LAPORAN TUGAS AKHIR

PENGEMBANGAN ANTARMUKA BERBASIS LABVIEW PADA ROKET MUATAN UNTUK KOMPETISI MUATAN ROKET INDONESIA 2019

Disusun oleh:

FUAD HAMMAMINATA ARYA ANJASMARA

20170120101

PROGRAM STUDI TEKNIK ELEKTRO FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH

YOGYAKARTA

2019

LAPORAN TUGAS AKHIR

PENGEMBANGAN ANTARMUKA BERBASIS LABVIEW PADA ROKET MUATAN UNTUK KOMPETISI MUATAN ROKET INDONESIA 2019

Tugas akhir ini disusun untuk memenuhi salah satu syarat untuk memperoleh gelar Sarjana Teknik

Disusun oleh:

FUAD HAMMAMINATA ARYA ANJASMARA

20170120101

PROGRAM STUDI TEKNIK ELEKTRO FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH

YOGYAKARTA

2019

HALAMAN PERNYATAAN

Yang bertanda tangan di bawah ini,

Nama	: Fuad Hammaminata Arya Anjasmara
NIM	: 20170120101
Tempat, tanggal lahir	: Wonosobo, 11 Juli 1997
Alamat tempat tinggal	: Kangkungan 04/02, Kadiluwih, Salam, Magelang
Alamat email	: fuadanjasmara@hotmail.com
No. HP	: 081217748764
Judul tugas akhir	: Pengembangan Antarmuka Berbasis Labview pada Roket Muatan untuk Kompetisi Muatan Roket Indonesia 2019

Dengan ini menyatakan bahwa naskah tugas akhir yang dibuat merupakan hasil pengerjaan dan penelitian yang dilakukan secara individu. Referensi yang dicantumkan sebagai landasan dalam pengerjaan berdasarkan penelitian atau tugas akhir yang telah dipublikasikan. Adapun tambahan referensi lainnya diperoleh dari situs referensi terkait dan tim Komurindo UMY.

Apabila di kemudian hari terdapat naskah yang tidak sesuai dengan pernyataan di atas, maka secara otomatis sanksi akademik yang sesuai ketentuan berlaku dan siap diterima. Demikian halaman pernyataan ini dibuat.

Yogyakarta, 09 April 2019

(FUAD H. A. A.)

HALAMAN PERSETUJUAN

TUGAS AKHIR

PENGEMBANGAN ANTARMUKA BERBASIS LABVIEW PADA ROKET MUATAN UNTUK KOMPETISI MUATAN ROKET INDONESIA 2019

Disusun oleh:

Fuad Hammaminata Arya Anjasmara

20170120101

PROGRAM STUDI TEKNIK ELEKTRO FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH YOGYAKARTA

YOGYAKARTA

2019

Telah Diperiksa dan Disetujui pada Tanggal

23 Maret 2019

Mengetahui,

Dosen Pembimbing I	**Dosen Pembimbing II**
Rama Okta Wiyagi, S.T., M.Eng.	**ing. Faaris Mujaahid, M.Sc.**
NIK. 19861017201504123070	**NIK. 19870718201704123101**

HALAMAN PENGESAHAN

TUGAS AKHIR

PENGEMBANGAN ANTARMUKA BERBASIS LABVIEW PADA ROKET MUATAN UNTUK KOMPETISI MUATAN ROKET INDONESIA 2019

Disusun oleh:

Fuad Hammaminata Arya Anjasmara

20170120101

Telah Dipertahankan dan Disahkan pada Tanggal 28 Maret 2019.

Susunan Dewan Penguji

Dosen Pembimbing I	**Dosen Pembimbing II**
Rama Okta Wiyagi, S.T., M.Eng.	**ing. Faaris Mujaahid, M.Sc.**
NIK. 19861017201504123070	**NIK. 19870718201704123101**

Penguji

Dr. Yessi Jusman, S.T., M.Sc.

NIK. 19840507201810201403

Skripsi Ini Telah Dinyatakan Sah sebagai Salah Satu Persyaratan untuk Memperoleh Gelar Sarjana Teknik

Mengesahkan,

Ketua Program Studi Teknik Elektro

Dr. Ramadoni Syahputra, S.T., M.T.

NIK. 19741010201010123056

"Sesungguhnya shalatku, ibadahku, hidupku, dan matiku hanyalah untuk Allah, Tuhan Semesta Alam." (Q.S. Al An'am : 162)

Dari Umar bin Al-Khattab –semoga Allah meridhoinya–, bahwa Rasulullah ﷺ bersabda, "*Amal itu tergantung niatnya, dan seseorang akan mendapatkan sesuai dari yang diniatkan. Barangsiapa yang hijrahnya karena Allah dan Rasul-Nya, maka hijrahnya kepada Allah dan Rasul-Nya, dan barangsiapa yang hijrahnya karena dunia atau karena wanita yang hendak dinikahinya, maka hijrahnya itu sesuai kepada ia berhijrah.*"

(H.R. Bukhari, Muslim, dan lainnya. Dapat dilihat pada *Hadits Arba'in* no. 1)

KATA PENGANTAR

Alhamdulillahirabbil'alamin, segala puji syukur penulis sampaikan hanya kepada Allah *Subhanahu wa Ta'ala*, yang tiada henti memberikan rahmat-Nya, yang tiada henti menolong hamba-Nya di saat bimbang, serta yang tiada henti menuangkan hidayah-Nya sehingga tugas akhir ini dapat terselesaikan dengan baik. Dari sekian waktu berlalu dan karena takdir-Nya, akhirnya tugas akhir ini disusun dengan judul "**Pengembangan Antarmuka Berbasis Labview pada Roket Muatan untuk Kompetisi Muatan Roket Indonesia 2019**".

Kompetisi Muatan Roket Indonesia dan Wahana Sistem Kendali atau biasa disingkat Komurindo merupakan salah satu kompetisi teknologi yang terkenal di kalangan pendidikan tinggi Indonesia. Kompetisi ini diselenggarakan oleh Lembaga Penerbangan dan Antariksa Nasional (LAPAN). Universitas Muhammadiyah Yogyakarta sebagai salah satu peserta dalam kompetisi ini diharapkan mampu mengambil banyak pelajaran agar menjadi kemajuan riset dan perkembangan bagi prodi Teknik Elektro UMY.

Dalam penulisan tugas akhir ini, penulis ingin berterima kasih kepada banyak pihak yang telah memberikan dukungan dan inspirasi kepada penulis, antara lain :

1. Kedua orangtua yang selalu turut serta mendukung dalam doa dan harta, juga doa dari seorang kakek tua yang telah mengajari hikmah kehidupan;
2. Dr. Ramadoni Syahputra, S.T., M.T., selaku Ketua Prodi Teknik Elektro;
3. Rama Okta Wiyagi, S.T., M.Eng., selaku Dosen Pembimbing I. Beliau sempatkan hadir pada sidang tugas akhir walaupun ada undangan dari rektorat di hari dan jam yang sama;
4. ing. Faaris Mujaahid, M.Sc., selaku Sekretaris Prodi Teknik Elektro, Dosen Pembimbing II, pemegang proyek tim UMY untuk Komurindo 2019, sekaligus plt. Dosen Pembimbing Akademik. Beliau sempatkan hadir pada

sidang tugas akhir walaupun dipenuhi kesibukan akreditasi prodi dan pekerjaan struktural lainnya;

5. Dr. Yessi Jusman, S.T., M.Sc., selaku Dosen Penguji pada sidang tugas akhir. Beliau dengan sabarnya menerima keputusan untuk memulai sidang terlebih dahulu sembari menunggu kedua dosen pembimbing menyusul;
6. Ustaz Dudu, Ustaz Arfiansyah, Ustaz Mahasin, Ustaz Hasan dan para pengajar lainnya di Pondok Pesantren Takwinul Muballighin;
7. Kelompok belajar bersama Shahibul Jannah: Adhit, Apri, Arif, Bama, Fadhil, Faisal, Ilyas, Mahesa, dan Yan;
8. Kelompok belajar bersama Yuk Mentoring: Abi, Azhom, Azzam, Deski, Iyin, Ray, Redho, dan Zainal;
9. Para pejuang Masjid Mardliyyah UGM;
10. Teman – teman, kakak angkatan, dan adik kelas yang tidak bisa disebutkan satu per satu.

Dalam sebuah penulisan tugas akhir tidak mungkin terlepas dari ketidaksempurnaan, baik dari segi teknik penulisan maupun dari segi materi yang disajikan. Tanpa mengurangi hormat, kritik dan saran sangat dinantikan untuk memenuhi kekurangan yang ada dalam tugas akhir ini. Semoga tulisan yang sedikit ini dapat menjadi salah satu referensi pengembangan antarmuka roket muatan bagi tim UMY maupun siapapun yang tertarik di bidang ini.

Yogyakarta, April 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

INTISARI

Universitas Muhammadiyah Yogyakarta sebagai salah satu kompetitor lomba roket nasional yang diadakan oleh Lapan, Komurindo 2019, memiliki potensi untuk mengembangkan antarmuka roket muatan yang lebih handal. Perancangan antarmuka ini dilakukan melalui dua langkah, yaitu perancangan *dummy data* dan antarmuka. Perancangan *dummy data* menggunakan Arduino UNO dan Arduino IDE, sedangkan perancangan antarmuka menggunakan LabVIEW 2017. Arduino IDE digunakan untuk memberikan perintah ke Arduino UNO agar dapat menghasilkan *dummy data* sehingga data tersebut dapat dijadikan contoh data yang masuk ke LabVIEW 2017. *Dummy data* dibuat melalui pin Analog 0 menggunakan perintah *randomSeed()* pada Arduino IDE. *Dummy data* dibuat sesuai format yang sudah ada pada penelitian sebelumnya. LabVIEW 2017 berperan dalam akuisisi data dan menampilkan hasil. Kemudian, pengujian data dilakukan melalui dua cara, yaitu pengujian secara kualitatif dan kuantitatif. Pengujian kualitatif dilakukan untuk menguji kehandalan dari sistem, sedangkan pengujian kuantitatif dilakukan untuk menguji besar error yang terjadi dalam jumlah pengujian tertentu. Dari hasil simulasi yang telah dilakukan, dapat diperoleh hasil bahwa sistem berjalan dengan cukup handal dan memiliki error sebesar 10% dari 50 kali pengambilan data. Dalam waktu 10 menit, terjadi error sebanyak tiga kali saja dan tidak memengaruhi kinerja LabVIEW sehingga LabVIEW berhenti bekerja. Hal ini sudah sesuai dengan kebutuhan tim Komurindo 2019.

Kata kunci: Komurindo 2019, *dummy data*, Arduino, LabVIEW.

ABSTRACT

Universitas Muhammadiyah Yogyakarta as the one of the competitor in the Lapan's competition, Komurindo 2019, have potentials to develop reliable interface for payload rocket. Designing process is in both ways, designing dummy data and interface. Designing dummy data is using Arduino UNO and Arduino IDE, while designing interface is using LabVIEW 2017. Arduino IDE gave command to Arduino UNO to produce dummy data so its data can be processed in LabVIEW 2017. Dummy data is produced from Analog 0 pin of Arduino UNO using randomSeed() code in Arduino IDE. Its format is determined as the research before. LabVIEW 2017 supporting data acqusition process and displaying result. Then, testing of data is in two methods, qualitative and quantitative. The qualitative method is trying to test the reliable of system. So, the quantitative method is trying to calculate the error of system in some system testing. From the simulation that was done, in result is system working reliable enough and 10 % error from 50 times data pickings. In 10 minutes, occurred three errors and not interfere LabVIEW process so LabVIEW is stop working. This matter is matched to the necessary of Komurindo 2019 team.

Keywords: Komurindo 2019, dummy data, Arduino, LabVIEW.

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

إِنَّ فِي خَلْقِ السَّمَاوَاتِ وَالْأَرْضِ وَاخْتِلَافِ اللَّيْلِ وَالنَّهَارِ لَآيَاتٍ لِّأُولِي الْأَلْبَابِ - 3:190

الَّذِينَ يَذْكُرُونَ اللَّهَ قِيَامًا وَقُعُودًا وَعَلَىٰ جُنُوبِهِمْ وَيَتَفَكَّرُونَ فِي خَلْقِ السَّمَاوَاتِ وَالْأَرْضِ رَبَّنَا مَا خَلَقْتَ هَٰذَا بَاطِلًا سُبْحَانَكَ فَقِنَا عَذَابَ النَّارِ - 3:191

Artinya:

Sesungguhnya dalam penciptaan langit dan bumi, dan silih bergantinya malam dan siang terdapat tanda-tanda bagi orang-orang yang berakal, (yaitu) orang-orang yang mengingat Allah sambil berdiri atau duduk atau dalam keadan berbaring dan mereka memikirkan tentang penciptaan langit dan bumi (seraya berkata): "Ya Tuhan kami, tiadalah Engkau menciptakan ini dengan sia-sia, Maha Suci Engkau, maka peliharalah kami dari siksa neraka. (Al Imran: 190-191)

Menurut riwayat dari Hasan Al-Basri, bahwa beliau pernah mengatakan, "Berpikir selama sesaat lebih baik daripada berdiri shalat semalam". Al-Fudail mengatakan bahwa Hasan Al Bashri pernah berkata, "Pikiran merupakan cermin yang memperlihatkan kepadamu kebaikan-kebaikan dan keburukan-keburukanmu". (Tafsir At Thabari, 883)

Roket merupakan salah satu teknologi yang terus dikembangkan sejak Perang Dingin antara Rusia dan Amerika hingga saat ini. Saat Perang Dingin, kedua negara ini saling klaim sebagai pelopor mengenai kesuksesan eksplorasi luar angkasa melalui teknologi antariksa mereka tersebut. Tujuan lainnya dari kompetisi teknologi ini adalah penguasaan militer. Akhirnya, pada masa kini, roket banyak dikembangkan dan dilombakan di masing-masing negara lainnya. Hal ini dapat kita simak di berbagai berita dan koran saat ini.

Kompetisi Muatan Roket Indonesia dan Wahana Sistem Kendali atau biasa disingkat Komurindo merupakan salah satu kompetisi teknologi yang terkenal di kalangan pendidikan tinggi Indonesia. Kompetisi ini diselenggarakan oleh Lembaga Penerbangan dan Antariksa Nasional (LAPAN), dimana diharapkan dari kompetisi ini lahir bibit-bibit unggul yang mencintai dan menguasai teknologi roket di bidang

struktur, kendali, maupun elektronika, termasuk sistem kontrol dan muatan roket. (LAPAN, 2017)

Universitas Muhammadiyah Yogyakarta sebagai salah satu peserta Komurindo tahun ini memiliki potensi untuk bisa mengembangkan roketnya lebih baik lagi dibandingkan tahun sebelumnya. Pada tahun 2017, Universitas Muhammadiyah Yogyakarta hampir mendapatkan kemenangan untuk divisi wahana sistem kendali. Teknologi yang digunakan mungkin sudah cukup baik, namun antarmuka yang dibuat untuk menampilkan pembacaan sensor roket masih membutuhkan banyak pengembangan, terutama pada kestabilan pembacaan sensor yang ditampilkan melalui antarmuka.

Selain itu, masih kurang penulisan terkait riset antarmuka pada roket muatan. Dalam hal ini, secara spesifik, adalah skripsi atau jurnal mengenai *Ground Control Station*. Kebanyakan yang menjadi perbincangan dalam penulisan skripsi atau jurnal mengenai roket muatan adalah mekaniknya. Masih jarang dituliskan mengenai antarmuka sebagai *Ground Control Station* dari pembacaan sensor dari roket yang dikembangkan secara sendiri.

Oleh karena itu, ditawarkan sebuah perancangan dan pengembangan antarmuka sebagai *Ground Control Station* untuk memantau kinerja roket, yaitu kondisi dan sikap roket pada saat diluncurkan hingga mencapai target. Di samping itu, diperlukan akurasi yang cukup baik dalam mengakuisisi data dengan baik ke komputer melalui aplikasi LabVIEW.

1.2 RUMUSAN MASALAH

Setelah dipaparkan latar belakang di atas, maka dapat dirumuskan rumusan masalah yang akan dihadapi dalam tugas akhir ini. Adapun dalam tugas akhir ini menghasilkan beberapa rumusan masalah yang diperoleh dari latar belakang di atas, antara lain:

a. Bagaimana cara merancang antarmuka yang lebih handal dalam pembacaan sensor dari roket menggunakan LabVIEW;

b. Bagaimana cara merancang *dummy data* menggunakan Arduino untuk melakukan simulasi akuisisi data ke komputer melalui LabVIEW;
c. Bagaimana pengujian antarmuka berbasis LabVIEW yang sesuai dengan kebutuhan Komurindo 2018-2019.

1.3 TUJUAN DAN BATASANNYA

Tujuan dalam tugas akhir ini adalah menghasilkan *Ground Control Station*. Batasan tugas akhir meliputi di dalam tujuan yang dipaparkan berikut ini.

a. Merancang antarmuka yang handal dalam menerima pembacaan sensor dari roket menggunakan LabVIEW;
b. Merancang dan menggunakan *dummy data* yang dihasilkan Arduino untuk melakukan simulasi akuisisi data melalui LabVIEW;
c. Melakukan pengujian antarmuka berbasis LabVIEW menggunakan *dummy data* sebagai sarana simulasi, sesuai dengan kebutuhan Komurindo 2018-2019.

1.4 MANFAAT

Dari penelitian ini, diharapkan menjadi sarana untuk meningkatkan dan mengembangkan kemampuan pembuatan antarmuka berupa *Ground Control Station* bagi tim roket Universitas Muhammadiyah Yogyakarta. Penelitian ini diharapkan memotivasi tim roket Universitas Muhammadiyah Yogyakarta untuk melanjutkan penyempurnaan antarmuka. Selain itu, penelitian ini juga diharapkan dapat membantu dalam memahami proses akuisisi data.

BAB II
LANDASAN TEORI

2.1 TINJAUAN PUSTAKA

Pada salah satu tugas akhir yang ada di Universitas Telkom pada tahun 2013, dikembangkan desain GUI untuk muatan roket oleh Masrul Musyawwir Mardis dan beberapa rekannya dari Prodi Teknik Telekomunikasi, Fakultas Ilmu Terapan, Universitas Telkom. Judul tugas akhirnya adalah "Desain Graphical User Interface untuk Muatan Roket sebagai Ground Stations". Dalam tugas akhir tersebut, dirancang antarmuka sebagai *Ground Station* menggunakan media transmisi frekuensi radio. Antarmuka tersebut dibuat menggunakan aplikasi yang memiliki pemrograman *Visual Basic*. Data yang diambil adalah sikap roket dan gambar bumi dari udara.

Selanjutnya, dikembangkan juga hal yang sama seperti di atas oleh Sugimiyanto dari Prodi Sistem Informasi, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia pada tahun 2014. Judul skripsi yang dikerjakan adalah "Membangun Aplikasi *Ground Control Station* sebagai Visualisasi Posisi Global Muatan Roket (*Payload*) di Bumi pada Kompetisi Muatan Roket Indonesia". Perbedaan dari penelitian sebelumnya adalah aplikasi yang digunakan untuk membuat antarmuka, yaitu pada penelitian ini menggunakan LabVIEW. Kelebihan dari penelitian ini dibandingkan dengan yang sebelumnya adalah menggunakan GPS untuk mengakses lokasi roket.

Ada pula penelitian yang berhubungan dengan sistem telemetrinya, agar data yang dikirim oleh roket dan diterima oleh komputer dapat menjadi lebih baik. Hal ini dikembangkan oleh Didik Hariyanto dan beberapa mahasiswanya di Prodi Pendidikan Elektro, Fakultas Teknik, Universitas Negeri Yogyakarta dan disampaikan pada Seminar Nasional Pendidikan Teknik Elektro 2014. Jurnal yang dibuat berjudul, "Pengembangan Sistem Telemetri antara *Payload* Roket dan *Ground Segment*". Namun, masih digunakan komunikasi serial dengan medium frekuensi radio, sehingga masih memiliki kelemahan pada jarak komunikasi antara roket dan komputer.

Pada tahun 2016, terdapat sebuah jurnal di *IEEE* berjudul, "*Wireless Sensing System for Flexible Arrayed Potentiometric Sensor Based on Xbee Module*" karya Jung-Chuan Chou (*IEEE*), Jie-Ting Chen, Yi-Hung Liao (*IEEE*), dan lainnya. Penelitian tersebut menggunakan komunikasi nirkabel yang lebih baik dibandingkan penelitian di atas sebelumnya. Penelitian ini menggunakan modul *Xbee*. Selain itu, antarmuka yang dikembangkan adalah menggunakan LabVIEW. Penelitian ini sedikit berbeda pembahasannya dari roket, karena data yang diambil adalah pH dan glukosa.

Selanjutnya, proyek akhir dari seorang mahasiswa asal Islandia bernama Þórarinn Árni Pálsson dari Jurusan Teknik Elektro Terapan, Universitas Reykjavik, Islandia. Judul penelitiannya adalah "*Drone Autopilot*" pada tahun 2017. Sesuatu yang dikembangkan pada proyek ini adalah kendali otomatis kapal tanpa awak. Proyek ini sangat bagus untuk pembelajaran dalam penelitian dibandingkan kebanyakan tugas akhir di Indonesia, yang hanya fokus pada satu hal saja dan monoton dalam penyampaian. Proyek ini menggunakan LabVIEW untuk menampilkan dan mengukur kecepatan rotasi, akselerasi, kecepatan, dan sudut pendayung kapal. Selain itu, fungsi dari antarmuka ini adalah untuk melakukan koreksi dan integrasi numerik (formula untuk mencapai hasil pengukuran).

Terakhir, sebuah jurnal berjudul "*Development of User Interface Based on LabVIEW for Unmanned Aircraft Application*", yang disusun pada tahun 2017 juga, oleh Faaris Mujaahid dan beberapa mahasiswanya dari Jurusan Teknik Elektro, Universitas Muhammadiyah Yogyakarta. Dari judul di atas, sudah tentu antarmuka dibuat menggunakan LabVIEW. Kemudian, riset yang dikembangkan di sini adalah menitikberatkan pada pengembangan antarmuka untuk roket muatan. Penelitian ini memang digunakan untuk keperluan riset lomba Komurindo. Dari sedikit penjelasan tersebut, terlihat bahwa riset yang dilakukan sangat identik dengan penelitian yang sedang dikerjakan saat ini, sehingga cukup sesuai dengan kebutuhan riset saat ini.

Dari beberapa referensi di atas, dapat dikembangkan menjadi sebuah tugas akhir yang cukup spesifik di bidang instrumentasi dan kendali.

2.2 ROKET

Roket MUTAN S-32 adalah roket yang didesain untuk mengikut lomba Komurindo tahun 2018-2019. Roket EDF (Electric Ducted Fan) ini memiliki bagian-bagian yang umum di temukan pada roket sistem EDF lainnya, berupa bagian launcher, telemetri dan roket. Bagian-bagian tersebut dirancang dengan mempertimbangkan aspek peraturan kontes yang telah disebutkan dalam panduan Komurindo 2018-2019. Hal ini dilakukan agar roket dapat meluncur secara optimal dan dapat mencapai target yang telah ditentukan sebelumnya.

Roket EDF ini terdiri dari tiga bagian, yaitu bagian motorik, bagian sensor, dan bagian pengendali. Bagian motorik adalah bagian roket yang berkaitan dengan komponen utama penggerak roket. Semua penjelasan di atas maupun di bawah nantinya mengenai roket dikutip dari Proposal Komurindo 2018, sebab segala hal terkait dengan piranti keras yang digunakan merupakan proyek tim roket Universitas Muhammadiyah Yogyakarta.

2.3 MOTORIK ROKET

Dalam aplikasi roket, khususnya lomba roket muatan di Indonesia, digunakan dua jenis motor listrik, yaitu EDF (*electric ducted fan*) dan servo. Kedua motor tersebut sangat berperan dalam aplikasi roket, terutama pada tugas mencapai target yang sudah ditentukan oleh panitia lomba Komurindo 2019 mendatang. Berikut ini adalah Tabel 1 yang mempermudah penjelasan dari kedua motor tersebut.

Tabel 1. Tabel Perbandingan Dua Motor yang Digunakan pada Roket

No.	Pembeda	Jenis Motor	
		EDF	Servo
1	Nama produk	Dr. Mad Thrust 90 mm 11-blade Alloy EDF 1400 kV Motor-2900 Watt (8S)	Turnigy ™ TGY-R5180MG 180°

2	Prinsip kerja dan fungsi	EDF memiliki prinsip kerja yaitu dengan memanfaatkan aliran fluida (udara) yang dihisap melalui sisi inlet, dan kemudian secara langsung mengeluarkannya melalui sisi outlet dengan bantuan putaran *blade*. Mekanisme ini menghasilkan dorongan dengan daya tertentu sesuai dengan kemampuan hisap dan keluaran EDF, serta putaran motor EDF.	Pengendalian sirip (*fin*) tersebut dilakukan oleh pengendali servo yang berfungsi sebagai pengendali sikap roket. Selain itu, servo juga digunakan untuk membuka katup parasut.
3	Ukuran	94 mm (diameter motor)	(22 ×12 × 21) mm
4	Berat	350 gr	12 gr
5	Kecepatan putar	45.000 rpm (maks.)	0.12 detik/60°- 0.10 detik/60°
6	Torsi	3,6 kg	2,0 kg cm
7	Tegangan yang dibutuhkan	29,6 V	4,8 – 6,0 V
8	Arus yang dibutuhkan	97 A	
9	Daya yang dibutuhkan	2400 – 2700 W	

Gambar 2.1 EDF

Gambar 2.2 Servo

2.4 SENSOR

Sensor yang digunakan pada roket MUTAN S-32 ini adalah sensor GY-87 Module IMU 10DOF. Sensor ini merupakan single board yang berisi beberapa sensor sekaligus yaitu MPU 6050, HMC5883L dan BMP180 yang memiliki fungsi masing-masing. Berikut ini merupakan penjelasan dari sensor-sensor tersebut melalui Tabel 2.

Tabel 2. Tabel Sensor pada GY-87 Module IMU 10DOF

No.	Nama Sensor	Pembeda	
		Jenis sensor	Fungsi
1	MPU 6050	*Accelerometer, Gyroscope*	Mengukur percepatan dinamik dan statik pada roket. Mengukur dan mempertahankan orientasi roket berdasarkan prinsip momentum sudut
2	HCM5883L	Magnetometer	Mengukur sudut arah roket
3	BMP180	Barometer	Mengukur tekanan udara

Gambar 2.3 Sensor GY-87

2.5 ARDUINO UNO

Arduino UNO digunakan sebagai pengontrol rangkaian elektronik pada roket MUTAN S-32. Salah satu jenis IDE (*Integrated Development Environment*) ini memiliki 14 pin digital input/output (6 pin PWM) dan 6 input analog. Selain itu, Arduino UNO dapat digunakan dalam banyak aplikasi, melalui *shield* yang sudah tersedia di pasaran.

Gambar 2.4 Arduino UNO

2.6 TELEMETRI

Telemetri merupakan perangkat yang digunakan sebagai media transmisi data secara nirkabel. MUTAN S-32 menggunakan jenis telemetri modul XBee Pro, yang merupakan jenis modul RF (Radio Frekuensi). Modul ini beroperasi pada frekuensi 2.4

GHz dan memiliki jangkauan 100 m (indoor) hingga 1500 m (outdoor). Modul ini membutuhkan arus sebesar 250 mA (untuk *Tx*) dan arus sebesar 50 mA (untuk *Rx*).

Gambar 2.5 Xbee-Pro

2.7 GPS

GPS *(Global Positioning System)* merupakan sebuah piranti/sistem penentu posisi roket saat mengudara, sehingga posisi roket dapat diketahui secara realtime. Roket MUTAN S-32 menggunakan GPS jenis M8N sebagai komponen yang menentukan posisi roket. GPS jenis ini menggunakan *u-blox M8N* untuk akurasi yang tinggi yang mendukung satelit GPS+GLONASS dan antena keramik CIROCOMM. GPS ini memiliki berat sekitar 10 gram dan dimensi (35 x 34) mm.

Gambar 2.6 GPS

2.8 BATERAI

Roket MUTAN S-32 menggunakan jenis baterai Li-Po TURNIGY NANO-TECH 1550 mAh 6S 65-130C dengan tegangan 22,2 volt per 6 sel. Kapasitas dari baterai yang digunakan adalah 1.550 mAh. Baterai digunakan sebagai catu daya bagi komponen-komponen elektronis yang ada pada roket.

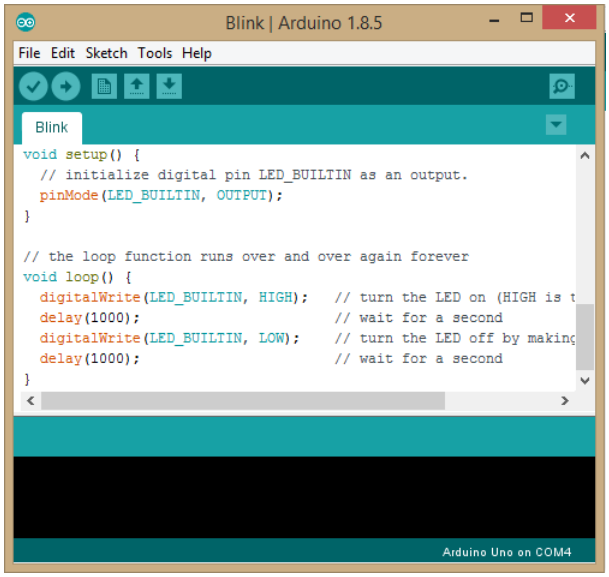
Gambar 2.7 Baterai Li-Po 1550mAh

2.9 ANTARMUKA (GCS)

Antarmuka yang dirancang pada tugas akhir ini adalah digunakan pada *Ground Control Station* (GCS), dimana GCS ini merupakan sebuah perangkat komunikasi data, baik menerima maupun mengirim data, untuk keperluan memantau dan atau mengendalikan roket muatan yang sedang meluncur. Kedua hal ini secara praktis tidak dapat dilakukan, namun secara teoritis dapat dilakukan menggunakan interupsi pada program LabVIEW maupun dari Arduino IDE. GCS ini menggunakan perangkat komputer atau laptop untuk aplikasinya, untuk melakukan proses di atas. Letak dari GCS ini adalah berada pada permukaan bumi, karena memang difungsikan sebagai pusat pemantauan roket yang sedang meluncur. Adapun definisi dari *Ground Control Station* (GCS) atau *Ground Segment* (GS) adalah perangkat *transmitter-receiver* di stasiun bumi yang dilengkapi dengan perangkat komputer yang berfungsi untuk mengendalikan dan atau memonitor wahana roket dan atau *payload* yang sedang meluncur. (LAPAN, 2018)

2.10 ARDUINO IDE

Arduino IDE (*Integrated Development Environment*) berperan dalam proyek ini, yaitu pada pembacaan data dan pengiriman data ke antarmuka. Data yang dibaca dari Arduino harus dipisahkan menjadi dua bit atas dan bawah untuk menghasilkan data yang stabil ketika akan ditampilkan pada antarmuka LabVIEW.



Gambar 2.8 Tampilan Program Blink *pada Aplikasi Arduino IDE*
(Diambil pada 24 September 2018 pukul 01.18)

Arduino IDE merupakan sebuah aplikasi *open-source* yang digunakan untuk melakukan pemrograman ke Arduino. Aplikasi ini menggunakan bahasa *C* yang mirip *Java* dalam pemrogramannya. Selain itu, Arduino IDE menyediakan banyak pilihan *library* yang dapat diakses sesuai kebutuhan proyek.

2.11 LABVIEW

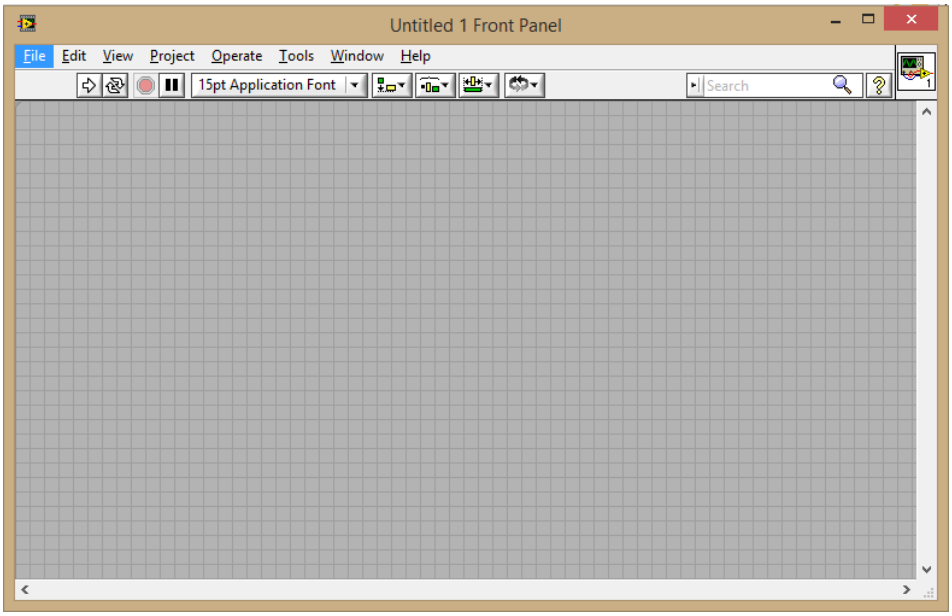
LabVIEW memiliki kepanjangan *Laboratory Virtual Instrument Engineering* Workbench. Perangkat lunak ini dikembangkan oleh sebuah perusahaan bernama *National Instruments Corporation*, yaitu sebuah perusahaan multinasional Amerika yang beroperasi secara internasional. Perusahaan ini berpusat di salah satu kota di negara bagian Texas, yaitu Austin. Perusahaan ini bergerak di bidang peralatan tes otomasi dan perangkat lunak instrumentasi visual.

Program LabVIEW disebut sebagai instrumen virtual atau sebutan aslinya adalah VI (*virtual instruments*), dimana tampilan dan operasi yang ada mengacu pada instrumen fisik, seperti osiloskop dan multimeter. Bahasa pemrogramannya bisa disebut sebagai VI. Oleh karena itu, ekstensi berkas dari program ini setelah disimpan adalah **.vi*. LabVIEW memiliki sebuah peralatan yang komprehensif untuk mengumpulkan, menganalisis, menampilkan, dan menyimpan data, juga dapat membantu menyelesaikan masalah pada desain antarmuka yang sedang dibuat.

Di dalam proses pembuatan antarmuka melalui LabVIEW, terdapat dua tampilan halaman yang muncul, yaitu *Front Panel* dan *Block Diagram*. Keduanya memiliki karakteristik dan fungsi masing-masing, namun keduanya saling berhubungan dan tidak dapat dipisahkan satu sama lain. Berikut ini merupakan penjelasan lebih lanjutnya.

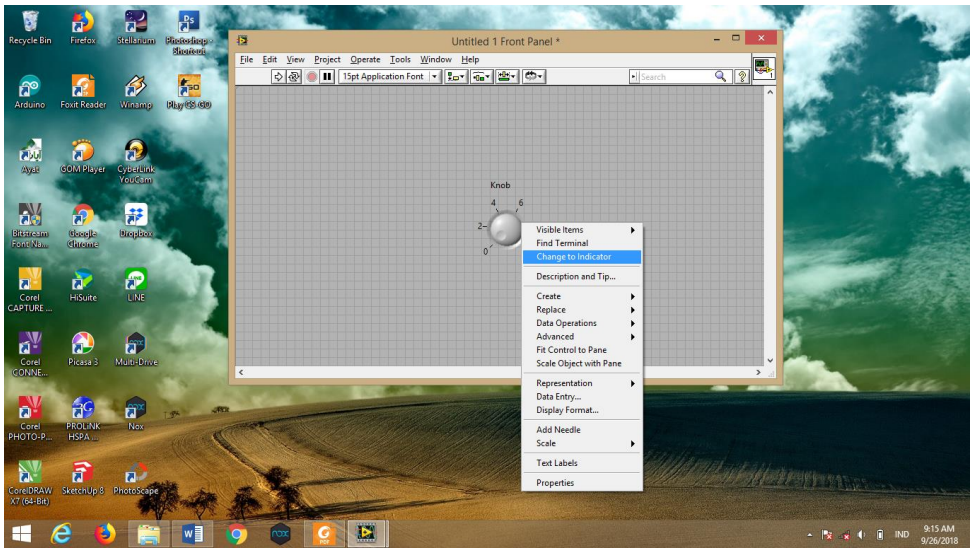
2.11.1 *FRONT PANEL*

Front Panel muncul ketika proyek LabVIEW dimulai. Nama awalnya ketika muncul adalah *Untitled 1 Front Panel* (lihat Gambar 2.9).



Gambar 2.9 Tampilan Front Panel LabVIEW
(Diambil pada 24 September 2018 pukul 08.52)

Bagian ini menampilkan *controls* (seperti *knob, button, graph*, dan lainnya) dan mewakili antarmuka grafis dari VI. Di antara *controls* tersebut, beberapa digunakan untuk melakukan proses pengambilan data (*parameter values*) dan menampilkan nilai dalam bentuk indikator angka dan grafik. Dalam praktisnya, tidak selamanya kendali hanya berperan sebagai kendali saja, namun dapat juga difungsikan sebagai indikator. Contoh, sebuah *knob*, selain dapat digunakan untuk mengatur nilai variabel tertentu, juga dapat difungsikan sebagai indikator sebuah nilai dari variabel (lihat Gambar 2.10). Caranya adalah dengan mengubah fungsi dari objek tersebut. Dengan begitu, maka fungsi objek akan berubah.

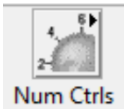
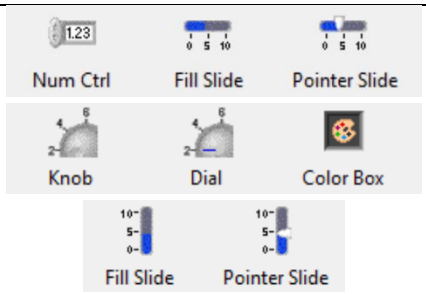
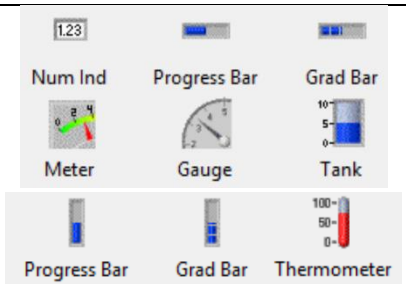
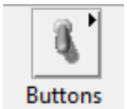
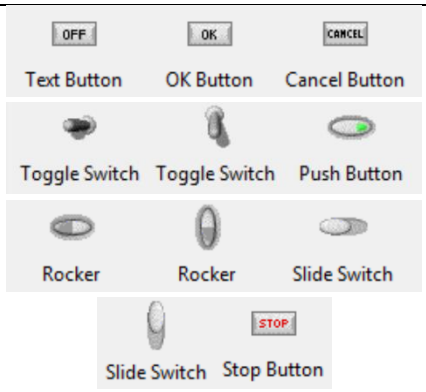


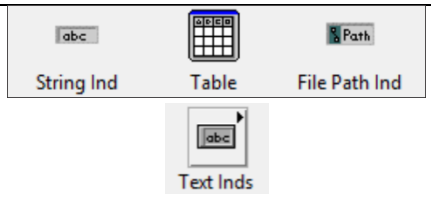
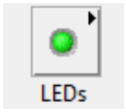
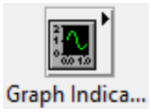
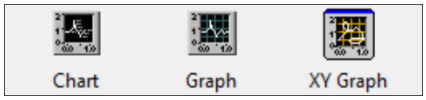
Gambar 2.10 Contoh Perubahan Fungsi Objek Kendali ke Indikator
(Diambil pada 26 September 2018 pukul 09.15)

Adapun indikator, biasanya digunakan sebagai kendali pasif, dimana mempengaruhi kinerja kendali dengan hasil atau nilai dari indikator itu sendiri. Contoh, sebuah *string control*, dapat mempengaruhi nilai keluaran sebuah variabel ketika nilai dari sebuah *string ind* dihubungkan ke rumus melalui *Diagram Block* (nanti akan dijelaskan di bagian *Diagram Block*). Selain dari *string ind*, juga dapat digunakan objek yang lain seperti *LEDs* (boolean) atau *Graph* (data per iterasi).

Untuk mempermudah dalam mengetahui beberapa jenis *controls* yang ada pada LabVIEW (versi 2010), maka penjelasan singkatnya ada pada Tabel 3 berikut ini.

Tabel 3. Macam-macam controls bawaan pada LabVIEW 2010 dan fungsinya
(Gambar diambil pada tanggal 28 September 2018)

Controls Palette (*Express Group*)	Jenis-jenis *Control*	Fungsi
		Menentukan/mengendalikan nilai yang bersifat angka/numerik untuk sebuah proses/keluaran
		Menampilkan nilai yang bersifat angka/numerik dari sebuah proses/keluaran
		Menentukan/mengendalikan nilai yang bersifat boolean untuk sebuah proses/keluaran. Dapat difungsikan sebagai indikator pula, pada beberapa kasus.
		Menentukan/mengendalikan nilai yang bersifat *string* untuk sebuah proses/keluaran

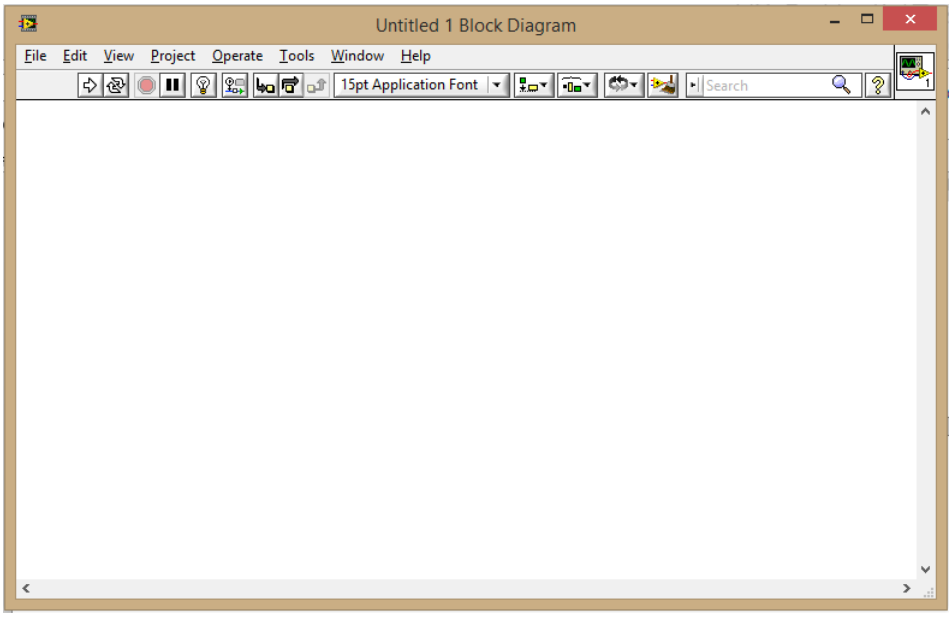
		Menampilkan nilai yang bersifat *string* dari sebuah proses/keluaran
		Menampilkan nilai yang bersifat boolean dari sebuah proses/keluaran (misal LED menyala untuk kondisi benar dan LED mati untuk kondisi salah)
		Menampilkan nilai yang bersifat angka/numerik dalam bentuk grafik

Controls di atas merupakan jenis yang paling sering digunakan dalam banyak aplikasi. Sebenarnya, ada banyak *controls* yang bisa diakses pada *Controls Palette* seperti *I/O* (komunikasi data masukan-keluaran), *Containers* (sejenis *grouping* untuk *control*), dan lainnya. *Controls* tersebut berada di *Controls Palette*, tepatnya di bagian *Express Group*. *Controls Palette* dapat diakses melalui menu *View* (bisa dilihat di gambar sebelumnya, Gambar 2.9 atau 2.10) atau klik kanan pada layar *Front Panel*.

2.11.2 *BLOCK DIAGRAM*

Block Diagram muncul dengan tampilan layar putih, meliputi VI dan struktur yang digunakan untuk mengendalikan objek pada *Front Panel*. *Block Diagram* terdiri dari *graphical source code*, yaitu sebuah kode sumber berbentuk grafis, yang lebih dikenal sebagai *G code* atau kode blok diagram, untuk menjalankan VI. Pada struktur tersebut, terdapat elemen-elemen pemrograman (seperti blok, fungsi, atau sub VI) yang saling terhubung untuk membangun program berbasis grafis (*graphical source code*). Maksudnya adalah jenis pemrograman ini menggunakan ikon grafis yang merepresentasikan fungsi-fungsi dari dan atau untuk mengendalikan objek pada *Front Panel*.

Berikut ini adalah Gambar 2.11, yang merupakan tampilan awal *Block Diagram* dengan judul *Untitled 1 Block Diagram* (tampilan awal sebelum disimpan dan dilakukan perancangan).



Gambar 2.11 Tampilan Awal Block Diagram *LabVIEW*
(Diambil pada 24 September 2018 pukul 08.54)

Untuk memudahkan dalam memahami hal tersebut, berikut ini adalah Gambar 2.12 yang merupakan contoh hasil perancangan VI dan fungsi pada *Block Diagram*.

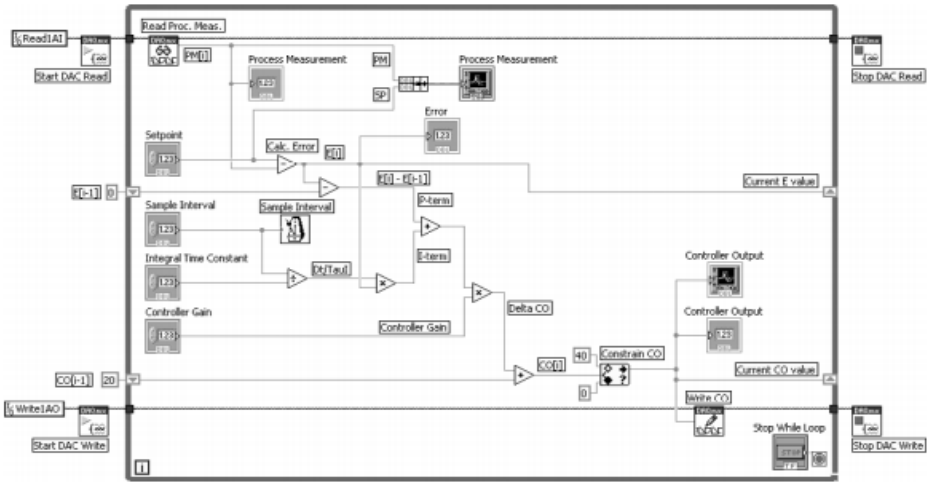


Figure 1.2
PI Controller VI, block diagram.

Gambar 2.12 Contoh hasil perancangan VI dan fungsi pada Block Diagram *(Larsen, 2011)*

Dari objek-objek yang terhubung menggunakan *wires* (garis penghubung antar objek) di atas, maka data mengalir melaluinya: dari *controls* kepada beberapa VI dan fungsi, dari beberapa VI dan fungsi ke indikator-indikator, dan dari beberapa VI dan fungsi ke beberapa VI dan fungsi lainnya. Proses pergerakan data dari satu objek ke objek lainnya menentukan perintah eksekusi dari VI dan fungsi. Hal ini biasa disebut sebagai *dataflow programming* (Larsen, 2011).

2.11.3 *GLOBAL VARIABLE*

Global Variable digunakan untuk mengakses dan meloloskan data melalui beberapa VI yang berjalan secara simultan. Fitur ini merupakan salah satu objek LabVIEW yang bersifat *built-in*, yaitu telah tersedia dari awal aplikasi diinstal. Saat fitur ini dibuat, LabVIEW secara otomatis akan membuat sebuah VI khusus, yang memiliki *Front Panel* namun tidak dengan *Block Diagram*-nya. Kontrol dan indikator ditambahkan ke *Front Panel* untuk mendefinisikan tipe-tipe data yang didapatkan oleh *Global Variable*, sehingga *Front Panel* digunakan sebagai pembawa data dari beberapa VI yang dapat mengakses data tersebut.

Logo ikon dari *Global Variable* adalah  . Objek ini tersedia di *Controls Pallette* ketika membuka *Block Diagram*. Tampilan *Front Panel* akan muncul ketika logo tersebut diklik dua kali. Objek ini dapat dibuat menjadi beberapa satuan VI, dimana setiap *Global Variable* hanya memiliki satu *Front Panel*, atau untuk mendapatkan sekumpulan variabel yang sejenis dalam satu *Global Variable*, maka dibuat satu *Global Variable* namun memiliki banyak *Front Panel*. (National Instruments, 2017)

2.11.4 NI-VISA

VISA, yang memiliki kepanjangan *Virtual Instrument Software Architecture*, merupakan sebuah standar untuk menentukan, memprogram, dan mencari permasalahan dari sistem instrumentasi melalui antarmuka GPIB, VXI, PXI, Serial, Ethernet, dan/atau USB. VISA menyediakan antarmuka pemrograman antara piranti

keras dan piranti lunak yang dikembangkan seperti LabVIEW, LabWindows/CVI, dan Measurement Studio (Microsoft Visual Studio). NI-VISA merupakan implementasi dari standar I/O VISA yang dikembangkan oleh National Instruments. NI-VISA meliputi *libraries*, *interactive utilities*, dan konfigurasi program. (National Instruments, 2014).

Gambar 2.13 Aplikasi VISA
(*https://www.ni.com/visa/*. Diakses pada 1 Oktober 2018 pukul 09.14)

VISA memiliki sebuah arsitektur yang terdiri dari dua komponen VISA yang utama, yaitu VISA *resource manager* dan VISA *resources*. VISA *resource manager* berfungsi untuk mengatur *resources* atau sumber-sumber yang digunakan VISA. Manajemen ini meliputi kemampuan untuk mencari *resources* dan membuka masing-masing sesi (proses pembacaan) *resources*. VISA *resources* terdiri dari sumber-sumber tertentu yang dapat diproses dalam *Block Diagram*. Sumber yang dimaksud adalah jenis antarmuka yang dapat berhubungan dengan LabVIEW (GPIB dan lainnya).

Dalam penerapannya, VISA memiliki beberapa langkah yang harus dilakukan, antara lain:

a. Membuka sesi untuk sebuah *resource* (sumber antarmuka);
b. Melakukan segala pengaturan pada *resource* (*baut rates*, *termination character*, dan lainnya);

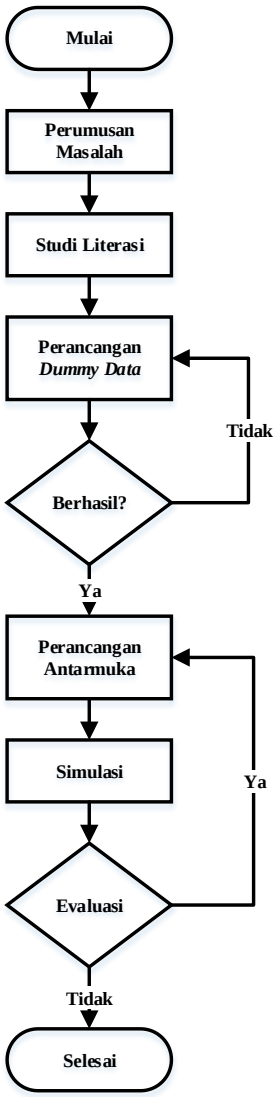
c. Melakukan *writes* (pemberian nilai) dan *reads* (pembacaan nilai) pada perangkat;
d. Menutup sesi *resource*;
e. Mengendalikan setiap eror yang kemungkinan terjadi.

Dari proses yang sudah disampaikan di atas, dapat disederhanakan ke dalam bentuk diagram blok agar mudah untuk dipahami. Berikut ini adalah Gambar 2.14, yang merupakan diagram blok dari proses aplikasi VISA.

Gambar 2.14 Diagram blok aplikasi VISA
(*http://www.ni.com/tutorial/3702/en/*. Diakses pada 3 Oktober 2018 pukul 13.58)

BAB III
METODE PENELITIAN

Dalam pengerjaan tugas akhir ini, dibutuhkan sebuah metode penelitian untuk menentukan langkah-langkah yang perlu dilakukan. Selain itu, metode penelitian ini akan mempermudah pembaca dalam memahami proses pengerjaan tugas akhir ini. Adapun langkah-langkah yang perlu dilakukan adalah sebagai berikut.



Gambar 3.1 Diagram alir metode penelitian
(Dibuat pada 31 Maret 2019 pukul 08.00)

Gambar 3.1 menjelaskan bagaimana pengerjaan tugas akhir dilakukan. Diawali dari mengangkat permasalahan yang ada, kemudian studi literasi, analisis dan pengembangan, simulasi, dan evaluasi. Proses evaluasi menentukan selesai atau tidaknya pengerjaan tugas akhir. Ketika masih banyak evaluasi yang belum terpecahkan, maka kembali ke proses analisis dan pengembangan. Hal ini akan berulang terus hingga evaluasi mencapai apa yang diinginkan dalam tugas akhir ini.

3.1 PERUMUSAN MASALAH

Langkah yang paling pertama adalah perumusan masalah. Hal ini dilakukan di awal, yaitu sebelum penulisan proposal tugas akhir. Permasalahan yang diangkat telah disampaikan di awal bab, pada bagian Rumusan Masalah. Dalam perumusan masalah, dilakukan pembatasan permasalahan yang diuraikan untuk memudahkan fokus pengerjaan pada tugas akhir.

3.2 STUDI LITERASI

Dalam proses tugas akhir, pasti membutuhkan proses mempelajari literasi apa saja yang diperlukan dalam melakukan penelitian. Proses ini dilakukan mulai dari awal perumusan masalah hingga penerapan ilmu pada perancangan dan pengembangan. Studi literasi dilakukan melalui banyak sumber, antara lain: buku, buku elektronik, jurnal, dan internet.

3.3 PERANCANGAN DAN PENGEMBANGAN

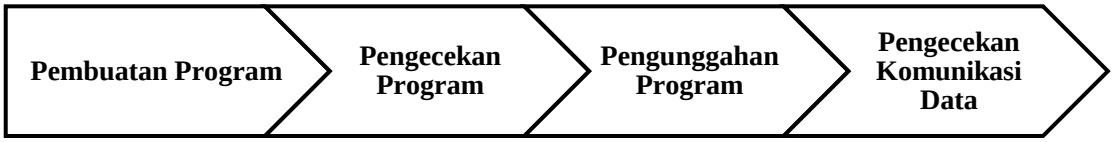
Perancangan dan pengembangan dilakukan menggunakan piranti lunak dan piranti keras, tetapi lebih dominan pada piranti lunak karena hal tersebut merupakan topik dari tugas akhir ini. Sebelum masuk ke dalam penjelasan mengenai perancangan dan pengembangan tugas akhir, berikut ini adalah alat dan bahan yang digunakan dalam perancangan tugas akhir ini.

1. Laptop Toshiba Satellite C50-B, Intel Core i3 1,8 MHz RAM 4 GB.
   Sistem Operasi: Windows 8.1 SP 1.

2. Arduino UNO.
3. Arduino IDE.
4. LabVIEW 2017.

3.3.1 PERANCANGAN *DUMMY DATA*

Pada piranti keras dilakukan pemrograman untuk menghasilkan *dummy data*. Pemrograman tersebut dilakukan menggunakan fungsi yang menghasilkan data acak, sehingga dihasilkan data ideal. Data ideal ini dirancang berdasarkan format data yang dihasilkan dari pengukuran sensor. Berikut ini adalah prosedur dalam perancangan *dummy data* menggunakan Arduino.



Gambar 3.2 Diagram alur perancangan dummy data *menggunakan Arduino*
(Dibuat pada 1 Maret 2019 pukul 08.32)

1. Membuka aplikasi Arduino IDE pada laptop.
2. Membuat program pada aplikasi Arduino IDE dengan cara mengetikkan kode program, dimulai dari inisialisasi atau deklarasi variabel, pengaturan serial dan pin Arduino, hingga program utama (Kode program terlampir).

Gambar 3.3 Tampilan program dummy data *pada Arduino IDE*
(Diambil pada 28 Februari 2019 pukul 11.22)

3. Pengaturan pin dibuat menggunakan fungsi *randomSeed()*, sedangkan program utama menggunakan fungsi *random()* untuk menghasilkan angka acak pada setiap format data. Format datanya adalah:

```
t%fp%fa%faccx%faccy%faccz%flo%fla%fcx%fcy%fcz%fgx%fgy%fgz%fhd%f
```

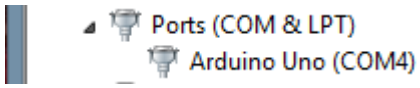
Format data tersebut diperoleh dari format pembacaan sensor yang digunakan pada penelitian sebelumnya. Hal ini kemudian dijadikan acuan dalam membuat data acak sebagai *dummy data*. Berikut ini adalah Tabel 4 yang memberikan keterangan atas format tersebut:

Tabel 4. Tabel keterangan format data untuk dummy data.

Format data	Penjelasan
`t`	Awalan untuk data **suhu**
`p`	Awalan untuk data **tekanan**
`a`	Awalan untuk data **ketinggian**
`accx`	Awalan untuk data **akselero (x)**
`accy`	Awalan untuk data **akselero (y)**
`accz`	Awalan untuk data **akselero (z)**
`lo`	Awalan untuk data **garis lintang**
`la`	Awalan untuk data **garis bujur**
`cx`	Awalan untuk data **kompas (x)**
`cy`	Awalan untuk data **kompas (y)**
`cz`	Awalan untuk data **kompas (z)**
`gx`	Awalan untuk data **giro (x)**
`gy`	Awalan untuk data **giro (y)**
`gz`	Awalan untuk data **giro (z)**
`hd`	Awalan untuk data **sudut kepala**
`%f`	Data acak yang dihasilkan

4. Setelah program selesai, kemudian program dicek terlebih dahulu, apakah masih ada kesalahan dari segi logika maupun format kode. Kemudian, ikon  (*Verify*) diklik.
5. Jika sudah keluar tulisan "*Done compiling*" pada bagian bawah Arduino IDE, maka program siap untuk diunggah ke Arduino UNO. Kemudian, ikon  (*Upload*) diklik. Namun, jika masih muncul tulisan selain "*Done compiling*", maka kode program perlu dicek kembali untuk dikoreksi kesalahannya dan kembali ke langkah ke 4.

6. Sebelum program diunggah ke Arduino, dipastikan dulu Arduino telah terhubung ke laptop melalui Serial Port. Hal ini dapat dicek melalui *Run -> Device Manager -> Ports*.



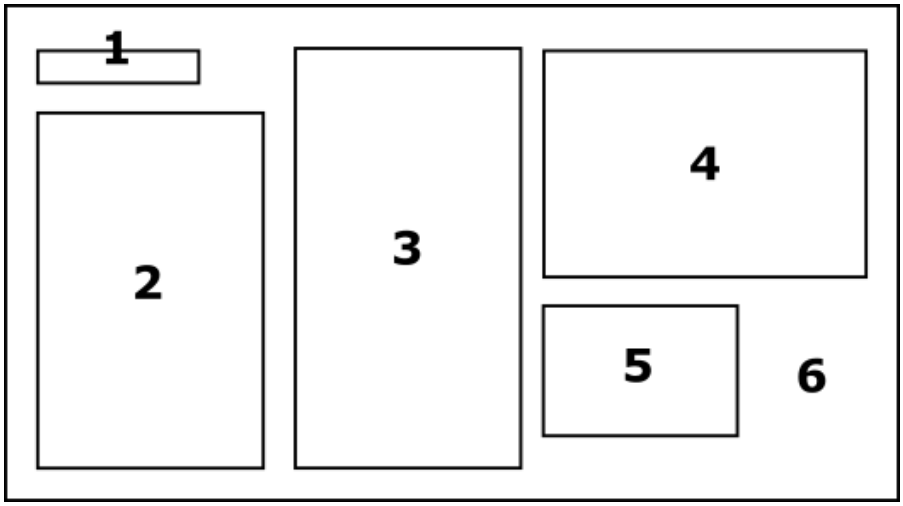
Gambar 3.4 Tampilan Arduino telah terhubung dengan laptop
(Diambil pada 28 Februari 2019 pukul 12.32)

Gambar di atas (Gambar 3.4) menunjukkan bahwa Arduino UNO telah terhubung dengan laptop melalui komunikasi serial dengan alamat COM4. Jika Arduino UNO belum terhubung ke laptop, maka laptop perlu dipasang *USB driver* Arduino terlebih dahulu.

7. Setelah program diunggah ke Arduino UNO, *Serial Monitor* dibuka untuk memastikan data telah terkirim ke serial atau belum.

3.3.2 PERANCANGAN ANTARMUKA

Dalam perancangan antarmuka (LabVIEW), diawali dari perancangan tampilan depan antarmuka. Hal ini dapat dicontoh dari antarmuka yang telah ada dari penelitian sebelumnya. Antarmuka yang dirancang merupakan pengembangan dari antarmuka yang telah ada sebelumnya. Berikut ini adalah gambaran antarmuka (*Front Panel*) yang dirancang (lihat Gambar 3.5).

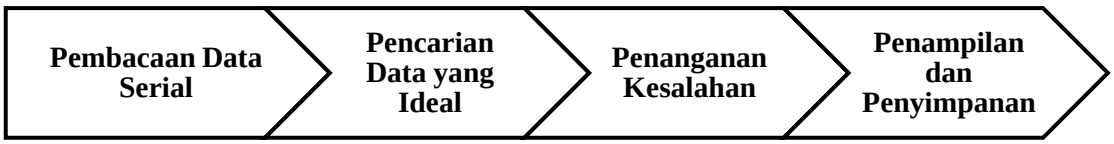


Gambar 3.5 Rencana tampilan antarmuka LabVIEW
(Dibuat pada 4 Oktober 2018 pukul 08.18)

Tampilan tersebut terbagi ke dalam 6 bagian, dimana nomor 1 hingga 5 merupakan objek. Objek yang dimaksud adalah *Controls* yang tersedia pada LabVIEW. Berikut ini keterangan komponen pada Gambar 3.5.

1. Pengaturan port komunikasi serial;
2. Indikator keluaran hasil akhir proses;
3. Indikator data per satuan paket;
4. Indikator data per segmen *string* dan waktu proses;
5. Indikator galat pada proses akhir;
6. Panel kendali.

Selain dari perancangan tampilan antarmuka, tentu dirancang pula blok diagram yang akan digunakan untuk melakukan proses pengolahan data. Blok diagram dirancang berdasarkan kebutuhan yang ada, yaitu kemungkinan kesalahan yang dibuat oleh piranti keras. Berikut ini adalah alur kerja dari blok diagram yang dirancang.



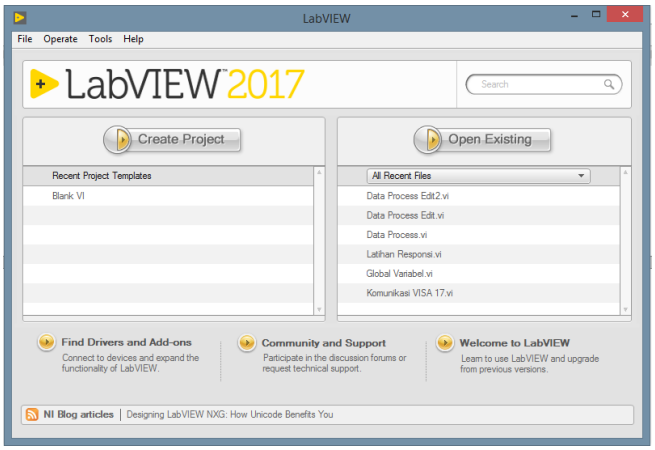
Gambar 3.6 Diagram alur perancangan blok diagram LabVIEW
(Dibuat pada 1 Maret 2019 pukul 08.45)

3.4 SIMULASI

Simulasi dilakukan setelah perancangan dan pengembangan dirasa sudah cukup siap. Tahap ini juga dapat disebut sebagai tahap uji coba antarmuka, yaitu antarmuka digunakan untuk memproses *dummy data* yang dihasilkan oleh piranti keras (mikrokontroler). Proses ini dilakukan melalui beberapa tahap, antara lain:

1. Arduino dihubungkan ke laptop.
2. Aplikasi Arduino IDE dibuka pada laptop, kemudian *Serial Monitor* dibuka dengan cara menekan ikon  (*Serial Monitor*).
3. Jika data sudah muncul pada *Serial Monitor*, maka baru bisa diuji coba ke LabVIEW.

4. NI LabVIEW 2017 dibuka, kemudian buka VI yang telah dibuat melalui bagian *Open Existing* atau di bawahnya.



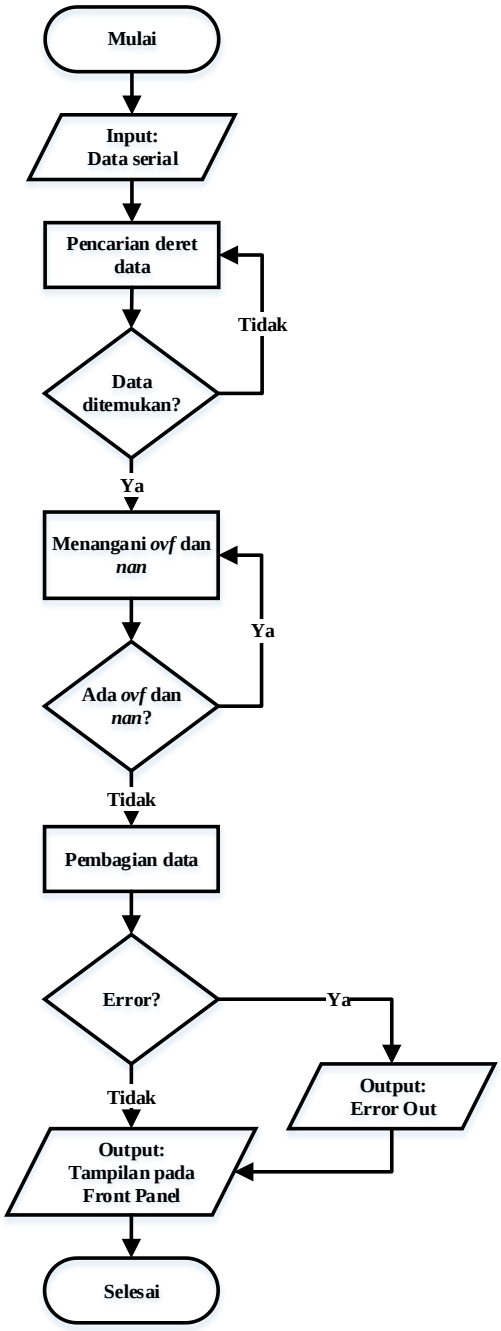
Gambar 3.7 Tampilan awal LabVIEW 2017
(Diambil pada 1 Maret 2019 pukul 10.08)

5. Apabila sudah keluar tampilan *Front Panel*, dipilih nama alamat yang tersedia (misal COM4) dari komunikasi serial antara laptop dengan Arduino. Jika belum tersedia nama alamat, maka perlu dicek kembali prosedur perancangan piranti keras.
6. Ikon (*Run*) diklik untuk menjalankan aplikasi LabVIEW secara sekali jalan, sedangkan ikon (*Run Continously*) diklik untuk menjalankan aplikasi LabVIEW secara berulang.

Dalam proses pengambilan data dari hasil simulasi menggunakan LabVIEW, digunakan dua metode pengambilan data, antara lain:

1. LabVIEW dijalankan secara satu per satu beberapa kali untuk mendapatkan data yang bersifat kuantitatif. Kemudian, dari sejumlah percobaan yang dilakukan tersebut nantinya dilihat berapa besar akurasi yang dihasilkan dari antarmuka.
2. LabVIEW dijalankan secara kontinyu dalam interval waktu tertentu untuk mendapatkan data bersifat kualitatif. Hal ini digunakan sebagai indikator kehandalan dari antarmuka yang dibuat.

Untuk mempermudah dalam memahami langkah-langkah simulasi dan metode pengambilan data, berikut ini disajikan diagram alir melalui Gambar 3.8. Diagram alir tersebut menggambarkan proses data yang terjadi pada proses simulasi.



Gambar 3.8 Diagram alir simulasi LabVIEW
(Dibuat pada 20 Maret 2019 pukul 07.45)

Setelah dilakukan langkah-langkah di atas, maka proses simulasi telah selesai dan diperoleh hasil. Jika pada proses simulasi masih terjadi kesalahan dan belum diperoleh hasil, maka dilakukan prosedur ulang terhadap simulasi. Selanjutnya, data yang telah diperoleh dari simulasi tersebut akan dievaluasi.

3.5 EVALUASI

Setelah simulasi dilakukan, maka hasil yang diperoleh dari simulasi tersebut perlu untuk dievaluasi. Pada tahap evaluasi, data hasil dianalisis dan didiskusikan bersama dosen pembimbing. Hasil analisis tersebut kemudian dinilai, apakah sudah menjawab tujuan dari tugas akhir atau belum. Proses evaluasi dilakukan secara kualitatif dan kuantitatif. Kualitatif yaitu menilai kehandalan dari antarmuka yang telah dibuat, sedangkan kuantitatif yaitu melihat jumlah kesalahan yang terjadi dari beberapa kali pengambilan data secara satu per satu. Berikut ini adalah tahap-tahap evaluasi.

1. Pengecekan *Dummy data* yang ditampilkan pada *Serial Monitor*, apakah data sudah bisa tampil dan berjalan normal.
2. Antarmuka diidentifikasi, yaitu dengan membandingkan hasil dengan format yang sudah dirancang.
3. LabVIEW dijalankan, kemudian dilihat waktu yang dibutuhkan dalam sebuah proses *tracing data*.
4. LabVIEW diidentifikasi, apakah ada error yang menyebabkan LabVIEW berhenti berjalan.
5. Jika ada error selain di atas, yaitu error dalam proses *parsing data*, cukup dilihat posisi errornya dimana.

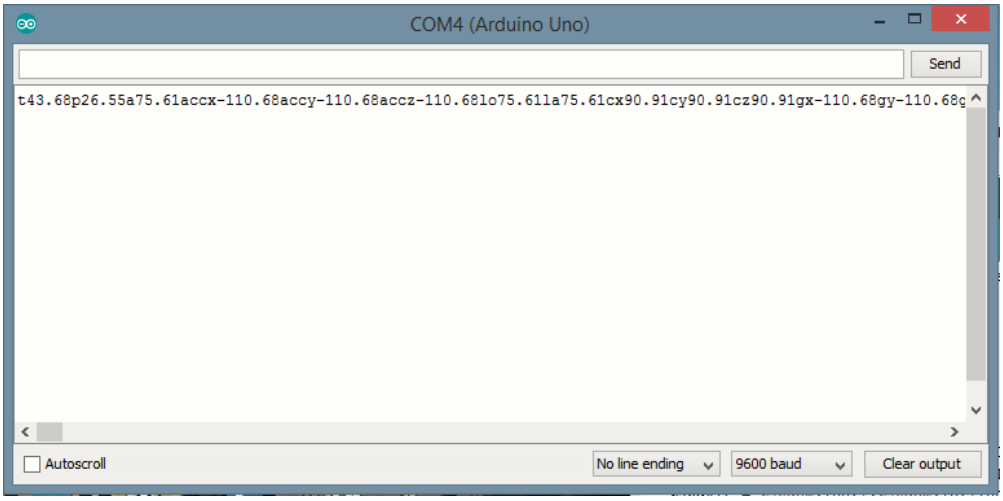
Jika hasil tersebut masih jauh dari tujuan yang ingin dicapai, maka proses penelitian akan kembali ke bagian perancangan dan pengembangan. Namun, jika hasil tersebut sudah sesuai dengan tujuan yang ingin dicapai, maka proses selanjutnya adalah pembuatan laporan akhir.

BAB IV
HASIL DAN PEMBAHASAN

4.1 HASIL

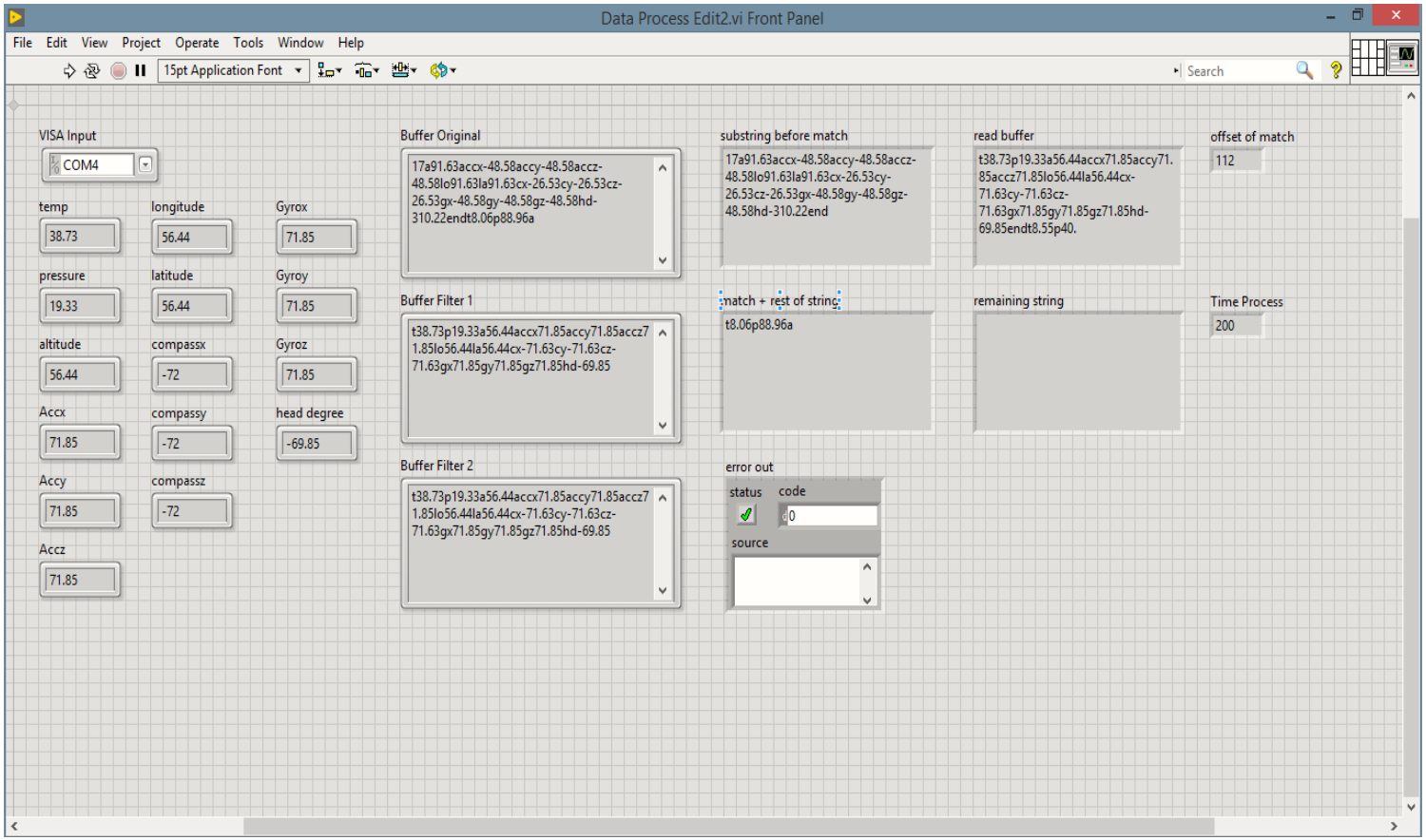
Pada sub bagian ini akan disajikan hasil dari metode penelitian yang telah dilakukan. Hasil yang disampaikan berupa gambar saja, agar mempermudah dalam penjelasan di sub bagian pembahasan nantinya. Selain itu, gambar akan lebih mudah untuk disimak.

Setelah dilakukan simulasi *dummy data* menggunakan Arduino UNO, diperoleh hasil berupa data dinamis yang terus berjalan hingga tak terhingga (selama Arduino UNO terhubung serial). Data tersebut diambil dari komunikasi COM4 antara Arduino UNO dengan laptop (Arduino IDE). Berikut ini (Gambar 4.1) merupakan hasil dari dibukanya *Serial Monitor*.



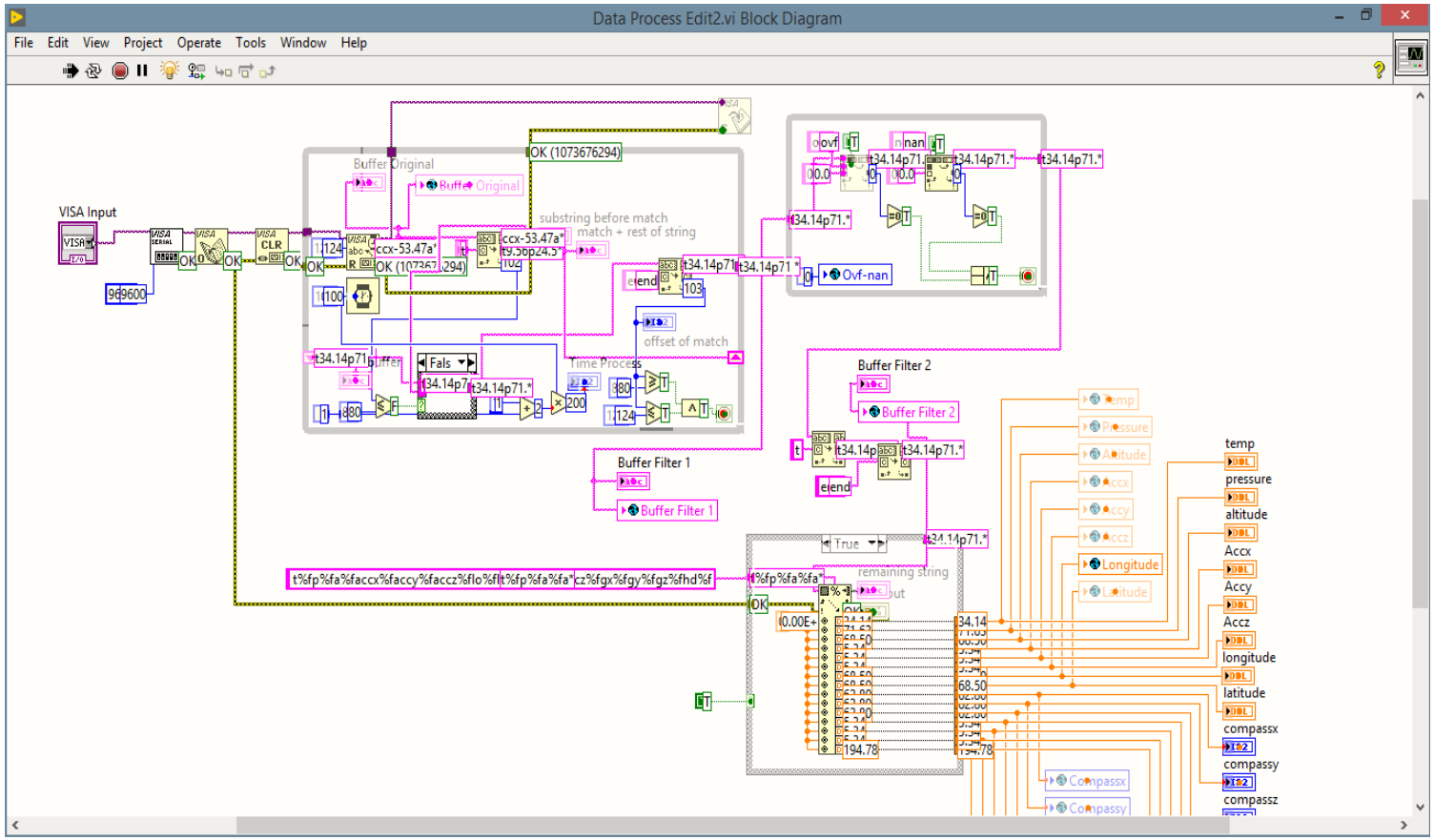
Gambar 4.1 Tampilan hasil dummy data *pada komunikasi serial*
(Diambil pada 4 Maret 2019 pukul 22.45)

Kemudian, setelah simulasi pada Arduino berhasil, simulasi dilanjutkan pada proses pengujian antarmuka yang telah dibuat menggunakan LabVIEW 2017. Setelah dilakukan beberapa kali pengembangan antarmuka, akhirnya diperoleh tampilan antarmuka yang sesuai dengan tujuan tugas akhir. Hasil tersebut dapat dilihat pada Gambar 4.2 berikut ini.



Gambar 4.2 Hasil pengolahan data melalui LabVIEW 2017
(Diambil pada 4 Maret 2019 pukul 23.03)

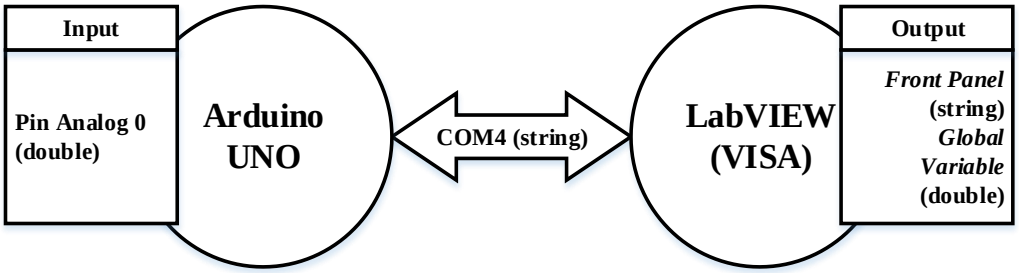
Dari *Front Panel* LabVIEW di atas, tersembunyi proses pengolahan data yang berada di bagian blok diagram. Proses ini dapat dilihat menggunakan ikon  ketika LabVIEW dijalankan. Berikut ini hasil dari proses tersebut.



Gambar 4.3 Proses pengolahan data pada blok diagram LabVIEW 2017
(Diambil pada 4 Maret 2019 pukul 23.05)

4.2 PEMBAHASAN

Pembahasan meliputi proses aliran data dari Arduino ke VISA dan aliran data dari VISA ke tampilan LabVIEW. Proses aliran data dari Arduino ke VISA diawali proses pemrograman dan menghasilkan *dummy data* yang kemudian dikirimkan ke serial sehingga dapat diterima oleh VISA. Hal ini akan dibahas lebih lanjut pada bagian 4.2.1. Selanjutnya, proses aliran data dari VISA ke tampilan LabVIEW melalui proses pengolahan data pada *Diagram Block* sehingga hasilnya dapat ditampilkan pada *Front Panel*. Hal ini akan dijelaskan pada bagian 4.2.2 dan 4.2.3. Berikut ini (Gambar 4.4) adalah blok diagram dari aliran data dari Arduino UNO ke LabVIEW, beserta tipe datanya (di dalam kurung).



Gambar 4.4 Blok diagram aliran data dari Arduino UNO ke LabVIEW
(Dibuat pada 14 Maret 2019 pukul 08.53)

4.2.1 PEMROGRAMAN DAN *DUMMY DATA*

Pertama, pemrograman mikrokontroler (Arduino) dan pengiriman *dummy data* ke komunikasi serial sebelum dibaca oleh LabVIEW melalui VISA. Hal ini dimulai dari perintah "*double randTemp, randPress, randAlt, randG, randC, randHD;*" yang merupakan sebuah inisialisasi/deklarasi variabel, yaitu menjelaskan jenis data dari variabel yang nantinya digunakan pada bagian program utama (*void loop()*). Format dari variabel-variabel di atas adalah ***double***, yaitu jenis data berjenis bilangan bulat desimal. Ada enam variabel yang dideklarasikan dengan cara dipisahkan dengan koma (,) untuk setiap variabel. Perintah deklarasi tersebut ditutup dengan titik koma (;) untuk menunjukkan bahwa program terbatas pada hal tersebut.

Kemudian, pengaturan Arduino dilakukan melalui fungsi "*void setup()*", dimana fungsi ini telah tersedia secara *default* pada Arduino IDE pada saat akan

diprogram. Kode program "*Serial.begin(9600);*" digunakan untuk mengatur besar *baud rate* pada komunikasi serial Arduino dengan laptop. *Baud rate* yang digunakan cukuplah 9600 karena besaran tersebut sudah terlalu besar untuk digunakan pada aplikasi yang kecil seperti ini. Hal ini karena pada tugas akhir ini, data yang dikirimkan per detik tidak sampai 9600 bit. Selanjutnya akan dibahas pada proses pengolahan data untuk lebih detailnya mengenai pengaruh *baud rate* terhadap kecepatan data. Terakhir, digunakan kode program "*randomSeed(analogRead(0));*" untuk menginisialisasi pin 0 Analog pada Arduino UNO untuk melakukan pembuatan angka acak.

Selanjutnya pembahasan mengenai kode program di dalam fungsi utama "*void loop()*". Kode program baris pertama pada fungsi utama atau baris ke sembilan dari seluruh kode program di Arduino IDE yaitu "*randTemp = random(50)+random(10)/100.0+random(10)/100.0;*" dan sejenisnya digunakan untuk mendefinisikan variabel tersebut dengan nilai acak yang dibuat menggunakan fungsi "*random()*". Di dalam fungsi "*random()*", dimasukkan angka untuk merepresentasikan angka acak yang berada di dalam interval 0 hingga angka tersebut (misal dimasukkan angka 100, maka angka acak yang dihasilkan adalah antara 0 hingga 100). Hal tersebut hanya memproduksi angka acak yang bersifat bilangan bulat. Untuk mendapatkan angka desimal, digunakan logika "nilai acak yang dihasilkan oleh fungsi *random()* dibagi dengan 10^x, dimana x bilangan bulat positif". Penggunaan kode "*random(10)/10.0*" menghasilkan bilangan desimal satu angka di belakang koma (0,1 – 0,9), sedangkan penggunaan kode "*random(10)/100.0*" menghasilkan bilangan desimal dua angka di belakang koma (0,01 – 0,09). Akhirnya, nilai-nilai acak itu dijumlahkan dalam satu rumus tersebut sehingga menghasilkan angka acak dengan interval 0.00 – 50.99. Untuk yang menggunakan fungsi "*random(-120, 120)*", berarti interval angka acak yang dihasilkan adalah antara -120 hingga 120.

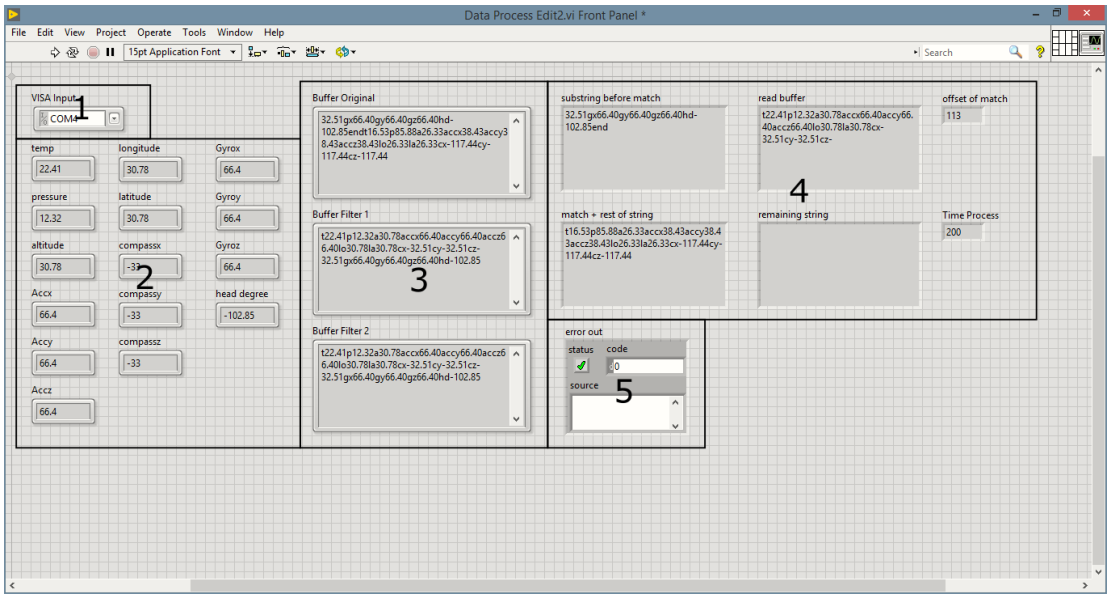
Proses yang tidak dilewatkan adalah penulisan data yang sudah dihasilkan dari pin 0 Analog tadi ke komunikasi serial. Penggunaan fungsi "*Serial.print(t)*" dan seterusnya digunakan untuk melakukan tugas tersebut. Urutan pengiriman ini disesuaikan dengan format yang data yang telah disampaikan pada bab metode

penelitian tentang perancangan piranti lunak atau program. Selain itu, untuk mengatur pola pengiriman data secara baik, maka dibuat jeda setiap 100 milisekon dengan kode program "*delay(100)*".

Seperti yang terlihat pada Gambar 4.1 (bagian hasil), hasil keluaran yang dihasilkan oleh pin 0 Analog dikirimkan ke komunikasi serial. Hasilnya adalah angka yang sangat panjang. Jenis dari data tersebut adalah ***string***. Ini disebabkan dari penggunaan kode program "*Serial.print()*", bukan "*Serial.println()*". Maksud dari "*println*" adalah *print line* atau cetak per baris.

Data memang dibuat tidak per baris, berdasarkan penelitian sebelumnya. Sebenarnya, data dapat dikirim secara per baris untuk memudahkan penerimaan data pada LabVIEW. Namun, ada kemungkinan data yang diterima tidak lengkap, akibat faktor internal dari perangkat keras, umumnya akibat dari Arduino UNO. Ini pun Arduino UNO yang bukan asli, artinya produksi selain dari *Arduino CC* (Italia). Hal ini sama sekali tidak mempengaruhi kinerja VISA, karena VISA hanya berfungsi untuk mengambil nilai dari komunikasi serial (COM4), baik datanya lengkap maupun tidak.

4.2.2 TAMPILAN LABVIEW



Gambar 4.5 Bagian-bagian dari hasil Front Panel
(Diambil pada 14 Maret 2019 pukul 16.59)

Kedua, tampilan panel antarmuka melalui LabVIEW. Seperti yang telah dijelaskan mengenai perencanaan dan pengembangan tampilan LabVIEW bahwa tampilan tersebut sesuai dengan kebutuhan KOMURINDO 2019. Hasilnya dapat dilihat dari Gambar 4.2 (bagian hasil). Bagian-bagian pada tampilan panel antarmuka memiliki penjelasan masing-masing. Hal ini dapat dilihat pada Gambar 4.5 di atas.

Pertama yang dibahas adalah bagian 1, yaitu port yang digunakan dalam komunikasi ini adalah COM4. Hal ini karena COM4 merupakan satu-satunya port yang tersedia antara laptop dengan Arduino. Port dibuat menggunakan *VISA Resource Name (Silver)* yang tersedia pada *Controls Pallette* di bagian *I/O*. Fungsi ini memungkinkan memilih jenis port menggunakan *combo box*.

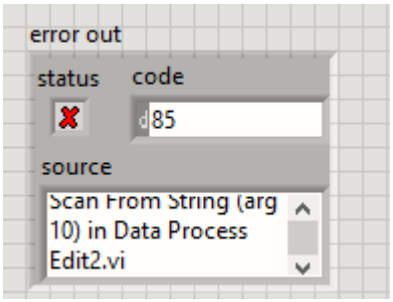
Kedua, bagian 2, yaitu tampilan hasil proses pengolahan data melalui blok diagram. Data yang ditampilkan berbentuk ***string*** karena penampil yang digunakan merupakan penampil data *string*. Penampil yang digunakan adalah *String Indicator (Silver)* yang tersedia pada *Controls Pallette* di bagian *String & Paths*.

Ketiga, bagian 3, yaitu tampilan data *buffer* yang diolah oleh blok diagram. Ada tiga indikator yang ditampilkan, yaitu Buffer Original, Buffer Filter 1, dan Buffer Filter 2. Buffer Original menampilkan data *buffer* dari serial. Kemudian, Buffer Filter 1 menampilkan data *buffer* hasil pengolahan *loop* pertama. Terakhir, Buffer Filter 2 menampilkan data *buffer* hasil pengolahan *loop* kedua (detailnya dijelaskan pada blok diagram). Bagian ini juga menggunakan *String Indicator (Silver)*, hanya saja pada bagian *visible items* – menu yang muncul ketika objek diklik kanan – dicentang pilihan *vertical scrollbar* agar terlihat fungsi *scroll* secara vertikal.

Keempat, bagian 4, yaitu tampilan data *buffer* di dalam *loop* pertama dan waktu proses dari *loop* tersebut. Tampilan data *buffer* dibagi menjadi beberapa bagian, yaitu *substring before match*, *match + rest of string*, *read buffer*, *remaining string*, dan *offset of match*. Intinya, semua objek tersebut menggunakan *String Indicator (Silver)*. Proses kerja objek-objek tersebut akan dijelaskan pada bagian proses pengolahan data.

Bagian terakhir adalah indikator error. Indikator ini menunjukkan apakah data yang diolah pada *Diagram Block* dalam kondisi utuh atau ada yang kurang dari format

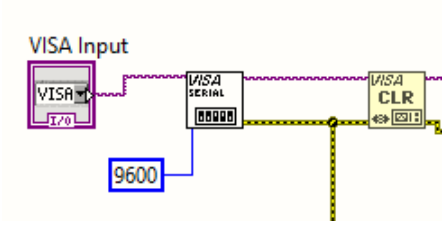
yang sudah ditentukan. Kehandalan didapatkan dari melakukan *stacking error* melalui menampilkan error pada *Front Panel* menggunakan objek *Error Out* yang tersedia pada *Pallette Controls* bagian *Array, Matrix, & Cluster*. Jika ada data yang kurang (*string* tidak lengkap), maka akan muncul indikator seperti pada Gambar 4.6. Hal ini tidak akan menghentikan proses dari LabVIEW.



Gambar 4.6 Kondisi error pada data yang diolah oleh LabVIEW
(Diambil pada 14 Maret 2019 pukul 12.25)

4.2.3 PROSES PENGOLAHAN DATA

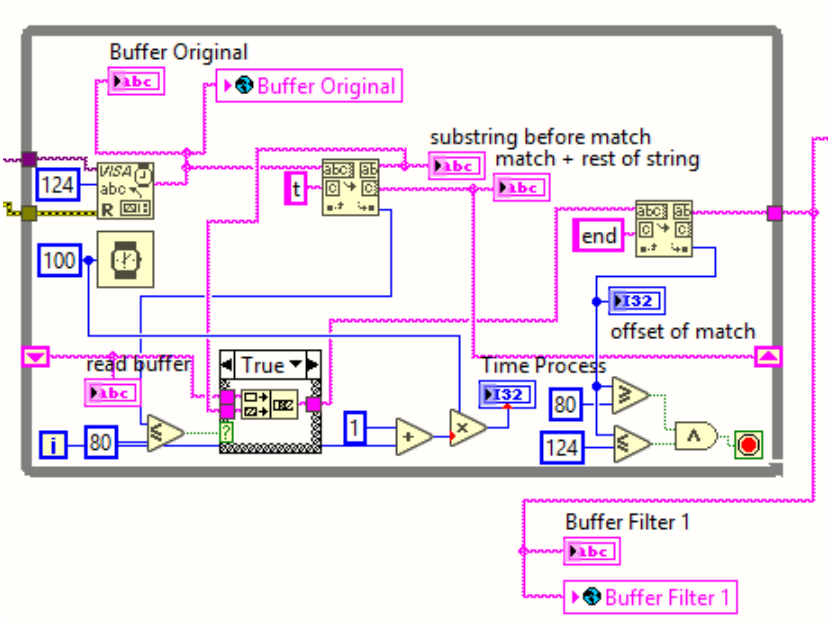
Ketiga, proses pengolahan data, yaitu mencari data (*tracing*) maupun akuisisi data melalui blok diagram di LabVIEW. Hal ini terjadi melalui enam proses. Setiap proses sangat berperan dalam pengolahan data.



Gambar 4.7 Blok diagram VISA untuk menerima data dari COM4
(Diambil pada 14 Maret 2019 pukul 21.48)

Proses yang pertama kali dilalui oleh data dari serial (COM4) adalah masuk ke LabVIEW. *VISA Serial* terhubung ke alamat serial yang sesuai dengan yang tersedia (COM4) melalui pilihan pada *Front Panel*. Besar *baud rate* ditentukan sebesar 9600 untuk membatasi besar data yang masuk ke LabVIEW. Dalam ukuran tersebut, data yang masuk harus berukuran maksimal 96.000 bit per detik dan/atau 12.000 byte. Itu merupakan ruang transfer data yang cukup besar untuk data yang dikirim melalui Arduino UNO yang hanya mengirimkan sebuah data dari satu pin saja.

Sebelum masuk ke *Loop()*, *buffer data* yang tersisa di VISA dibersihkan dulu. Hal ini didukung *VISA CLR* yang berfungsi membersihkan data *cache* yang tersisa ketika melakukan pembacaan data sebelumnya. Tanpa adanya *VISA CLR*, akan timbul error ketika pembacaan string di bagian blok diagram *tracing* data karena ada noise berupa data *cache* tersebut.



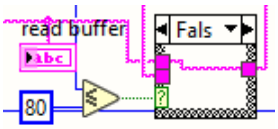
Gambar 4.8 Blok diagram pengolah buffer data yang pertama
(Diambil pada 14 Maret 2019 pukul 22.16)

Selanjutnya, data yang telah diperoleh VISA dimasukkan ke dalam *while loop()* di atas. *Loop()* tersebut hanya berulang ketika logika yang diterima bernilai ***FALSE***. Oleh karena itu, terdapat gerbang logika *AND* yang berperan menghasilkan nilai ***TRUE*** ketika besar data sudah sesuai dengan ukuran yang ditentukan yaitu antara 80 sampai 124 byte.

Proses diawali dengan mengambil data sebesar 124 byte dari VISA menggunakan *VISA Read* yang diatur *byte count*-nya. Data keluar melalui *read buffer*, dan selanjutnya ditampilkan pada *Front Panel* melalui *Buffer Original*. Selain ditampilkan, juga data dikirim ke *Global Variable*.

Search/Split String pertama digunakan untuk mencari nilai sebelum "t" dan setelah "t". Data sebelum "t" diambil melalui *substring before match*, sedangkan data

setelah “t” diambil melalui *match + rest of string*. Keduanya kemudian ditampilkan pada *Front Panel* melalui *Indicator String (Silver)*. Jika ada data sebelum “t”, maka data tersebut akan dikirim ke *Concatenate String* (berada di dalam *Case Structure*) untuk digabungkan dengan data yang tersimpan pada *shift register*. *Shift register* berperan menyimpan sementara data setelah “t” yang diproses pada iterasi sebelumnya. Hal ini bertujuan untuk mencegah adanya data yang hilang pada setiap proses. Tentu saja, proses didukung dengan objek *Wait* yang membuat iterasi bekerja teratur setiap 100 milisekon. Dengan adanya jeda, data akan menunggu untuk diproses sehingga data saling terhubung pada setiap jeda.

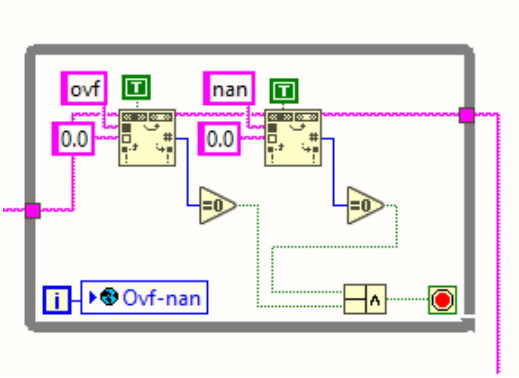


Gambar 4.9 Blok diagram Case Structure ketika posisi data lebih dari 80
(Diambil pada 15 Maret 2019 pukul 10.07)

Fungsi dari *Case Structure* adalah memastikan posisi “t” (*offset of match*) melebihi dari angka 80 atau tidak. Pada Gambar 4.8, ditampilkan *Case Structure* ketika posisi “t” masih kurang dari 80, sehingga perlu untuk digabungkan dengan data dari *shift register*. Akan tetapi, jika posisi “t” sudah kurang dari 80, maka tidak perlu digabungkan data tersebut dengan data dari *shift register* karena kemungkinan data tersebut sudah lengkap secara format. Cukup data dari *shift register* langsung diproses. Gambar 4.9 di atas menjelaskan bahwa posisi “t” sudah lebih dari 80, maka tidak perlu dilakukan penggabungan data menggunakan *Concatenate String*.

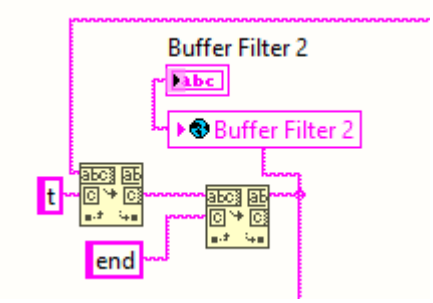
Data kemudian masuk ke *Search/Split String* kedua untuk memastikan data diakhiri dengan “end”. Data yang diambil adalah data sebelum “end” dengan fitur *substring before match* pada objek tersebut. Dalam hal ini, ada kemungkinan error karena data tidak selalu cukup lengkap karena proses dari *shift register* memiliki resiko hilangnya data. Oleh karena itu, digunakan *offset of match* untuk memastikan bahwa data telah mencapai ukuran panjang yang ditentukan yaitu di atas 80 dan di bawah 124. Perbandingan digunakan sebelum gerbang *AND* untuk memastikan hal itu. Jika data masih belum sesuai dengan format tersebut, maka iterasi akan diulang kembali.

Namun, jika data sudah sesuai format, maka iterasi selesai dan data ditampilkan ke *Buffer Filter 1* dan dikirim ke *Global Variable*.



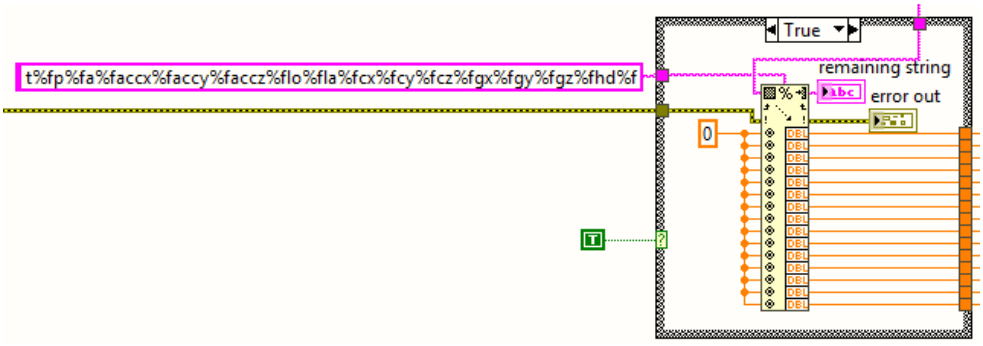
Gambar 4.10 Blok diagram pengolah buffer data yang kedua
(Diambil pada 15 Maret 2019 pukul 07.33)

Data yang sudah diolah dari *loop()* pertama kemudian masuk ke *loop()* kedua. *Loop()* ini berfungsi untuk mengantisipasi error yang terjadi ketika data berubah menjadi "ovf" atau "nan" dalam deretan string yang sudah diolah. Menurut penelitian sebelumnya, error ini terjadi karena kesalahan sensor GY-05. Logika *loop()* kedua ini masih sama dengan *loop()* pertama, yaitu akan terus berulang ketika bertemu kondisi ***FALSE***. Objek *Search and Replace String* yang digunakan di atas berfungsi mengganti komponen variabel yang mengandung unsur "ovf" dan/atau "nan" menjadi nol agar tidak membuat proses *tracing* data menjadi error. Dalam proses *tracing* data tidak dikenal format "ovf" dan "nan". Keluaran dari objek tersebut menghasilkan logika ***TRUE*** yang masuk ke gerbang logika *AND*, sehingga menghasilkan ***TRUE*** yang menyebabkan *loop()* berhenti mengulang proses dan proses berlanjut.



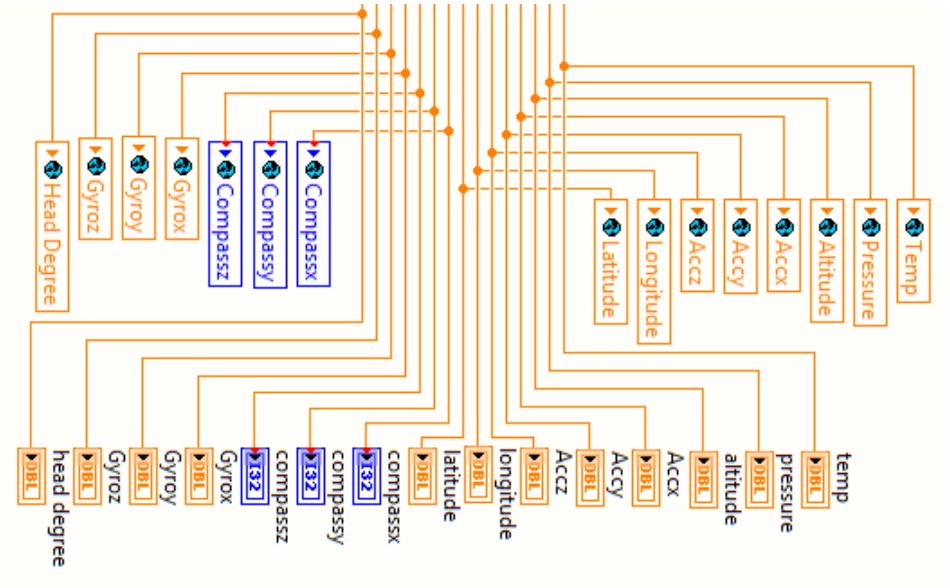
Gambar 4.11 Blok diagram pengolahan buffer data yang ketiga
(Diambil pada 15 Maret 2019 pukul 07.55)

Terdapat dua objek *Search/Split String* yang digunakan untuk melakukan *splitting* atau pencarian komponen *string* yang berawalan "t" dan diakhiri "end". Proses ini dilakukan untuk benar-benar memastikan data yang akan masuk ke proses *parsing* data per variabel di blok diagram berikutnya. Objek pertama akan meloloskan komponen *string* setelah "t" melalui fitur *match + rest of string*, sedangkan objek kedua akan meloloskan komponen *string* sebelum "end" melalui fitur *substring before match*. Setelah proses itu selesai, maka data akan dikirim ke *Global Variable* dan ditampilkan pada *Front Panel*. Selain dari itu, data lanjut ke proses *parsing*.



Gambar 4.12 Blok diagram parsing data
(Diambil pada 15 Maret 2019 pukul 08.30)

Kemudian, data diolah menggunakan objek *Scan From String* untuk mendapatkan nilai satuan dari setiap variabel. Data masuk ke objek tersebut melalui *input string*. Data tersebut dibandingkan dengan format variabel yang ditentukan pada bagian *format string*. Saat data sesuai dengan format, maka akan dibagi menjadi 15 variabel bertipe data ***double***. Jika ada variabel yang kurang, maka otomatis data pada variabel tersebut dan setelahnya di-nol-kan. Saat terjadi error, error langsung ditampilkan melalui *error out*. Jika tidak ada penampil tersebut, proses LabVIEW akan terhenti karena error harus ditampilkan secara langsung sehingga proses harus dihentikan. Hal ini sangat mengganggu kehandalan dari proses pengolahan data yang diharapkan kontinyu tanpa halangan. Selain itu, untuk mengetahui letak error yang ada pada variabel, sisa variabel yang tidak sesuai format data langsung ditampilkan pada *Front Panel* melalui *remaining string*.



Gambar 4.13 Blok diagram penampil akhir dan Global Variable
(Diambil pada 15 Maret 2019 pukul 09.39)

Akhirnya, variabel-variabel yang diperoleh dari hasil *parsing* ditampilkan ke *Front Panel* dan dikirim ke *Global Variabel*. Tipe data dari semua variabel adalah ***double***, kecuali *compassx*, *compassy*, dan *compassz* (***int32,*** sejenis tipe data *int*). Tampilan data di *Global Variable* dapat dilihat di Lampiran.

4.2.4 ANALISIS PENGUJIAN KUANTITATIF DAN KUALITATIF

Pengujian secara kuantitatif memberikan hasil berupa empat parameter, yaitu jumlah percobaan, waktu proses, terjadi error atau tidak, dan posisi variabel yang terjadi error ketika proses *parsing data*. Berikut ini adalah Tabel 5 yang menampilkan hasil pengujian kuantitatif. Tabel di bawah ini merupakan penyederhanaan dari tabel hasil yang terlampir pada lampiran tugas akhir ini. Warna abu-abu pada sebuah baris menunjukkan adanya error pada percobaan tersebut.

Tabel 5. Hasil pengujian kuantitatif LabVIEW

Percobaan ke-	Waktu Proses (milisekon)	Error (Ya/Tidak)	Posisi Error (Variabel ke-)
1	400	Tidak	-
2	300	Tidak	-
3	100	Ya	10
4	200	Tidak	-

5	300	Tidak	-
6	300	Tidak	-
7	200	Tidak	-
8	200	Tidak	-
9	800	Tidak	-
10	300	Tidak	-
11	200	Tidak	-
12	200	Tidak	-
13	500	Tidak	-
14	200	Tidak	-
15	100	Tidak	-
16	100	Ya	4
17	700	Tidak	-
18	200	Tidak	-
19	200	Tidak	-
20	200	Tidak	-
21	200	Tidak	-
22	800	Tidak	-
23	200	Tidak	-
24	600	Tidak	-
25	200	Tidak	-
26	300	Tidak	-
27	200	Tidak	-
28	200	Tidak	-
29	100	Ya	12
30	200	Tidak	-
31	100	Tidak	-
32	200	Tidak	-
33	500	Tidak	-
34	600	Tidak	-
35	200	Tidak	-
36	100	Ya	3
37	300	Tidak	-
38	200	Tidak	-
39	500	Tidak	-
40	200	Tidak	-
41	100	Ya	10
42	200	Tidak	-
43	200	Tidak	-

44	200	Tidak	-
45	100	Tidak	-
46	200	Tidak	-
47	500	Tidak	-
48	200	Tidak	-
49	200	Tidak	-
50	200	Tidak	-

Keterangan dari Tabel 5 adalah sebagai berikut.

1. Kolom pertama meliputi jumlah pengambilan data.
2. Kolom kedua meliputi waktu proses dari *while loop* pertama (proses pencarian deret data). Waktu proses dalam satuan milisekon (ms).
3. Kolom ketiga meliputi terjadi atau tidaknya error setelah proses *parsing data* atau pembagian data menjadi lima belas variable yang akan ditampilkan ke *Front Panel*.
4. Kolom keempat meliputi posisi error dari deret data ketika dilakukan proses *parsing data*. Posisi ini terdeteksi ketika data yang masuk ke *Scan From String* ada yang tidak sesuai dengan format yang diberikan ke *Scan From String*. Angka yang muncul itu merupakan variabel ke-X yang mulai tidak sesuai dengan format yang diberikan.

Kemudian, dari tabel di atas dapat dihitung error yang terjadi dan waktu rata-rata proses yang terjadi pada *while loop* pertama. Berikut ini adalah proses perhitungannya.

$$t_{rata-rata} = \frac{total\ waktu\ proses}{jumlah\ datum} = \frac{13500\ ms}{50} = 270\ ms$$

$$error = \frac{jumlah\ error}{jumlah\ datum} = \frac{5}{50} = \frac{1}{10} = 10\%$$

Ketika diujikan, proses data berjalan cukup cepat sesuai kebutuhan lomba (data terbaca per detik sudah cukup baik). Seperti pada tabel di atas, dapat diperoleh rata-rata data terbaca dari 50 kali pengambilan data adalah 270 milisekon. Proses data berjalan dengan waktu minimal 100 milisekon dan waktu maksimal 800 milisekon. Dari 50 kali pengambilan data tersebut, hanya terjadi lima kali kesalahan (error sebesar

10% dari total pengujian) berupa kehilangan data. Ada **kemungkinan** ini terjadi akibat kesalahan dari Arduino UNO, karena LabVIEW hanya berperan dalam menerima data saja. Hal ini masih sekedar hipotesis saja karena belum dilakukan pengujian antara Arduino UNO yang asli (buatan Italia) dan yang palsu (contoh buatan Cina).

Terakhir, pengujian kualitatif dilakukan dengan cara menjalankan LabVIEW selama 10 menit. Dari pengujian ini, diperoleh hasil tiga kali error yang ditampilkan pada *Error Out*. Tidak ada error yang bersifat menyebabkan LabVIEW berhenti dalam melakukan proses data. Namun, ketika pengujian dilakukan di atas 10 menit, yaitu 15 menit, laptop justru menjadi *Blue Screen of Death* atau *BSoD* (sebuah tampilan error pada Windows) ketika memasuki lima menit akhir. Hal ini terjadi karena program LabVIEW dijalankan secara *run continously* (ikon ). Windows 8.1 tidak mampu menerima proses simulasi selama itu, sehingga akhirnya Windows menampilkan *blue screen*. Oleh karena itu, pengujian hanya dilakukan dalam interval waktu 10 menit saja.

BAB V
PENUTUP

5.1 KESIMPULAN

Sebuah tugas akhir pada akhirnya selesai dan dapat diambil beberapa kesimpulan dari pembahasan yang telah disampaikan sebelumnya. Hal tersebut merupakan jawaban dari tujuan yang telah disusun di awal penulisan tugas akhir ini. Simpulan-simpulan tersebut antara lain:

1. *Dummy data* dapat dihasilkan melalui Arduino UNO untuk menunjang proses simulasi akuisisi data. *Dummy data* dihasilkan melalui pin Analog yang diatur sebagai *randomSeed()*. Selain itu, *dummy data* dapat disesuaikan dengan format data yang digunakan.
2. Dalam pengujian yang bersifat kualitatif, antarmuka yang dihasilkan menggunakan LabVIEW 2017 cukup handal dan cepat ketika dijalankan secara kontinyu pada interval waktu 10 menit. Kinerja LabVIEW tidak berhenti ketika terdapat error terjadi. Error yang terbaca pada *Error Out* adalah sebanyak 3 kali saja.
3. Dalam pengujian yang bersifat kuantitatif (sebanyak 50 kali), kesalahan yang timbul dari kurangnya data ketika dibaca LabVIEW masih bisa ditolerir, yaitu sebesar 10%. Kemudian, waktu sekali proses dari LabVIEW cukup cepat yaitu rata-rata 270 milisekon. Proses pengolahan data berkisar dari 100 milisekon hingga 800 milisekon.
4. Data tersimpan sementara pada *Global Variable*, sehingga data hilang setelah ada data baru yang masuk ke *Global Variable*.
5. Pengujian pengolahan data menggunakan antarmuka LabVIEW 2017 berhasil dan sesuai kebutuhan riset tim Komurindo UMY 2018-2019.

5.2 SARAN

Dalam pembuatan tugas akhir ini terdapat banyak sekali kekurangan yang disadari maupun tidak disadari. Saran merupakan cara untuk mendapatkan kemudahan dalam riset dan pengembangan antarmuka bidang roket muatan menggunakan LabVIEW di masa depan. Oleh karena itu, berikut ini adalah beberapa saran yang dapat disampaikan.

1. Antarmuka perlu dikembangkan lagi pada segi *buffer data* dan penanggulangan error yang muncul pada saat hilang data.
2. Diperlukan pengujian secara utuh dengan *Ground Control System* yang telah dikembangkan sebelumnya.
3. Hasil pada LabVIEW diekspor menjadi program yang dapat dieksekusi (*.exe*) agar menanggulangi permasalahan yang menyebabkan terjadi *blue screen* pada laptop/komputer.
4. Perlu dicari tahu mengenai apa penyebab data yang hilang dari Arduino UNO.
5. Tampilan GPS belum tersedia.
6. Perbanyak literatur mengenai komunikasi data dan akuisisi data.
7. Perbanyak menuliskan setiap perkembangan dari riset yang telah dilakukan sebelumnya, agar dapat digunakan sebagai referensi penulisan di kemudian hari.

DAFTAR REFERENSI

Chou, Jung-Chuan; Chen, Jie-Ting; Liao, Yi-Hung; dkk (2016) "*Wireless Sensing System for Flexible Arrayed Potentiometric Sensor Based on Xbee Module*", Jurnal IEEE Vol. 16, No. 14, 15 Juli 2018. *https://ieeexplore.ieee.org/ document/ 7471476/*. Diakses pada 5 September 2018 pukul 08.14.

Hariyanto, Didik; Yatmono, Sigit; Nugraha, Ariadie Chandra (2014) "Pengembangan Sistem Telemetri antara *Payload* Roket dan *Ground Segment*", Jurnal Seminar Nasional Pendidikan Teknik Elektro 2014. Yogyakarta: Universitas Negeri Yogyakarta.

Kurniawan, Reza; Nugroho, Tri Andi; Nugroho, Damai Bowo (2017) "Proposal KOMURINDO 2018-2019: Kategori Wahana Sistem Kendali". Tidak dipublikasikan.

Larsen, Ronald W. (2011) "*LabVIEW for Engineers*" New Jersey: Prentice Hall melalui Pearson Education, Inc. *http://users.encs.concordia.ca~tmg/images/ 9/9b/Larsen_LabVIEW_for_Engineers_1st_txtbk.PDF*. Diakses pada 24 September 2018 pukul 09.54.

Lembaga Penerbangan dan Antariksa Nasional. (2017) "Buku Panduan Kompetisi Muatan Roket Indonesia dan Wahana Sistem Kendali – Kompetisi Muatan Balon Atmosfer 2018 - 2019". *http://komurindo-kombat.lapan.go.id/*. Diakses pada 17 Mei 2018 pukul 21.38.

Lembaga Penerbangan dan Antariksa Nasional. "Workshop Komurindo 2017: Persaingan Antar Tim Gairahkan Nuansa Kompetisi". *https://lapan.go.id/ index.php/subblog/read/2017/3585/Workshop-Komurindo-2017-Persaingan-Antar-Tim-Gairahkan-Nuansa-Kompetisi/1600*. Diakses pada tanggal 30 Juli 2018 pukul 06.18.

Mardis, Marsul Musyawwir; Mulyana, Asep; Sunarya, Unang (2013) "Desain Graphical User Interface untuk Muatan Roket sebagai Ground Stations",

sebagai tugas akhir untuk mendapatkan gelar Diploma bidang Teknik Telekomunikasi, Universitas Telkom. Bandung: Universitas Telkom.

Mujaahid, Faaris; Hizbullah, Amir Malik; Syahfitra, Febrian Dhimas; dkk (2017) "*Development of User Interface Based on LabVIEW for Unmanned Aircraft Application*", Jurnal Teknologi Elektro Universitas Muhammadiyah Yogyakarta (JET-UMY), Vol. 1, No. 2, Juni 2017. Yogyakarta: Universitas Muhammadiyah Yogyakarta.

National Instruments. (2013) "*Getting Started by LabVIEW*". *http://www.ni.com/pdf/manuals/373427j.pdf*. Texas: National Instruments Corporation.

National Instruments. (2019) "*Global Variable*". *http://zone.ni.com/reference/en-XX/help/371361P-01/lvconcepts/glob_variables/*. Diakses pada 13 Maret 2019 pukul 13.21.

National Instruments. (2018) "*NI-VISA Overview*". *http://www.ni.com/tutorial/3702/en/*. Diakses pada 1 Oktober 2018 pukul 08.43.

Pálsson, Þórarinn Árni. (2017) "*Drone Autopilot*", sebagai proyek akhir untuk mendapatkan gelar *Bachelor of Science* di bidang *Applied Electrical Engineering*, Universitas Reykjavic, Islandia. Diakses pada 4 September 2018 pukul 21.52.

Shirathal Mustaqim – Jalan yang Lurus. "Tafsir At Thabari, Isi Tafsir Al Qur'an: Surat Al Imran 190-191". *http://www.shirathal-mustaqim.org/index.php/quran/view_tafsir_ind?idb=271*. Diakses pada 19 September 2018 pukul 09.03.

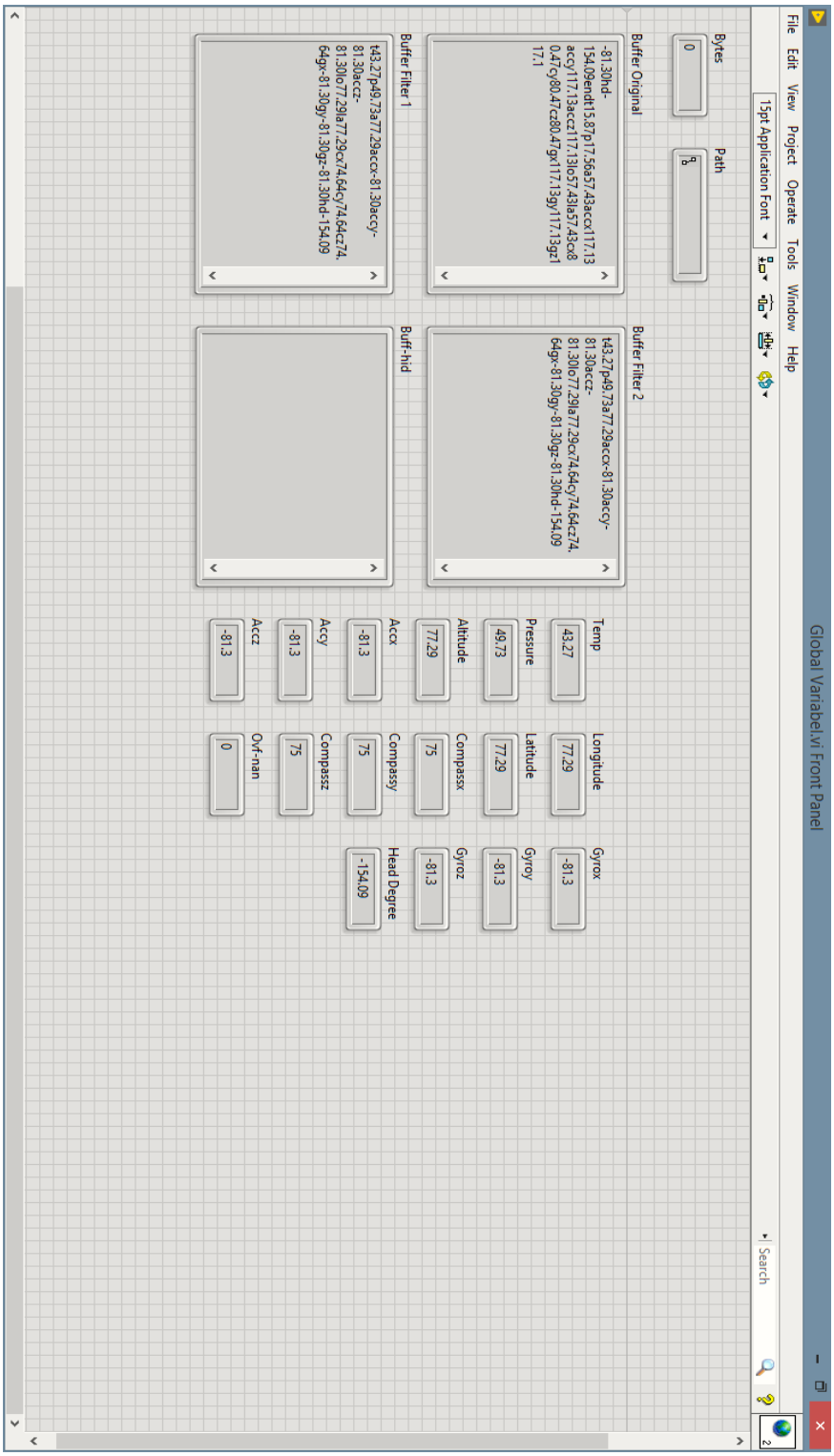
Sugimiyanto. (2014) "Membangun Aplikasi *Ground Control Station* sebagai Visualisasi Posisi Global Muatan Roket (*Payload*) di Bumi pada Kompetisi Muatan Roket Indonesia", sebagai tugas akhir untuk mendapatkan gelar Strata Satu di bidang Sistem Informasi, Fakultas Teknik dan Ilmu Komputer. Bandung: Universitas Komputer Indonesia.

LAMPIRAN

Lampiran 1: Kode program Arduino

```
double randTemp, randPress, randAlt, randG, randC, randHD;

void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop() {
  randTemp = random(50)+random(10)/10.0+random(10)/100.0;
  randPress = random(100)+random(10)/10.0+random(10)/100.0;
  randAlt = random(100)+random(10)/10.0+random(10)/100.0;
  randG = random(-120, 120)+random(10)/10.0+random(10)/100.0;
  randC = random(-120, 120)+random(10)/10.0+random(10)/100.0;
  randHD = random(-359, 359)+random(10)/10.0+random(10)/100.0;

//  String Data[] = {"t", randTemp, "p", randPress, "a", randAlt, "accx", randG,
//  "accy", randG, "accz", randG, "lo", randAlt, "la", randAlt, "cx", randC,
//   "cy", randC, "cz", randC, "gx", randG, "gy", randG, "gz", randG, "hd",
randHD};

  Serial.print("t");        //min: 4, max: 5
  Serial.print(randTemp);
  Serial.print("p");        //min: 4, max: 6
  Serial.print(randPress);
  Serial.print("a");        //min: 4, max: 6
  Serial.print(randAlt);
  Serial.print("accx");     //min: 7, max: 10
  Serial.print(randG);
  Serial.print("accy");     //min: 7, max: 10
  Serial.print(randG);
  Serial.print("accz");     //min: 7, max: 10
  Serial.print(randG);
  Serial.print("lo");       //min: 5, max: 7
  Serial.print(randAlt);
  Serial.print("la");       //min: 5, max: 7
  Serial.print(randAlt);
  Serial.print("cx");       //min: 5, max: 8
  Serial.print(randC);
  Serial.print("cy");       //min: 5, max: 8
  Serial.print(randC);
  Serial.print("cz");       //min: 5, max: 8
  Serial.print(randC);
  Serial.print("gx");       //min: 5, max: 8
  Serial.print(randG);
  Serial.print("gy");       //min: 5, max: 8
  Serial.print(randG);
  Serial.print("gz");       //min: 5, max: 8
  Serial.print(randG);
  Serial.print("hd");       //min: 5, max: 8
  Serial.print(randHD);
  Serial.print("end");    //total (w/out end):
                           //min: 78, max: 117

  delay(100);
}
```

Lampiran 2: Tampilan hasil *Global Variable*



Lampiran 3: Tabel hasil uji coba secara kuantitatif

Percoba-an	Data + Waktu	Error
1	t29.08p27.35a51.93accx16.92accy16.92accz16.92lo51.93la51.93cx9.65cy9.65cz9.65gx16.92gy16.92gz16.92hd-3.15 + 400 ms	
2	t46.97p49.35a82.26accx-97.36accy-97.36accz-97.36lo82.26la82.26cx18.35cy18.35cz18.35gx-97.36gy-97.36gz-97.36hd145.49 + 300 ms	
3	t33.31p90.48a73.90accx69.03accy69.03accz69.03lo73.90la73.90cx73719.15gz119.15h10gz-77.10hd257.76 + 100 ms	gz119.15h10gz-77.10hd257.76
4	t8.55p22.42a94.45accx6.99accy6.99accz6.99lo94.45la94.45cx51.14cy51.14cz51.14gx6.99gy6.99gz6.99hd-193.35 + 200 ms	
5	t32.69p69.03a89.90accx50.45accy50.45accz50.45lo89.90la89.90cx4.32cy4.32cz4.32gx50.45gy50.45gz50.45hd263.97 + 300 ms	
6	t43.95p81.29a98.44accx-117.91accy-117.91accz-117.91lo98.44la98.44cx-66.05cy-66.05cz-66.05gx-117.91gy-117.91gz-117.91hd54.48 + 300 ms	
7	t28.37p35.71a3.83accx-1.71accy-1.71accz-1.71lo3.83la3.83cx103.68cy103.68cz103.68gx-1.71gy-1.71gz-1.71hd-88.67 + 200 ms	
8	t27.95p94.97a68.91accx116.26accy116.26accz116.26lo68.91la68.91cx106.88cy106.88cz106.88gx116.26gy116.26gz116.26hd-184.85 + 200 ms	
9	t31.01p36.55a87.10accx6.39accy6.39accz6.39lo87.10la87.10cx84.23cy84.23cz84.23gx6.39gy6.39gz6.39hd-217.70 + 800 ms	
10	t35.31p80.37a6.50accx-82.51accy-82.51accz-82.51lo6.50la6.50cx48.48cy48.48cz48.48gx-82.51gy-82.51gz-82.51hd-245.08 + 300 ms	
11	t49.78p21.77a79.83accx-4.72accy-4.72accz-4.72lo79.83la79.83cx-62.69cy-62.69cz-62.69gx-4.72gy-4.72gz-4.72hd198.18 + 200 ms	
12	t33.18p62.26a44.38accx56.58accy56.58accz56.58lo44.38la44.38cx-33.38cy-33.38cz-33.38gx56.58gy56.58gz56.58hd305.23 + 200 ms	
13	t19.06p9.37a24.04accx-68.09accy-68.09accz-68.09lo24.04la24.04cx-103.72cy-103.72cz-103.72gx-68.09gy-68.09gz-68.09hd33.25 + 500 ms	

14	t46.69p36.63a55.78accx-111.31accy-111.31accz-111.31lo55.78la55.78cx-75.49cy-75.49cz-75.49gx-111.31gy-111.31gz-111.31hd318.48 + 200 ms	
15	t0.91p34.21a49.59accx114.72accy114.72accz114.72lo49.59la49.59cx62.63cy62.63cz62.63gx114.72gy114.72gz114.72hd-197.92 + 100 ms	
16	t25.71p65.31a39.93.47lo21.66la21.66cx-71.75cy-71.75cz-71.75gx93.47gy93.47gz93.47hd54.81 + 100 ms	.47lo21.66la21.66cx-71.75cy-71.75cz-71.75gx93.47gy93.47gz93.47hd54.81
17	t9.75p78.46a18.06accx49.56accy49.56accz49.56lo18.06la18.06cx97.06cy97.06cz97.06gx49.56gy49.56gz49.56hd139.71 + 700 ms	
18	t4.60p97.64a80.94accx68.05accy68.05accz68.05lo80.94la80.94cx33.28cy33.28cz33.28gx68.05gy68.05gz68.05hd-216.41 + 200 ms	
19	t8.82p23.45a18.11accx-38.36accy-38.36accz-38.36lo18.11la18.11cx99.73cy99.73cz99.73gx-38.36gy-38.36gz-38.36hd-71.55 + 200 ms	
20	t49.18p26.66a62.01accx83.70accy83.70accz83.70lo62.01la62.01cx64.55cy64.55cz64.55gx83.70gy83.70gz83.70hd173.87 + 200 ms	
21	t27.95p18.28a20.59accx-25.31accy-25.31accz-25.31lo20.59la20.59cx4.76cy4.76cz4.76gx-25.31gy-25.31gz-25.31hd35.09 + 200 ms	
22	t26.16p84.73a58.12accx66.91accy66.91accz66.91lo58.12la58.12cx-75.88cy-75.88cz-75.88gx66.91gy66.91gz66.91hd60.74 + 800 ms	
23	t36.72p46.43a14.70accx2.18accy2.18accz2.18lo14.70la14.70cx118.18cy118.18cz118.18gx2.18gy2.18gz2.18hd269.32 + 200 ms	
24	t14.41p90.21a3.56accx77.26accy77.26accz77.26lo3.56la3.56cx-42.16cy-42.16cz-42.16gx77.26gy77.26gz77.26hd-273.60 + 600 ms	
25	t29.43p76.12a40.02accx55.94accy55.94accz55.94lo40.02la40.02cx90.79cy90.79cz90.79gx55.94gy55.94gz55.94hd292.41 + 200 ms	
26	t5.72p99.36a65.40accx55.61accy55.61accz55.61lo65.40la65.40cx13.43cy13.43cz13.43gx55.61gy55.61gz55.61hd172.14 + 300 ms	
27	t2.22p58.97a2.62accx-67.57accy-67.57accz-67.57lo2.62la2.62cx62.64cy62.64cz62.64gx-67.57gy-67.57gz-67.57hd143.73	

	+ 200 ms	
28	t1.67p20.60a55.69accx99.57accy99.57accz99.57lo55.69la55.69cx103.22cy103.22cz103.22gx99.57gy99.57gz99.57hd54.71 + 200 ms	
29	t29.88p2.82a55.18accx-96.38accy-96.38accz-96.38lo55.18la55.18cx100.70cy100.70cz100.70-92.21gz-92.21hd297.74 + 100 ms	-92.21gz-92.21hd297.74
30	t46.50p13.10a0.39accx19.97accy19.97accz19.97lo0.39la0.39cx-8.34cy-8.34cz-8.34gx19.97gy19.97gz19.97hd-21.85 + 200 ms	
31	t10.65p49.15a2.47accx-85.42accy-85.42accz-85.42lo2.47la2.47cx69.21cy69.21cz69.21gx-85.42gy-85.42gz-85.42hd20gz51.20hd183.69 + 100 ms	
32	t8.13p62.02a58.42accx-2.26accy-2.26accz-2.26lo58.42la58.42cx-59.26cy-59.26cz-59.26gx-2.26gy-2.26gz-2.26hgz-97.25hd-332.66 + 200 ms	gz-97.25hd-332.66
33	t1.88p78.81a83.44accx70.85accy70.85accz70.85lo83.44la83.44cx91.46cy91.46cz91.46gx70.85gy70.85gz70.85hd-283.70 + 500 ms	
34	t3.58p85.40a46.09accx-16.98accy-16.98accz-16.98lo46.09la46.09cx73.24cy73.24cz73.24gx-16.98gy-16.98gz-16.98hd307.03 + 600 ms	
35	t48.16p73.85a88.97accx52.08accy52.08accz52.08lo88.97la88.97cx118.25cy118.25cz118.25gx52.08gy52.08gz52.08hd-200.98 + 200 ms	
36	t37.24p53.600.19lo4.81la4.81cx105.33cy105.33cz105.33gx-40.19gy-40.19gz-40.19hd253.03 + 100 ms	.19lo4.81la4.81cx105.33cy105.33cz105.33gx-40.19gy-40.19gz-40.19hd253.03
37	t39.00p96.25a79.64accx98.18accy98.18accz98.18lo79.64la79.64cx103.69cy103.69cz103.69gx98.18gy98.18gz98.18hd-226.44 + 300 ms	
38	t31.94p27.06a57.14accx-26.36accy-26.36accz-26.36lo57.14la57.14cx-115.41cy-115.41cz-115.41gx-26.36gy-26.36gz-26.36hd64.63 + 200 ms	
39	t11.90p12.20a36.13accx67.78accy67.78accz67.78lo36.13la36.13cx-110.04cy-110.04cz-110.04gx67.78gy67.78gz67.78hd207.99 + 500	
40	t24.64p59.47a51.64accx6.23accy6.23accz6.23lo51.64la51.64cx85.37cy85.37cz85.37gx6.23gy6.23gz6.23hd-207.98 + 200	

41	t43.82p83.86a51.25accx-41.41accy-41.41accz-41.41lo51.25la51.25cx-32.17gz102.17hd329.81 + 100 ms	gz102.17hd329.81
42	t2.62p44.03a42.47accx63.85accy63.85accz63.85lo42.47la42.47cx69.62cy69.62cz69.62gx63.85gy63.85gz63.85hd43.42 + 200 ms	
43	t48.83p81.36a28.02accx41.43accy41.43accz41.43lo28.02la28.02cx-106.41cy-106.41cz-106.41gx41.43gy41.43gz41.43hd268.07 + 200 ms	
44	t22.64p72.75a25.39accx-36.12accy-36.12accz-36.12lo25.39la25.39cx-110.26cy-110.26cz-110.26gx-36.12gy-36.12gz-36.12hd182.34 + 200 ms	
45	t22.64p72.75a25.39accx-36.12accy-36.12accz-36.12lo25.39la25.39cx-110.26cy-110.26cz-110.2gx104.00gy104.00gz104.00hd22.84 + 100 ms	
46	t27.75p64.84a40.28accx109.84accy109.84accz109.84lo40.28la40.28cx74.17cy74.17cz74.17gx109.84gy109.84gz109.84hd-186.53 + 200 ms	
47	t47.04p90.98a19.29accx70.44accy70.44accz70.44lo19.29la19.29cx105.61cy105.61cz105.61gx70.44gy70.44gz70.44hd295.58 + 500 ms	
48	t8.58p33.40a2.32accx-96.38accy-96.38accz-96.38lo2.32la2.32cx5.25cy5.25cz5.25gx-96.38gy-96.38gz-96.38hd189.01 + 200 ms	
49	t22.15p64.50a17.05accx-40.72accy-40.72accz-40.72lo17.05la17.05cx-49.27cy-49.27cz-49.27gx-40.72gy-40.72gz-40.72hd178.31 + 200 ms	
50	t26.15p79.04a33.99accx-51.37accy-51.37accz-51.37lo33.99la33.99cx36.88cy36.88cz36.88gx-51.37gy-51.37gz-51.37hd25.51 + 200 ms	