_BATT,0,100,0,100);

    cp5.getController("Battery").setValue(lipo);//rpm

  if((lipo)<=15){
  pushMatrix();
    translate(txb,tyb);
    beginShape();
      int m = millis();
      fill(m%1000,0,0);
      vertex(22, 8);
      vertex(1, 20);
      vertex(19, 20);
      vertex(-2, 32);
    endShape(CLOSE);
    textSize(25);
    text("!", -8, 30);
  popMatrix();
  }
  //----------------------------------------------Kilat kuning
  else{
  pushMatrix();
    translate(txb,tyb);
    beginShape();
      fill(255,255,0);
      vertex(20, 10);
      vertex(3, 20);
      vertex(17, 20);
      vertex(0, 30);
    endShape(CLOSE);
  popMatrix();
  }
  //----------------------------------------------Font Baterai
  pushMatrix();
    translate(txb,tyb);
    fill(255);
    textSize(15);
    text("BATTERY", 75, 60);
  popMatrix();
  //-------------------------------------------------Text Arus
  pushMatrix();
  translate(txb,tyb);
  fill(255,255,0);
  textSize(18);
  text("Current : " + int(I_BATT) +"A",75,-20);
  popMatrix();
popMatrix();
}
```

***Listing* Program Yaw :**

```
void yawcompass(){

//-----------------------------------------Background Circle
  pushMatrix();
  noStroke();
  fill(108, 156, 255); // Fill sky color
  ellipse(150, 150, 210, 210);

  if ((int)yaw >= 1 && (int)yaw <= 360)
    yawDeg = (int)yaw;
  else yawDeg = 360 + (int)yaw;

  textSize(20);
  fill(255, 255, 255);
  textAlign(CENTER);
  text("Heading: " + (int) +yawDeg+" Deg", 150, 315); //150, 315
  translate(0, 0);          //posisi pointer
  CompassPointer();
popMatrix();

//-----------------------------------------------Circular Scale
pushMatrix();
  pushMatrix();
    translate(150, 150);
    stroke(255);
    strokeWeight(3);
    noFill();
    ellipse(0, 0, 215, 215);
    SpanAngle=180;
    NumberOfScaleMajorDivisions=18;
    NumberOfScaleMinorDivisions=36;
    circularScale();
    rotate(PI);
    SpanAngle=180;
    circularScale();
    rotate(-PI);
  popMatrix();
  //-------------------------------------------------Polarized
    textSize(20);
    fill(255, 255, 255);
    textAlign(CENTER);
    text("W", 20, 157);
    text("E", 280, 157);
    text("N", 150, 25);
    text("S", 150, 290);
    textSize(15);
    rotate(PI/4.5);
    text("NW", 78, 15);
    text("SE", 343, 35);
    rotate(-PI/2.5);
    text("NE", 180, 185);
    text("SW", -85, 235);
popMatrix();
}
```

```
//-------------------------------------------------------Jarum
void CompassPointer() {
  translate(150,150);
  pushMatrix();
    rotate(radians(yaw));
    stroke(0);
    strokeWeight(2);
    fill(255, 255, 255);

    beginShape();
    vertex(0, -105);
    vertex(12, 100);
    vertex(0, 80);
    vertex(-12, 100);
    endShape(CLOSE);

    ellipse(0, 0, 9, 9);
    ellipse(0, 0, 3, 3);
  popMatrix();
}
```

Listing Program Pitch & Roll :

```
void pitchroll(){

//-------------------------------------------Pengontrol Pitch
pushMatrix();

  translate(130,124);    //Posisi lingkaran keseluruhan (130,124);
  fill(79,213,245);      //Warna (biru) atas lingkaran
  ellipse(0,0,220,220);  //Posisi & besar lingkaran
  fill(208,119,0);       //Warna bawah lingkaran
  beginShape();          //Membuat kotak variable warna coklat
  vertex(-110,110);
  vertex(110,110);
  vertex(110,-PITCH);
  vertex(-110,-PITCH);
  endShape(CLOSE);

  textSize(20);
  fill(255);
  textAlign(CENTER);
  text("Pitch= "+int(PITCH)+"°"+"  Roll= "+int(ROLL)+"°",0,165);

//-------------Penghalang kotak warna coklat diluar lingkaran
  pushMatrix();
    stroke(100);
    strokeWeight(27);
    noFill();
    ellipse(0,0,230,230);
    triangle(-70,-105,-105,-105,-105,-70);
    triangle(70,-105,105,-105,105,-70);
    triangle(70,105,105,105,105,70);
    triangle(-70,105,-105,105,-105,70);
  popMatrix();
```

```
//--------------------Menampilkan skala disekeliling lingkaran
  pushMatrix();
    rotate(HALF_PI);
    SpanAngle=180;
    NumberOfScaleMajorDivisions=18;
    NumberOfScaleMinorDivisions=36;
    circularScale();
  popMatrix();
//-------------------------------------------------Membuat ROLL
  rotate(radians(ROLL));
//-----------------------------------------Membuat garis ukur
  stroke(255);
  fill(255);
  strokeWeight(2);
  textSize(12);
  for (int i=-4; i<5; i++)
  {
    if ((i==0)==false)
    {
      line(40, 20*i, -40, 20*i);
    }
  }

  strokeWeight(2);
  for (int i=-9; i<10; i++)
  {
    if ((i==0)==false)
    {
      line(20, 10*i, -20, 10*i);
    }
  }

//-----------------------------------------Membuat angka ukur
  pushMatrix();
    text("0", 48, 5);
    text("0", -58, 5);
    text("-20", 45, 25);
    text("-20", -63, 25);
    text("+20", 45, -15);
    text("+20", -68, -15);
    text("-40", 45, 45);
    text("-40", -63, 45);
    text("+40", 45, -35);
    text("+40", -68, -35);
    text("-60", 45, 65);
    text("-60", -63, 65);
    text("+60", 45, -55);
    text("+60", -68, -55);
    text("-80", 45, 85);
    text("-80", -63, 85);
    text("+80", 45, -75);
    text("+80", -68, -75);
  popMatrix();
```

```
//-------------------------Membuat garis Vertikal & Horizontal
  stroke(255, 0, 0);
  strokeWeight(2);
  line(-45, 0, 45, 0);
  line(0, 90, 0, -90);
  fill(100, 255, 100);
  stroke(0);
  strokeWeight(1);
  triangle(0, -107, -6, -90, 6, -90);
  triangle(0, 107, -6, 90, 6, 90);
  stroke(255);
  strokeWeight(3);
  noFill();
  ellipse(0, 0, 217, 217);
popMatrix();
}

  float SpanAngle=120;
  int NumberOfScaleMajorDivisions;
  int NumberOfScaleMinorDivisions;
  float yaw = 0.0f;
  int yawDeg = 0;

void circularScale()
{
  pushMatrix();
  float GaugeWidth=304;
  textSize(GaugeWidth/20);
  float StrokeWidth=1;
  float an;
  float DivxPhasorCloser;
  float DivxPhasorDistal;
  float DivyPhasorCloser;
  float DivyPhasorDistal;
  strokeWeight(2*StrokeWidth);
  stroke(255);
  noFill();

  float DivCloserPhasorLenght=GaugeWidth/2-GaugeWidth/9-
StrokeWidth;
  float DivDistalPhasorLenght=GaugeWidth/2-GaugeWidth/7.5-
StrokeWidth;

for (int
Division=0;Division<NumberOfScaleMinorDivisions+1;Division++)
  {
    an=SpanAngle/2+Division*SpanAngle/NumberOfScaleMinorDivisions;

    DivxPhasorCloser=DivCloserPhasorLenght*cos(radians(an));
    DivxPhasorDistal=DivDistalPhasorLenght*cos(radians(an));
    DivyPhasorCloser=DivCloserPhasorLenght*sin(radians(an));
    DivyPhasorDistal=DivDistalPhasorLenght*sin(radians(an));
    line(DivxPhasorCloser,DivyPhasorCloser,DivxPhasorDistal,
    DivyPhasorDistal);
  }

  DivCloserPhasorLenght=GaugeWidth/2-GaugeWidth/10-StrokeWidth;
  DivDistalPhasorLenght=GaugeWidth/2-GaugeWidth/7.4-StrokeWidth;
```

```
  for (int Division=0; Division<NumberOfScaleMajorDivisions+1;
Division++)
  {
    an=SpanAngle/2+Division*SpanAngle/NumberOfScaleMajorDivisions;

    DivxPhasorCloser=DivCloserPhasorLenght*cos(radians(an));
    DivxPhasorDistal=DivDistalPhasorLenght*cos(radians(an));
    DivyPhasorCloser=DivCloserPhasorLenght*sin(radians(an));
    DivyPhasorDistal=DivDistalPhasorLenght*sin(radians(an));
    if
(Division==NumberOfScaleMajorDivisions/2|Division==0|Division==Num
berOfScaleMajorDivisions)
    {
      strokeWeight(5);
      stroke(255);

      line(DivxPhasorCloser, DivyPhasorCloser, DivxPhasorDistal,
DivyPhasorDistal);
      strokeWeight(3);
      //stroke(100, 255, 100);

      line(DivxPhasorCloser, DivyPhasorCloser, DivxPhasorDistal,
DivyPhasorDistal);
    } else
    {
      strokeWeight(3);
      stroke(255);
      line(DivxPhasorCloser, DivyPhasorCloser, DivxPhasorDistal,
DivyPhasorDistal);
    }
  }
  popMatrix();
}
```

***Listing* Program Attitude** :

```
int rotX, rotY, rotZ;

void attitude(){

  pushMatrix();
  scale(5);
  fill(255);
  textSize(15);
  text("ATTITUDE", 35, 168);
  popMatrix();

  pushMatrix();
  lights();
  smooth();
  translate(120, 180);
  rotateX(radians(PITCH+75));
  rotateY(radians(ROL));
  rotateZ(radians(YAW));

  stroke(255, 10, 10);
  strokeWeight(14);
  beginShape(LINES);
```

```
    //Panah Tengah
    vertex(0, 0, -20);
    vertex(0, 0-(height/5), -20);
    vertex(0, 0-(height/5), -20);
    vertex(0-(width/10)+10, 0-(height/5)+10, -20);
    vertex(0, 0-(height/5), -20);
    vertex(0+(width/10)-10, 0-(height/5)+10, -20);
  endShape();

  strokeWeight(20);
  beginShape(LINES);
  //Garis Kanan Atas
    vertex(0, 0, 0);
    vertex(width/3, 0-(height/3), 0);
  //Garis Kiri Atas
    vertex(0, 0, 0);
    vertex(0-(width/3), 0-(height/3), 0);
  //Garis Kiri Bawah
    vertex(0, 0, 0);
    vertex(0-(width/3), height/3, 0);
  //Garis Kanan Bawah
    vertex(0, 0, 0);
    vertex(width/3, height/3, 0);
  endShape();

  strokeWeight(20);
  fill(#2D00FF);
  //Motor Kanan Atas
  ellipse(width/3, 0-(height/3), width/5, height/5);
  //Motor Kiri Atas
  ellipse(0-(width/3), 0-(height/3), width/5, height/5);
  //Motor Kiri Bawah
  ellipse(0-(width/3), height/3, width/5, height/5);
  //Motor Kanan Bawah
  ellipse(width/3, height/3, width/5, height/5);

  noStroke();
  fill(255);
  sphere(height/10);
  popMatrix();
}
```

***Listing* Program Graphic** :

```
LineGraph lineGraphX, lineGraphY, lineGraphZ;

void graphic() {

  pushMatrix();
  lineGraphX.setValue(frameCount, mouseX);
  lineGraphY.setValue(frameCount, mouseY);
  lineGraphZ.setValue(frameCount, (mouseX+30));

  lineGraphX.update();
  lineGraphY.update();
  lineGraphZ.update();
```

```
  lineGraphX.draw();
  lineGraphY.draw();
  lineGraphZ.draw();
  popMatrix();
}

class LineGraph {
  float [] values;
  float gX, gY, gWidth, gHeight;
  color lineColor;
  int cursor;
  float lastValue = 0;
  float minValue  = 0;
  float maxValue  = 0;

  LineGraph(int nValues, float gX, float gY, float gWidth, float
gHeight) {
    values = new float[nValues];
    this.gX = gX;
    this.gY = gY;
    this.gWidth = gWidth;
    this.gHeight = gHeight;
    lineColor = color(255);
    maxValue = 10;
    minValue = -maxValue;
  }

  void setValue(int iValue, float newValue) {
    lastValue = values[cursor];
    iValue = (iValue % values.length);
    values[iValue] = (0.8*lastValue + 0.2*newValue);
    cursor = iValue;
  }

  void update() {
        minValue = min(values)<minValue ? min(values) : minValue;
        maxValue = max(values)>maxValue ? max(values) : maxValue;
  }

  void draw() {
   pushMatrix();
    noStroke();
    scale(2);
    fill(255);
    textSize(11);
    text("YAW :", 60, 300);
    text("PITCH :", 220, 300);
    text("ROLL :", 380, 300);
    text("GRAPHIC", 230, 340);
    fill(255,0,0);
    rect(100,292,10,10);
    fill(0,255,0);
    rect(260,292,10,10);
    fill(0,0,255);
    rect(420,292,10,10);
   popMatrix();
    stroke(255);
    strokeWeight(4.0);
```

```
    float x0 = map(cursor, 0, values.length, gX, gX+gWidth),
    x1 = x0,
    y0 = gY - gHeight/2,
    y1 = gY + gHeight/2;
    line(x0, y0, x1, y1);
    line(-30,250,-30,y1);
    line(930,250,930,y1);
    stroke(lineColor);
    for (int iValue=0; iValue<values.length-1; iValue++) {
      if (iValue < cursor) {
        x0 = map(iValue, 0, values.length, gX, gX+gWidth);
        x1 = map(iValue+1, 0, values.length, gX, gX+gWidth);
        y0 = map(values[iValue], minValue, maxValue, gY-gHeight/2,
gY+gHeight/2);
        y1 = map(values[iValue+1], minValue, maxValue, gY-
gHeight/2, gY+gHeight/2);

        line(x0, y0, x1, y1);
      }
    }
  }
}
```

Listing **Program Altitude** :

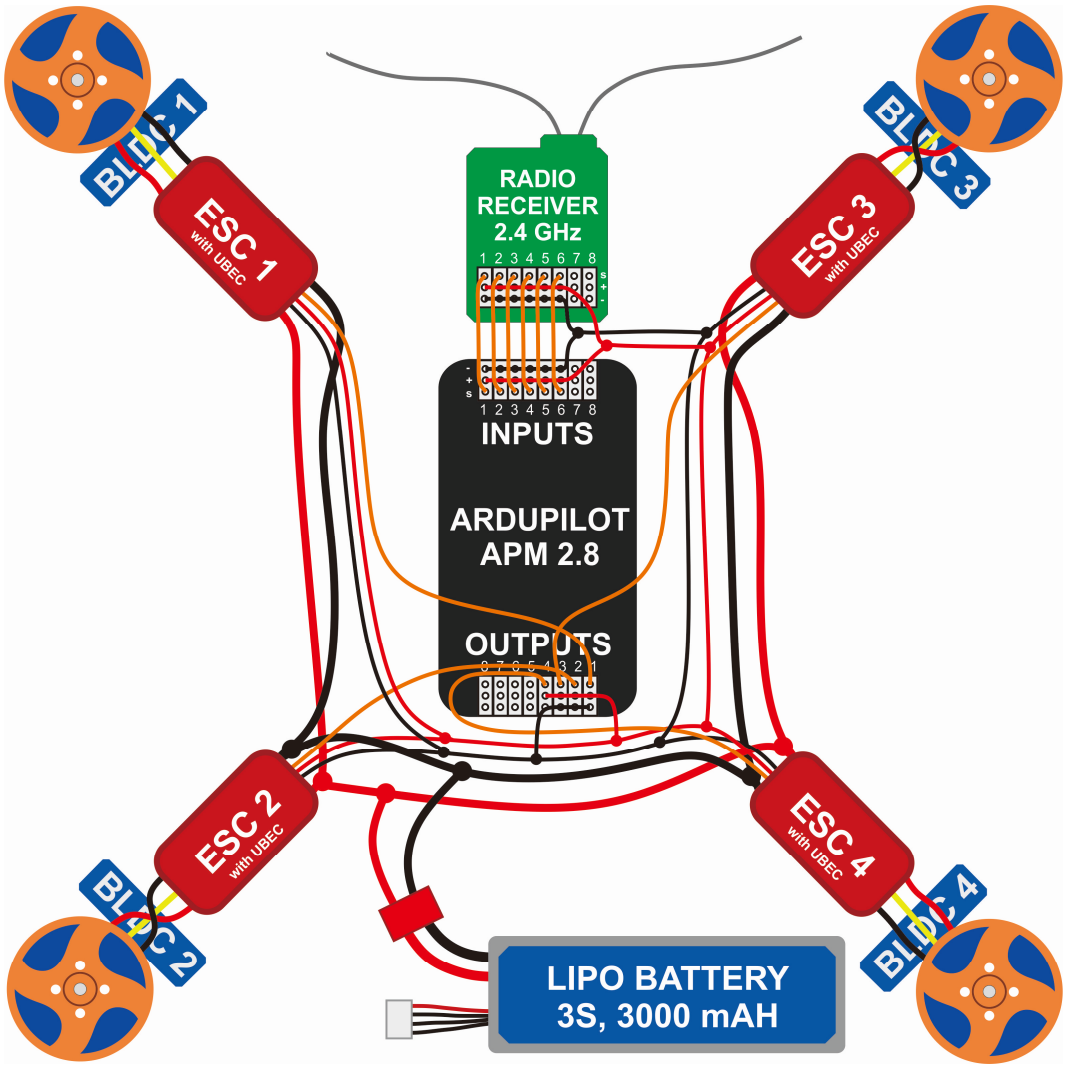
```
void altimeter(){

pushMatrix();
    float alt  = map(ATD,0,50,0,50);
    cp5.getController("Altimeter").setValue(alt);
    if(alt<=0){alt=0;}
    if(alt>=50){alt=50;}
    image(Qcop,txa,tya+220+(-alt*5));
  pushMatrix();
    fill(255);
    textSize(12);
    text("MDPL",txa+153,tya+260+(-alt*4.80));
  popMatrix();
  //-------------------------------------------Font Altimeter
  pushMatrix();
    translate(txa,tya);
    fill(255);
    textSize(15);
    text("ALTIMETER", 106, 270);
  popMatrix();
popMatrix();
}
```

Listing **Program RPM Meter** :

```
void RPMmeter(){

pushMatrix();
  //-----------------------------------------------------Frame
  pushMatrix();
    translate(trX+50,trY+50);
```

```
      pushMatrix();
        fill(255);
        textSize(15);
        text("RPM-METER", 99, 285);
      popMatrix();

    noStroke();
    fill(150,0,0);
    beginShape();
      vertex(0, 10);
      vertex(10, 0);
      vertex(200, 190);
      vertex(190, 200);
    endShape(CLOSE);

    beginShape();
      vertex(0, 190);
      vertex(10, 200);
      vertex(200, 10);
      vertex(190, 0);
    endShape(CLOSE);

  //-----------------------------------------------Body Frame
    beginShape();
      fill(150,100,100);
      vertex(100, 40);
      vertex(140, 80);
      vertex(100, 180);
      vertex(60, 80);
    endShape(CLOSE);
  popMatrix();

//-------------------------------------------------------Posisi
  cp5.getController("motor1_kiri_depan")
     .setPosition(trX,trY);
  cp5.getController("motor3_kanan_depan")
     .setPosition(trX+200,trY);
  cp5.getController("motor2_kiri_belakang")
     .setPosition(trX,trY+200);
  cp5.getController("motor4_kanan_belakang")
     .setPosition(trX+200,trY+200);
  cp5.getController("Battery")
     .setPosition(txb+25,tyb+0);
  cp5.getController("Altimeter")
     .setPosition(txa+110,tya+0);

//-------------------------------------------------------Kontrol
  cp5.getController("motor1_kiri_depan")
     .setValue(map(rpm1*378,0,13500,0,13500));
  cp5.getController("motor3_kanan_depan")
     .setValue(map(rpm3*378,0,13500,0,13500));
  cp5.getController("motor2_kiri_belakang")
     .setValue(map(rpm2*378,0,13500,0,13500));
  cp5.getController("motor4_kanan_belakang")
     .setValue(map(rpm4*378,0,13500,0,13500));
popMatrix();
}
```

Lampiran 4. ***Wiring Sistem Quadcopter*** **Ardupilot APM**



RIWAYAT HIDUP

Abdullah Hajis, Lahir di Kabupaten Bekasi, Jawa Barat pada tanggal 3 Mei 1994. Bertempat tinggal di Kp. Blokang, RT 004, RW 07, Desa Sukamanah, Kecamatan Sukatani, Kabupaten Bekasi. Peneliti menyelesaikan pendidikan formal di SDN Karang Setia 03 Kecamatan Karang Bahagia Kabupaten Bekasi pada tahun 2000 dan lulus pada tahun 2006. Kemudian melanjutkan pendidikan di SMPN 2 Sukatani Kecamatan Sukatani Kabupaten Bekasi pada tahun 2006 dan lulus pada tahun 2009. Selanjutnya peneliti melanjutkan pendidikan di SMKN 1 Tambelang Kecamatan Tambelang Kabupaten Bekasi mengambil jurusan Teknik Mekatronika pada tahun 2009 dan lulus pada tahun 2012 dengan mendapatkan predikat sebagai siswa terbaik. Setelah tamat SMK, peneliti melanjutkan ke Pendidikan Tinggi pada tahun 2012 memperoleh beasiswa Bidikmisi melalui jalur SNMPTN tertulis di Universitas Negeri Jakarta dengan memilih program studi Pendidikan Teknik Elektronika serta mempelajari konsentrasi Teknik Instrumentasi dan Kendali. Selama kuliah di Universitas Negeri Jakarta, peneliti aktif di Klub Robotika Universitas Negeri Jakarta dengan melakukan penelitian dan kegiatan kompetisi seperti Kontes Robot Indonesia (KRI) dan Kompetisi Muatan Roket dan Roket Indonesia (KOMURINDO). Peneliti juga aktif di organisasi kemahasiswaan Racana Universitas Negeri Jakarta dengan pengalaman pernah diamanahkan menjadi Sekretaris Dewan masa bakti 2014 dan Ketua Dewan Racana Universitas Negeri Jakarta masa bakti 2015.