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.

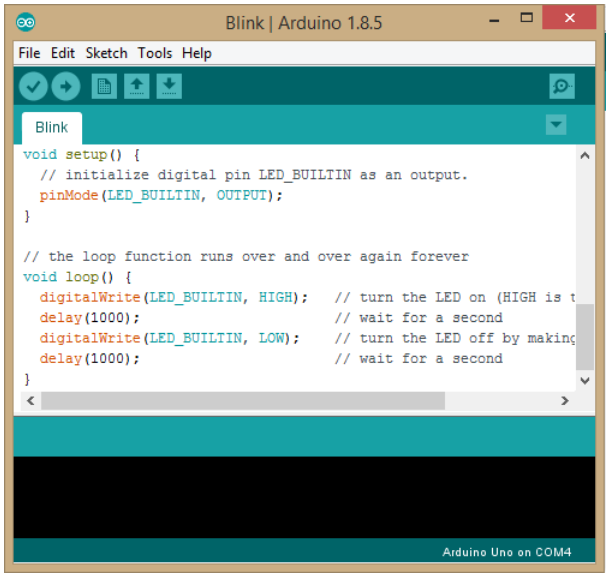
Gambar 2.7 Baterai Li-Po 1550mAh

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

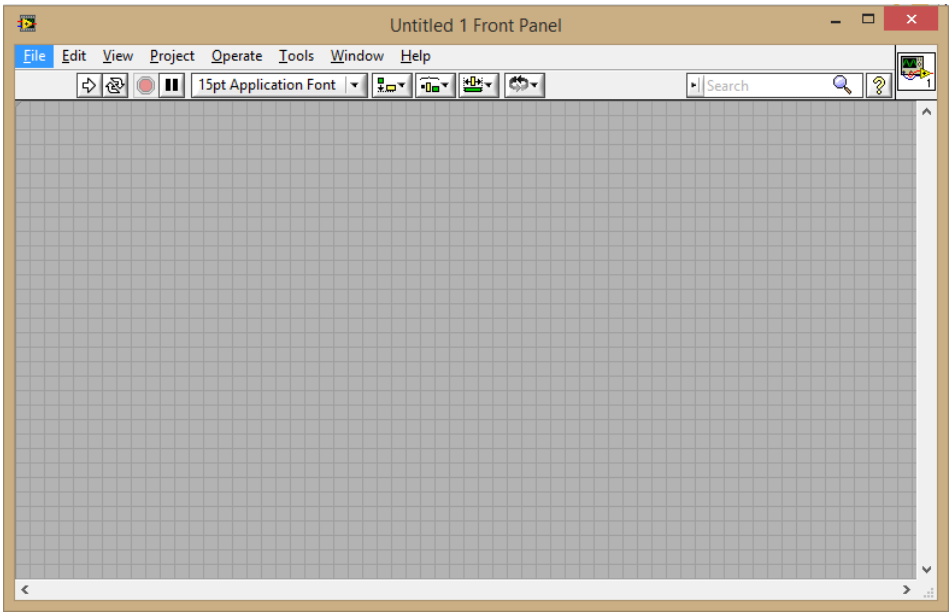
LabVIEW memiliki kepanjangan *Laboratory Virtual Instrument Engineering* Workbench. Perangkat lunak ini dikembangkan oleh sebuah perusahaan bernama *National Instruments Corporation*, yaitu sebuah perusahaan multinasional Amerika yang beroperasi secara internasional. Perusahaan ini berpusat di salah 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.

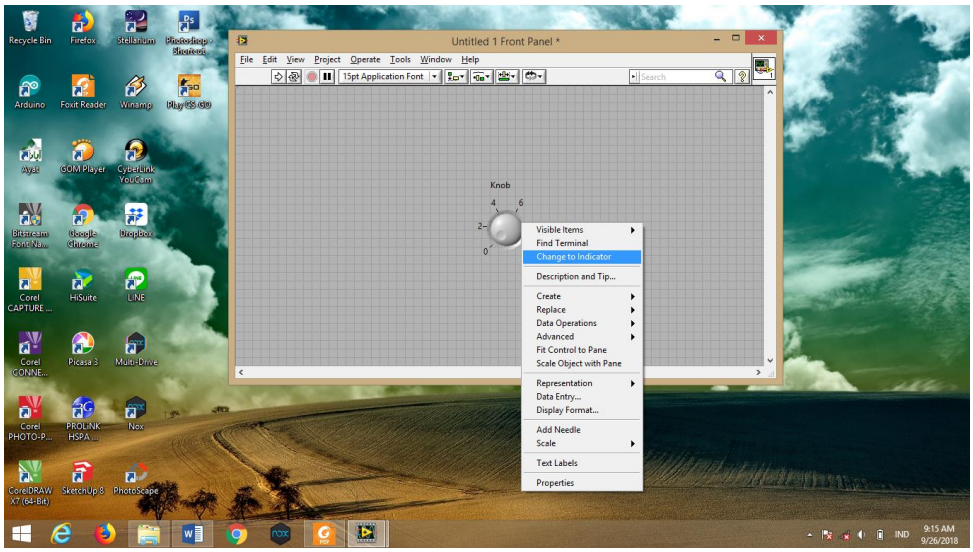
2.11.1 *FRONT PANEL*

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.

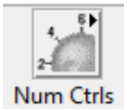
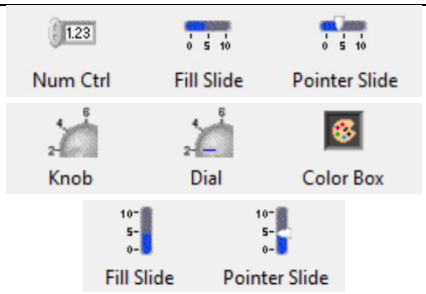
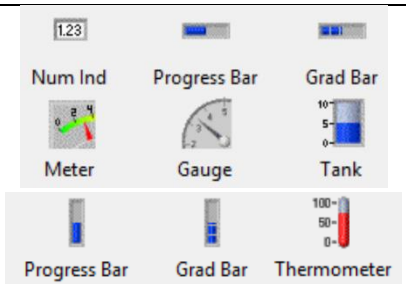
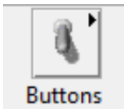
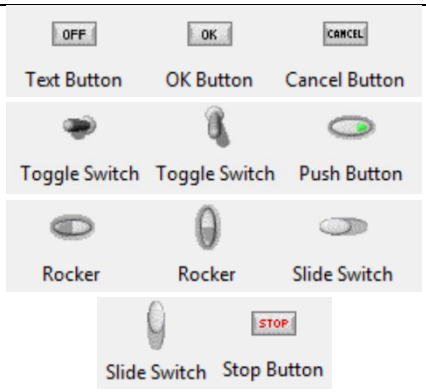


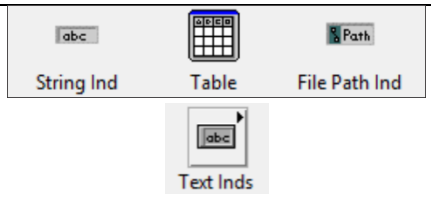
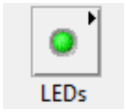
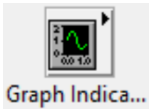
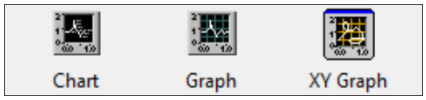
Gambar 2.10 Contoh Perubahan Fungsi Objek Kendali ke Indikator
(Diambil pada 26 September 2018 pukul 09.15)

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

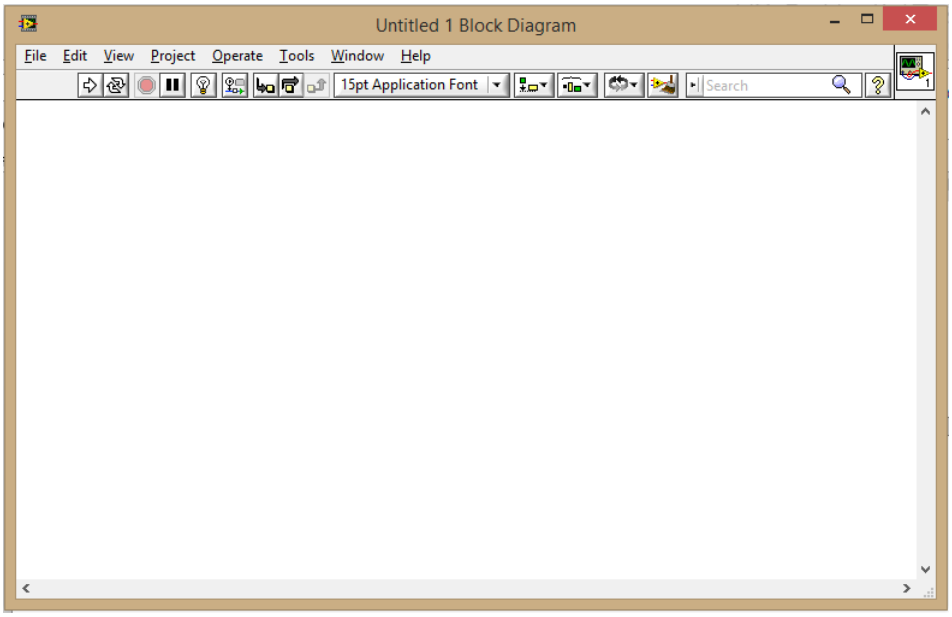
		Menampilkan nilai yang bersifat *string* dari sebuah proses/keluaran
		Menampilkan nilai yang bersifat boolean dari sebuah proses/keluaran (misal LED menyala untuk kondisi benar dan LED mati untuk kondisi salah)
		Menampilkan nilai yang bersifat angka/numerik dalam bentuk grafik

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*.

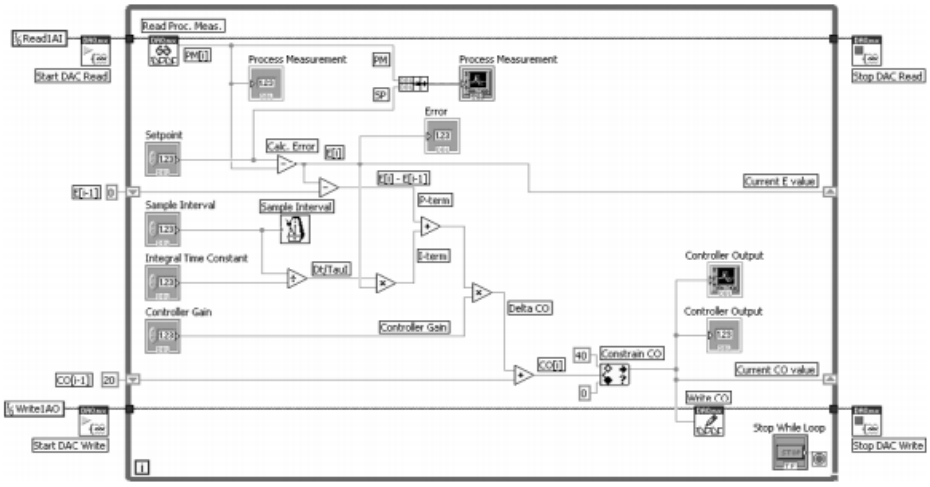


Figure 1.2
PI Controller VI, block diagram.

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)