BAB II
TINJAUAN PUSTAKA

2.1. Kajian Pustaka

(Ronisah, 2007) Menyatakan bahwa *Graphical User Interface* (GUI) adalah sebuah program aplikasi berorientasi visual yang dibangun dengan obyek grafis sebagai pengganti perintah text untuk pemakai berinteraksi.

(Stevani, 2013) Menyatakan bahwa LabVIEW merupakan bahasa pemrograman grafis yang digunakan untuk memperoleh data dari suatu instrument kemudian memproses data, menganalisa data serta dapat melakukan fungsi kontrol.

(Siswo,Munarto, & Vicky, 2013) Menyatakan pengukuran besaran fisis merupakan salah satu langkah dalam akuisisi data, dan temperatur merupakan salah satu besaran fisis yang sering digunakan dalam sistem kontrol, baik hanya untuk monitoring atau proses pengendalian lebih lanjut. Saat ini alat yang telah ada berupa sistem akuisisi data yang terhubung dengan PC (*PC-based*) atau sistem yang dapat berdiri sendiri (*portable*). Dalam kaitannya dengan hal tersebut, dibuatlah *data logger* alternatif dengan mikrokontroler ATMega8535 dan antarmuka berupa LabVIEW sebagai *data logging*. *Laboratory virtual instrument engineering workbench* merupakan bahasa pemrograman berbasis grafis atau blok, sedangkan pemrograman lainnya menggunakan *text*.

Adapun karya ilmiah ini berbeda dengan karya-karya ilmiah yang ada sebelumnya seperti telah disebutkan diatas. Dalam hal ini belum ditemukan karya ilmiah yang berkaitan dengan perancangan dan pembuatan karya ilmiah yang berkaitan dengan penerimaan objek yang bergerak seperti *radiosonde*. Sejauh ini yang ada hanyalah penggunaan perangkat lunak LabVIEW untuk instrumentasi *virtual* sebuah sensor saja.

2.2. Atmosfer

Dalam Kamus Besar Bahasa Indonesia (KBBI), atmosfer didefinisikan sebagai lapisan udara yang menyelubungi bumi sampai ketinggian 300 km

(terutama terdiri atas campuran berbagai gas, yaitu nitrogen, oksigen, argon, dan sejumlah kecil gas. Kerapatan partikel atmosfer meningkat dengan makin berkurangnya ketinggian. Massa dan tekanannya pun meningkat semakin dekat permukaan bumi. Karena bagian terbesar bahan pengisi atmosfer berada di bagian bawah, maka perubahan massa atmosfer terhadap ketinggian pada bagian bawah relative cepat. Atmosfer setinggi 5,5 sampai dengan 5,6 km telah mencakup 50% dari massa total dan pada ketinggian 40 km telah mencakup 99,99%. (Handoko 1994)

Atmosfer terdapat beberapa lapisan berdasarkan ketinggiannya, yaitu troposfer, statosfer, mesosfer, termosfer, ionosfer, dan eksosfer.

2.2.1. Troposfer

Dalam Kamus Besar Bahasa Indonesia (KBBI), troposfer didefinisikan sebagai bagian paling bawah atmosfer bumi yang tingginya dari permukaan bumi berkisar 9-17 km, berada di atas khatulistiwa lebih tinggi dari permukaan di atas daerah kutub.

2.2.2. Stratosfer

Dalam Kamus Besar Bahasa Indonesia (KBBI), stratosfer didefinisikan sebagai lapisan udara di antara 12-50 km di atas permukaan bumi, stratosfer terletak di atas troposfer. Beberapa ciri khas lapisan ini adalah sebagai berikut:

a. Lapisan ini adalah lapisan kedua dari bawah setelah troposfer.
b. Terdiri dari 3 wilayah yaitu :
   Stratosfer bawah : Ketinggian 12-20 km
   Stratosfer tengah : Ketinggian 20-35 km
   Stratosfer atas : Ketinggian 35-50 km
c. Lapisan ini tidak mengalami turbulensi maupun sirkulasi.
d. Stratosfer merupakan lapisan atmosfer utama yang mengandung gas ozon

2.2.3. Mesosfer

Lapisan atmosfer ketiga dari bawah ini memiliki beberapa ciri khas sebagai berikut:

a. Ketingian 50-80 km
b. Perubahan suhu terhadap ketinggian adalah *lapse rate.*
c. Suhu udara sekitar -5' C pada dasar lapisan hingga -95'C pada puncaknya.
d. Tidak mengalami turbulensi atau sirkulasi udara.
e. Merupakan daerah penguraian O_2 menjadi atom O.

Batas atasnya adalah lapisan mesopause dengan perubahan suhu terhadap ketinggian mulai bersifat isothermal (Handoko 1994).

2.2.4. Termosfer

Dalam Kamus Besar Bahasa Indonesia (KBBI), termosfer didefinisikan sebagai bagian atmosfer bumi yang bermula dari kira-kira 50 mil di atas permukaan bumi sampai angkasa luar dan ditandai dengan suhu udara semakin tinggi terus-menerus

2.2.5. Ionosfer

Dalam Kamus Besar Bahasa Indonesia (KBBI), Ionosfer didefinisikan sebagai lapisan atmosfer bumi, pada ketinggian 100 km di atas lapisan stratosfer, mengandung ion dan elektron bebas yg dihasilkan oleh radiasi matahari.

2.2.6. Eksosfer

Dalam Kamus Besar Bahasa Indonesia (KBBI), eksosfer didefinisikan sebagai daerah di luar atmosfer yang ketinggiannya kurang lebih 500 km, benda-benda yang sangat ringan di ruang ini akan terlempar ke luar angkasa.

2.3. *Radiosonde*

Instrumentasi *radiosonde* dibawa oleh balon udara ke lapisan troposfer atau diatasnya, dari pengukuran profil vertikal diperoleh data suhu dan

kelembaban relatif. Selain itu, perubahan posisi selama peluncuran juga digunakan untuk mengukur kecepatan dan arah angin. Hasil pengukuran dikirimkan ke stasiun bumi melalui pemancar radio. *Radiosonde* dikembangkan dari alat perekam mekanik yang dibawa oleh balon, dengan pengukuran meteorologi melalui *radiosonde* dilakukan sejak tahun 1930-an. Sejak saat itu, banyak perangkat *radiosonde* diproduksi antar negara. Beberapa desain sendiri diproduksi oleh perorangan untuk pelayanan meterologi nasional, tetapi ada beberapa produsen komersial yang memasok desain *radiosonde* mereka sendiri yang berkuantitas guna memasok kebutuhan *radiosonde* untuk diluncurkan sehari-hari. (Harrison,2015).

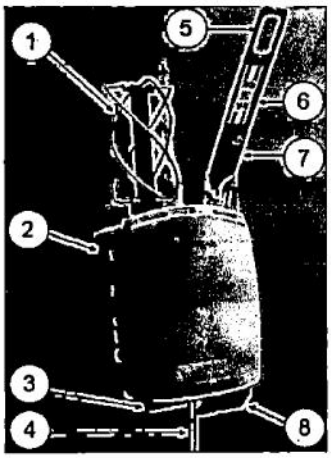
2.3.1. Sistem *Radiosonde*

Sistem *radiosonde* terdiri dari sensor tekanan, suhu, kelembaban, dan GPS yang digunakn untuk mengambil data profil atmosfer serta dilengkapi dengan rangkaian pemancar. Telemetri radio digunakan untuk mengirim data atmosfer. Teknologi *radiosonde* sebelumnya memberikan dasar yang baik yang dapat digunakan untuk memahami prinsip-prinsip operasinya karena mereka mengguanakan teknik sensor sederhana dan metode pengiriman informasi kembali secara langsung oleh radio.

Salah satu perlengkapan utama untuk *radiosonde* adalah catu daya baterai, yang berfungsi mempertahankan pengoperasian perangkat ketika perangkat itu berada dalam kondisi lebih dingin dan lebih jauh dari stasiun penerima. *Radiosonde* awal menggunakan versi ringan dari baterai basah (timbal-asam), tetapi saat ini baterai tersebut jarang digunakan sebagai catu daya dan menggantinya dengan sel primer seperti baterai *alkaline* atau bahkan baterai *lithium*. Pengurangan kapasitas baterai dengan suhu berarti bahwa daya yang dikonsumsi oleh elektronik *radiosonde* harus dipertimbangan dalam menentukan massa baterai yang digunakan. Hal ini memerlukan kompromi antara kekuatan pemancar, dan kapasitas baterai. Sistem radio yang efisien, dengan penerima sensitif dan *gain* tinggi atau antena pelacak juga dapat mengurangi daya pemancar yang dibutuhkan untuk mendapatkan berbagai radio substansial. (Harrison,2015).

2.3.2. *Vaisala Radiosonde RS92-SGP*

Vaisala Radiosonde RS92-SGP menawarkan ketersediaan data, akurasi kelembaban, tekanan, suhu, dan pengukuran angin yang sangat baik. RS92-SGP memiliki sensor tekanan silikon, dua buah sensor kelembaban, dan sensor suhu yang mampu merespon perubahan suhu dengan cepat. RS92-SGP dapat diberi tambahan sensor jika diperlukan. Perinsip kerja RS92-SGP sesuai dengan standar Eropa ETSI (*European Telecommunications Standards Institute*) untuk *radiosonde* digital yang beroperasi di pita 400 MHz. Bagian-bagian Vaisala *Radiosonde* RS92-SGP dapat diterangkan melalui gambar berikut :



1 = GPS antenna
2 = Battery case
3 = Additional sensor interface connector
4 = Antenna
5 = Temperature sensor
6 = Humidity sensors
7 = Sensor boom
8 = GC25 interface

Gambar 2.1 Vaisala *Radiosonde* RS92-SGP (Vaisala Oyj, 2015)

2.3.3. Muatan *Rev-Gaardan*

Rev-Gaardan merupakan salah satu *radiosonde* rakitan yang diikut sertakan pada KOMBAT (Kompetisi Muatan Balon Atmosfer) 2015. Muatan Rev-Gaardan terdiri dari sensor kelembaban, suhu, tekanan, GPS, kamera, dan radio telemetri sebagai media komunikasi dengan *Ground Segment* (GS).

2.4. Komunikasi Data

2.4.1. Pengertian Komunikasi Data

Komunikasi data adalah komunikasi dimana sumbernya adalah data. Data ini merupakan semua informasi yang berbentuk digital (bit 0 dan 1). Transmisi

data berarti pengiriman data antara sebuah komputer dengan terminal atau piranti terminal data (*Data terminal Equipment*). Transmisi suara dapat saja dijadikan transmisi data jika informasi suara tersebut dirubah (dikodekan) menjadi bentuk digital

2.4.2. Pengiriman Seri dan Paralel

Pada Transmisi data, karakter-karakter di sajikan dalam bentuk data yang terdiri dari sederetan angka biner, atau bit (*binary digit*). Setiap bit hanya bernilai biner 1 atau biner 0. Pemindahan, penyimpanan, dan pengolahandata di dalam komputer, atau mikroprosesor, dapat dikerjakan berdasar atas operasi 8-bit, 16-bit, atau 32-bit, tergantung jenis komputer yang digunakan. Setiap 8 bit disebut satu byte. Data dapat dikirimkan ke terminal atau modem menggunakan cara pengiriman seri atau paralel. Pada cara pengiriman paralel, bit-bit yang membentuk karakter dikirimkan secara serempak melewati sejumlah penghantar yang terpisah. Pada saat komputer mempunyai data untuk dikirimkan, jalur data-tersedia DAV (*Data Valid*) diatur tinggi. Pada saat terminal siap menerima data, jalur data diterima DAC (*Data Acepted*) juga akan diatur tinggi. Prosedur *handshaking* ini selalu terjadi setiap kali ada karakter yang dikirim komputer. *Handshaking* ini diperlukan untuk mengakomodasi ketepatan waktu pengiriman data antara komputer dan terminal, atau periferal. Beberapa bentuk *handshaking* secara umum diperlukan karena komputer dan terminal mungkin beroperasi pada kecepatan yang berbeda. Jalur *handshake* biasanya di tambahkan untuk mengendalikan waktu yang tepat untuk pengiriman data. Penghantar yang diperlukan untuk antarmuka paralel disebut lebar bus (*bus width*) adalah 10 penghantar. Setiap penghantar mempunyai fungsi khusus, beberapa diantaranya untuk membawa data, sementara yang lain membawa informasi kendali dan sinkronisasi. Karena dalam sistem pengiriman paralel diperlukan sejumlah penghantar untuk mengirimkan data, sistem pengiriman paralel hanya ekonomis untuk jarak pendek. Masalah yang timbul pada pengiriman paralel yaitu pada *skew*. *Skew* adalah efek yang terjadi pada pengiriman sejumlah bit secara serempak dan tiba pada tempat yang dituju dalam waktu yang tidak bersama-

sama. Efek ini semakin berpengaruh dengan semakin panjangnya kabel yang digunakan, hal ini akan menimbulkan kesalahan pada data yang diterima. Pengiriman paralel biasanya digunakan untuk menghubungkan komputer dengan pencetak cepatan tinggi atau dengan *disk drive* yang berkecepatan tinggi dan panjang kabel relatif pendek.

Pengiriman seri biasanya digunakan untuk sambungan dengan jarak relatif lebih jauh. Data paralel internal dimasukkan ke pengubah paralel ke seri. Pengubah paralel ke seri biasanya dengan IC (*Integrated Circuit*) juga melakukan sejumlah fungsi yang lain dan dikenal sebagai UART (*Universal Asynchronous Recheiver-Transmitter*), ACIA (*Ashynchronous Communication Interface Adapter*), PIA (*Pheripheral Interface Adapter*), dan Iain-lain. Kanal seri mengirimkan setiap karekter per elemen sehingga hanya diperlukan dua penghantar, yaitu kirim data (TXD), dan terima data (RXD). Masing-masing elemen isyarat ekivalen dengan satu bit, dua atau tiga bit (disebut dibit atau tribit), atau kurang dari satu bit (penyandian Manchester), karena bit-bit dikirimkan secara berurutan dan tidak serempak, kecepatan pemindahan data lebih rendah dibanding pengiriman secara paralel. Pengiriman akan dimulai dari LSB (*Least Significant Bit*), dan diakhiri dengan MSB (*most significant bit*). Setiap karakter yang dikirimkan, disajikan dengan suatu urutan bit tertentu sesuai dengan sandi yang digunakan. Penerima harus mencacah isyarat data yang sama, pada waktu yang tepat sebelum membentuk kembali karakter yang diterima.

Pengiriman seri menimbulkan tiga masalah penyesuaian: penyesuaian bit, penyesuaian karakter, dan penyesuaian blok. Agar diterima dengan benar, selang waktu yang digunakan oleh pengirim dan penerima harus sama satu terhadap yang lain. Untuk itu, pengirim dan penerima harus menambahkan detak. Istilah detak (*clock*) digunakan untuk menunjuk sembarang pulsa sumber pewaktuan (*timing pulse*). Detak penerima harus menunjukkan waktu yang tepat kapan isyarat harus dicacah oleh penerima untuk menentukan status logika dari setiap bit yang diterima. Supaya data dapat diterima dengan benar, detak penerima harus sesuai dengan detak pengirim. Jika penerima telah menerima bit sinkronisasi, maka seharusnya segera menerima karakter sinkronisasi. Sehingga, penerima harus

mampu membedakan kelompok-kelompok karakter yang tepat. Dengan kata lain, penerima harus mampu menentukan bahwa suatu bit adalah bit awal (LSB) dari suatu karakter. Selain itu, penerima juga harus dapat mengenali awal dan akhir setiap blok data. Penyesuaian yang diperlukan dapat diperoleh secara sinkron maupun tak sinkron.

2.4.3. Pengiriman Sinkron dan Tak Sinkron

Kedua jenis pengiriman data, sinkron dan tak sinkron, banyak digunakan dalam terminal-terminal. Pemilihan antara pengiriman sinkron dan tak sinkron harus berdasarkan pertimbangan laju tanggapan, biaya terminal dan kanal telepon. Umumnya, pengiriman tak sinkron tidak mahal. Setiap *byte* yang diterima dibedakan dengan bit awal dan bit akhir, sehingga penyesuaian dapat diperoleh dengan mudah. Karena detak penerima selalu dimulai kembali setelah satu karakter diterima dan hanya perlu pada keadaan sinkron untuk selang waktu 8 bit, maka penyesuaian bit juga bukan merupakan persoalan besar. Pengiriman tak sinkron hanya cocok untuk laju yang rendah. Karena : (a) bit awal dan akhir mengurangi efisiensi pengiriman bit menjadi 80%, dan (b) detak yang beroperasi bebas hanya memenuhi syarat pada laju rendah.

Pengiriman sinkron lebih mahal dibanding pengiriman tak sinkron, tetapi dapat bekerja pada laju yang lebih tinggi. Karena data biasanya dikirimkan tanpa pembatas, diperlukan adanya *buffering* baik pada pengirim maupun penerima. Laju pengiriman dapat diubah dengan mengubah detak pengiriman dan kecepatan data pada waktu yang sama. Kerugian pengiriman biasanya berkisar sampai 5%.

2.4.4. *Full Duplex dan Half Dupex*

Hampir sebagian besar sistem komunikasi beroperasi dengan cara *half-duplex* atau *full-duplex*. Sistem komunikasi *half-duplex* dapat mengirimkan data secara bolak-balik (dua arah), tetapi pada satusaat hanya dapat mengirimkan data pada satu arah saja. Proses untuk mengubah arah pengiriman memerlukan tambahan perangkat lunak, dan memerlukan waktu yang disebut *turnaround time*.

Dalam beberapa hal, *turnaround time* berkisar sampai beberapa mili detik, apabila sering terjadi akan menurunkan unjuk kerja rangkaian.

Rangkaian *full-duplex* adalah rangkaian yang dapat mengirimkan data dalam dua arah pada waktu yang sama. Dalam beberapahal, dua kanal yang terpisah digunakan untuk pengiriman pada masing-masing arah. Seringkali, komunikasi *full-duplex* digunakan untuk mengirimkan data meskipun sesungguhnya tidak perlu pengiriman data secara serempak pada kedua arah tersebut. Ini dilakukan untuk memperkecil *turnaround tune* yang berakibat menurunnya waktu tanggapan dari komputer yang digunakan. Jaringan-jaringan komputer yang menggunakan komputer mini, atau mikro, juga sering menggunakan operasi *full-duplex* agar biaya tetap rendah. (Sony, 2010).

2.5. Tipe Data

Sebuah tipe data (atau hanya tipe) didefinisikan dalam tiga komponen :

a) Satu paket nilai (atau objek data)
b) Satu paket operasi yang dapat diterapkan untuk semua nilai dalam paket
c) Sebuah representasi data, yang menentukan bagaimana nilai-nilai yang disimpan

Sebuah bahasa pemrograman menyediakan beberapa tipe data yang telah ditetapkan yang dikenal sebagai tipe data *built-in*. Sebuah bahasa pemrograman juga dapat membiarkan programmer menentukan jenis data mereka sendiri, yang dikenal sebagai tipe data *user-defined.*

Sebuah tipe data yang terdiri dari sebuah atom, nilai terpisahkan, dan yang didefinisikan tanpa bantuan dari setiap jenis data lainnya, yang dikenal sebagai tipe data primitif. Tipe data *user-defined* didefinisikan dalam hal tipe data primitif dan tipe yang ditetapkan pengguna lainnya data. Biasanya, bahasa pemrograman tidak membiarkan *programmer* mengembangkan atau mendefinisikan kembali tipe data primitif. (Koleman,1979).

2.5.1. *Array*

Array adalah struktur data yang panjangnya sudah ditetapkan yang digunakan untuk menyimpan lebih dari satu nilai dari tipe data yang sama. Sebuah *array* terdiri atas elemen-elemen dan dimensi. Elemen-elemen adalah data yang menyusun *array.* Sedangkan dimensi adalah ukuran *array.* Sebuah *array* dapat memiliki satu atau lebih dimensi, dengan jumlah elemen maksimum sebanyak 2147483647 buah per dimensi. Elemen *array* dapat bertipe data numerik, *Boolean, path, string, waveform* atau *cluster*, asalkan dalam satu *array*, semua elemen tersebut bertipe data yang sama. Setiap elemen *array* memiliki nomor indeks, yang menunjukkan lokasi elemen tersebut dari yang lain.untuk *array* 1D, maka *array* tersebut hanya memiliki satu buah indeks. Sedangkan untuk *array* 2D (sering disebut sebagai Matriks), memiiki 2 jenis indeks, yaitu indeks baris dan indeks kolom. Nomor indeks tersebut selalu dimulai dari 0.

Array biasa digunakan untuk menyimpan data dalam jumlah yang besar, yang diperoleh dari suatu struktur pengulangan. *Array* sangat berguna untuk menyederhanakan program dalam melakukan operasi atau komputasi yang berulang pada sekumpulan data yang sejenis.

2.5.2. *String*

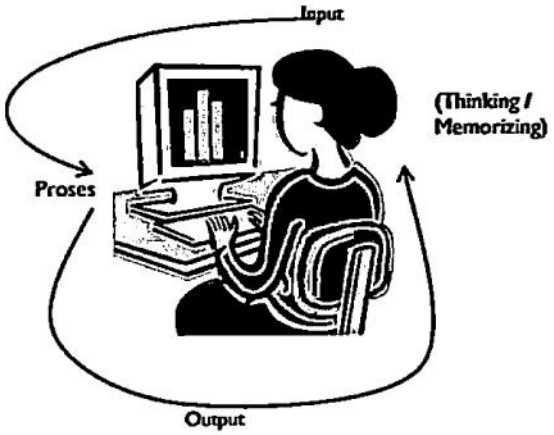
String merupakan susunan karakter-karakter ASCII (*American Standard Code for Internasional Interchange*), baik yang dapat ditampilkan (*displayable*) maupun yang tidak dapat ditampilkan (*non-displayable*). Beberapa manfaat *string* adalah sebagai berikut :

a. Memberikan informasi berbentuk teks yang jelas.
b. Merupakan kode standar untuk komunikasi data secara serial.
c. Sebagai format data untuk pengolahan *file*, seperti pembacaan dan penyimpanan *file*. (Artanto, 2012).

2.6. Interaksi Manusia dan Komputer (IMK)

Ilmu interaksi manusia dan komputer merupakan ilmu yang mempelajari tentang cara mendesain, mengevaluasi, dan mengimplementasikan sistem komputer yang interaktif sehingga pengguna dapat menggunakannya dengan mudah.

Pada prinsipnya, kerja komputer adalah menerima masukan (*input*), memproses (*process*), dan memberikan keluaran (*output*). Pengguna memberikan memasukkan perintah atau data kepada komputer. Masukan ini dilakukan dengan berbagai macam piranti masukan (*input device*). Masukan ini akan direspon oleh komputer. Komputer melakukan pengolahan dan melaksanakan perintah sesuai dengan perintah pengguna kemudian menyajikan kembali hasil pemrosesan tersebut kepada pengguna. Hal ini terjadi berulang kali sehingga terjadi interaksi. Interaksi tersebut merupakan komunikasi dua arah antara manusia atau pengguna dengan komputer. Perhatikan gambar berikut ini.

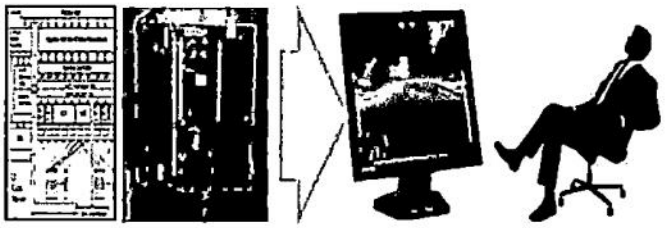


Gambar 2.2 Interaksi Manusia dengan Komputer

Perintah dan masukan pengguna merupakan bahasa atau simbol yang dapat dimengerti oleh manusia. Setelah masukan pengguna diterima, komputer melakukan proses dengan kerumitan yang sangat tinggi. Semua data pada komputer dinyatakan dalam notasi biner. Notasi biner ini merupakan represntasi dari suatu *threshold* tegangan tertentu sedemikian sehingga nilai 1 untuk tegangan

tertentu dan nilai O untuk tegangan yang lain. Tidak hanya representasi data saja yang rumit, pengolahan data komputer juga merupakan proses yang tidak kalah rumitnya. Pada prosesor, dilakukan operasi yang mengeksekusi perintah-perintah tingkat rendah yang dalam satu detik dapat melaksanakan jutaan instruksi. Satu instruksi tingkat rendah berlangsung sangat singkat, tergantung dari kecepatan prosesor yang digunakan.

Berbagai kerumitan yang ada pada pemrosesan data pada komputer tidak akan diperlihatkan kepada pennggguna. Pengguna hanya diberikan antarmuka tempat data dimasukkan dan hasil pengolahan dikeluarkan. Antarmuka yang paling umum adalah layar monitor.



Gambar 2.3 Antarmuka Komputer

Melalui antarmuka yang baik, interaksi akan optimal. Waktu jeda untuk memasukkan *input* dan waktu untuk memahami *output* dapat ditekan seminimal mungkin.

Disadari atau tidak, pada dasarnya penggunaan komputer oleh manusia merupakan sebuah interaksi atau dialog. Interkasi terjadi antara manusia dengan layar monitor. Tidak hanya mata yang terlibat dengan melihat layar monitor. Berbagai macam aksi dilakukan tangan dengan menekan tombol pada *keyboard* berupa tombol angka dan huruf, atau dengan melakukan satu sentuhan kecil pada *mouse*. Hasil yang terlihat di layar monitor terus dipantau dan menentukan aksi selanjutnya. Contohnya, ketika seseorang mengetikkan kata "selamat pagi" namun ternyata ia salah mengetik "semalat pagi" pengguna segera melakukan aksi untuk memperbaiki. Aksi yang dilakukan mungkin berbeda untuk satu *user* dengan *user* yang lain. Ada yang menggunakan *backspace*, ada menggunakan *key* panah kiri,

atau pintasan dengan menggunakan CTRL untuk berpindah ke kata sebelumnya baru melakukan pergeseran.

Berbagai macam aksi ini ditentukan oleh beberapa faktor. Misalnya dari aspek teknologi aplikasi, mungkin telah tersedia perbaikan otomatis sehinga pengguna tidak perlu memperbaikinya, atau terdapat pintasan untuk berpindah antar kata, sehingga pengguna dapat berpindah dari satu kata ke kata yang lain dengan cepat. Selain itu, pengetahuan pengguna terhadap aplikasi yang ia gunakan juga menentukan langkah-langkah yang diambilnya. Boleh jadi aplikais telah menyediakan berbagai macam fitur yang diperlukan, namun karena ketidak tahuan pengguna, maka fitur tersebut tidak digunakan. Pengguna biasanya tidak menyadari adanya interaksi dengan komputer. Pengguna baru menyadari proses interaksi ini apabila terdapat masalah yang mengganggu interaksinya, misalnya proses atau respon yang lambat, layar yang kurang menarik, layar yang sulit dibaca, teks peringatan yang sulit dipahami dan sebagainya. Pada interaksi antara manusia dengan komputer terdapat beberapa model yang umum digunakan, antara lain:

a) *Command line interface*
b) *Menu*
c) *Natural language*
d) *Question/answer and query dialogue*
e) *Form-fills and spreadsheets*
f) *Windows, Icons, Menus and Pointers*

2.6.1. Bidang yang Berkaitan dengan IMK

Untuk mendapatkan interaksi yang baik antara manusia dengan komputer, pada perancangan desain antarmuka diperlukan berbagai macam pertimbangan dari berbagai sudu disiplin ilmu. Bidang studi yang berperan dalam ilmu interaksi komputer adalah sebagai berikut :

a) Teknik elektronika & ilmu komputer : Bidang ini merupakan bidang dasar dalam pembahasan interaksi manusia dengan komputer karena komputer sendiri merupakan subjek dalam interaksi tersebut. Teknologi yang berkembang pada komputer akan sangat mempengaruhi pola interaksi yang terjadi. Teknologi tersebut tidak hanya teknologi perangkat keras, namun juga perangkat lunak. Berbagai macam piranti masukan maupun keluaran semakin memanjakan pengguna. Perangkat lunak yang memungkinkan penyajian data dalam berbgaia format seperti teks, video, maupun suara juga semakin maju. Keduanya merupakan hal fundamental dalam perancangan sistem interaksi antara manusia dengan komputer

b) Psikologi : Bidang ini memperhatikan sifat, kebiasaan, persepsi, pengolahan kognitif, dan ketrampilan motorik pengguna

c) Perancangan grafis dan tipografi : Bidang ini memperhatikan bagaimana representasi grafis (gambar) yang mewakili berbagai macam kalimat. Gambar digunakan sebagai sarana dialog yang cukup efektif saat terjadi interkasi antara manusia dengan komputer. Sebuah gambar atau ikon dapat secara cepat memberikan informasi atau makna bagi *user* tanpa harus banyak berkata-kata. Respon kecepatan pemahaman terhadap gambar juga lebih baik dari pada respon terhadap teks yang panjang.

d) Ergonomik : Bidang ini berkaitan dengan aspek fisik untuk mendapatkan lingkungan kerja yang nyaman. Aspek fisik ini misalnya berkaitan dengan meja, kursi, lampu, *keyboard, mouse*, dan sebagainya.

e) Antropologi : Bidang ini merupakan cabang ilmu pengetahuan yang mempelajari tentang manusia, dan memberi suatu pandangan tentang cara kerja berkelompok yang masing–asing anggotanya dapat memberikan konstribusi sesuai dengan bidangnya. Kehadiaran teknologi informasi (dengan sarananya adalah komputer) sedikit banyak telah mempengaruhi cara kerja dan pola hubingan sekelompok orang.

f) Linguistik : Bidang ini merupakan cabang ilmu pengetahuan yang mempelajari tentang bahasa. Bahasa sebagai sarana komunikasi mendasar diperlukan dalam

pembuatan desain interaksi antara manusia dengan komputer. Melalui bahasa, berbagai macam simbol dan suara diintepretasikan sebagai perintah yang akan dikerjakan oleh komputer. Contoh bahasa diantaranya bahasa grafis, bahasa alami, bahasa menu, dan bahasa perintah. Bahasa yang kita gunakan sehari-hari merupakan contoh bahasa alami.

g) Sosiologi : Bidang ini mempelajari tentang pengaruh sistem manusia-komputer dalam struktur sosial. Berbagai macam dampak sosial dapat timbul oleh adannya perkembangan teknologi komputer yang semakin canggih baik dari sisi *hardware* maupun *software*.

2.6.2. Kebergunaan (*Usability*) Antarmuka

Dalam mendesain antarmuka komputer, perlu diperhatikan faktor tingkat kebergunaan/ usability. *Usability* memegang peranan cukup besar agar sistem secara baik dapat diterima *user* dan seluruh pihak yang terkait dengan sistem.

Menurut Nielsen ada lima hal yang menentukan *usability* yaitu:

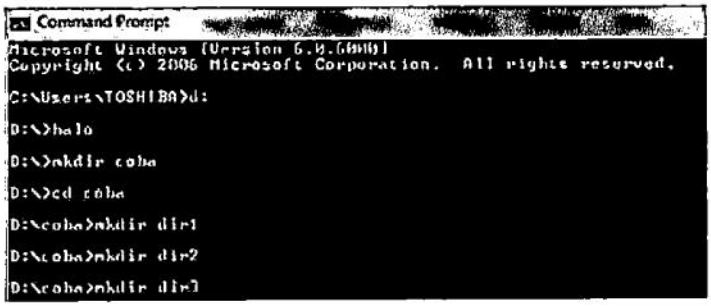
a) *Learnability:* Pengguna dapat segera memulai pekerjaanya semenjak dimulainya penggunaan sistem.
b) *Efficiency:* Pengguna dapat meningkatkan produktifitasnya setelah pertama kali belajar.
c) *Memorability*: Pengguna dapat menggunakan sistem kembali dengan cepat setelah lama tidak menggunakan aplikasi tersebut tanpa perlu belajar dari awal kembali
d) *Errors*: Pengguna harus mampu diarahkan untuk sekecil mungkin berbuat kesalahan. Apabila pengguna melakukan kesalahan harus ada langkah penanganan yang dapat memulihkan kesalahan tersebut dengan segera.
e) *Satisfaction*: Pengguna harus merasa nayaman dengan sistem aplikasi yang digunakannya.

2.6.3. Konsep Interaksi Komputer

Interkasi antara manusia dengan komputer dapat dipandang sebagai dialog antara keduanya. Interaksi ini melibatkan manusia sebagai pengguna, prosesor komputer, dan antarmuka diantara keduanya yang menjadi sarana *input* dan *output*.

Setiap aplikasi memiliki pola interaksi sendiri-sendiri sesuai dengan kebutuhan dan teknologi yang ada. Berikut ini adalah beberapa bentuk pola interaksi manusia dan komputer :

a) Antarmuka dengan baris perintah tunggal : Interaksi ini merupakan interaksi yang paling primitif. Perintah yang dilakukan dilakukan melalui *keyboard* dengan sebuah *key* maupun beberapa karakter yang membentuk kata kunci tertentu.
b) Antarmuka dengan baris perintah terstuktur : Pola interaksi ini merupakan kumpulan dari banyak perintah tunggal menjadi satu rangkaian. Pola ini sering dijumpai pada file .BAT pada sistem operasi windows dan .sh pada Linux (Catatan: pada linux, ekstension ini bukanlah hal yang wajib dan dapat diubah sesuai kebutuhan). Dalam interaksi tingkat tinggi, pola interaksi ini relatif jarang ditemui.



Gambar 2.4 Antarmuka baris perintah terstruktur

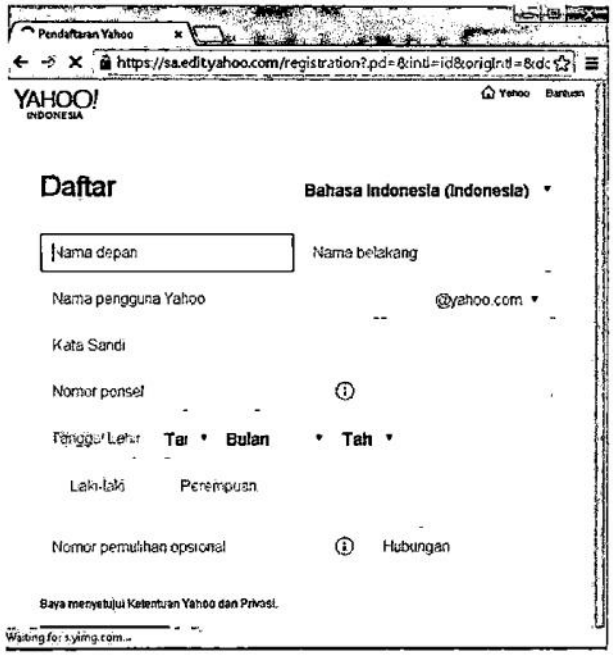
c) Menu dan WIMP (*Window Icon Menu Pointer*) : Menu merupakan sekumpulan pilihan yang diberikan kepada pengguna. WIMP atau *Window*

Icon Menu Pointer merupakan model interaksi standar yang berkembang pesat saat ini khususnya pada PC (*Personal Computer*).

d) Bahasa sehari-hari/alami : Pola interaksi dengan menggunakan bahasa alami lebih luas dari perintah baris (*command line*). Interaksi dengan bahasa alami dapat dikategorikan menjadi dua macam, yaitu interaksi secara tidak langsung dan interaksi secara langsung. Contoh untuk interaksi tidak langsung adalah bahasa pemrograman. Dengan bahasa pemrograman manusia memerintahkan komputer melalui *source code* untuk melakukan suatu tugas tertentu. Walaupun bahasa pemrogaman ini tidak langsung di mengerti oleh komputer, namun antarmuka terdepan antara komputer dengan pengguna adalah bahasa alami. Contoh bahasa pemrograman tidak langsung adalah dalam bahasa Pascal, LISP, dan Prolog. Sementara contoh pola interaksi langsung adalah *Speech Recognition.* Dengan teknologi *Speech Recognition*, pengguna dapat memerintahkan secara lisan tanpa harus mengetik perintah ke komputer.

e) Dialog dengan tanya jawab terstruktur/*Query :* Interaksi jenis ini sering digunakan pada sistem informasi, dimana pengguna mengajukan serangkaian pertanyaan atau perintah yang akan dikerjakan oleh komputer. *Query* merupakan pengembagan lebih lanjut dari interaksi dengan bahasa alami yang difokuskan untuk tujuan tertentu. Perintah *query* biasanya berkaitan dengan dengan aplikasi *data base.*

f) Formulir isian dan kertas kerja : *Form* merupakan tabulasi isian yang yang merupakan bentuk elektronik dari borang. *Form* yang baik harus mendukung kemudahan dan kenyamanan dengan tetap mejaga kualitas data. *Form Menu* merupakan sekumpulan pilihan yang diberikan kepada pengguna. Sekumpulan pilihan ini dapat disejadikan dengan beberapa cara misalnya pada registrasi yahoo, calon pengguna diminta mengisi *form* seperti gambar berikut:



Gambar 2.5 Formulir isian

2.6.4. Prinsip dalam Desain Antarmuka

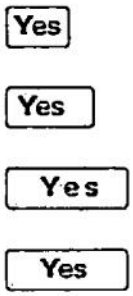
Untuk mendapatkan antarmuka yang baik, perlu diperhatikan prinsip prinsip dasarnya pada saat mendesain. Para ahli mencoba memformulasikan berbagai macam prinsip pembangunan antarmuka yang baik. Prinsip desain merupakan serangkaian panduan yang akan membantu desainer mengambil keputusan perancangan selama proses tersebut berjalan. Prinsip ini juga merupakan petunjuk umum yang dihasilkan dari pengalaman para pakar, yang boleh jadi bebeda antara seorang pakar dengan pakar yang lain. Namun diantara berbagai pendapat tersebut terdapat prinsip umum yang berlaku untuk semua. Beberapa prinsip yang akan dibahas adalah sebagai berikut.

a) Ben Shneiderman's dengan prinsipnya yang dikenal sebagai "*Eight Golden Rules of Dialog Design*",
b) Deborah J. Mayhew's melalui "*General Principles of User Interface Design*",
c) IBM (*International Business Machines*) melalui "*Design Principles for Tomorrow*"

2.6.5. Delapan Prinsip Emas Ben Shneiderman

Ben Shneiderman mengusulkan delapan prinsip yang didasarkan pada pengalamannya secara heuristik namun dapat diterapkan pada berbagai macam sistem yang interaktif setelah melalui proses yang panjang dengan pemilahan, perluasan, dan penginterpretasian. Kedelapan prinsip tersebut adalah:

a) Upayakan untuk tetap konsisten : Konsisten pada serangkaian aksi yang mirip merupakan hal yang penting. Konsistensi ini berkaitan dengan tampilan, menu, bantuan, perintah, pesan-pesan yang disampaikan, dan istilah yang digunakan di layar.



Gambar 2.6 Tombol "Yes"

Pada gambar tersebut terdapat berbagai macam tombol "Yes" dengan berbagai macam variasinya. Namun aplikasi yang konsisten, sekali memilih bentuk tombol "Yes" maka bentuk tersebut harus senantiasa digunakan pada setiap kali dibutuhkan.

b) Gunakan pintasan pada bagian yang sering digunakan : Dengan pintasan, banyaknya interaksi dapat ditekan seminimal mungkin. Semakin sedikit interaksi semakin mudah penggunaan, semakin kecil pula kesalahan. Berbagai macam singkatan, tombol, perintah cepat dengan menggunakan *key* tertentu (*Function, shit, control, tab*, dsb.) sangat berguna bagi pengguna yang mahir dengan aplikasi yang digunakannya. Contoh pintasan dengan *key*:

- [Alt] + [tab]
- [Windows]+[e]

- [Windows]+[d]
- [Windows]+[r]
- [alt]+[f4].
- [Control]+[Alt]+[Del]

c) Sediakan *feedback* yang informatif : Untuk operasi yang umum berikan *feedback* yang umum sedangkan untuk operasi yang khusus apalagi berbahaya berikan *feedback* dengan penekanan. Desain yang dibuat harus tetap menginformasikan kepada pengguna mengenai aksi atau interpretasi, perubahan status, adanya kesalahan, atau adanya ekspesi yang relevan dan menarik secara melalui bahasa yang jelas, singkat, tidak ambigu, dan familiar bagi pengguna.

d) Dialog memiliki lingkup tertentu : Serangkaian akasi harus diatur kedalam kelompok sedemikian sehingga terdapat bagian awal, pertengahan, dan penutup. *Feedback* yang informatif pada sekelompok aksi tersebut memberikan kenyamanan bagi pengguna dan selanjutnya menjadikan petunjuk untuk serangkaian langkah berikutnya.

e) Sediakan penanganan kesalahan yang sederhana : Dalam mendesain sistem, harus dibuat sedapat mungkin pengguna tidak melakukan kesalahan yang fatal. Apabila terjadi kesalahan yang serius, sistem harus dapat mendeteksi dan melakukan penanganan kesalahan secara sederhana dan komprehensif.

f) Perbolehkan pengguna melakukan aksi mundur atau pembatalan : Fitur ini menghilangkan kekhawatiran pengguna karena pengguna melakukan kesalahan, maka aksi tersebut dapat di batalkan. Pembatalan ini mungkin dapat berupa pembatalan sebuah aksi, sekelompok data entri, ataupun sekelompok aksi.

g) Berikan kontrol internal : Pengguna cenderung merasa tahu dan mereka mampu mengendalikan sistem. Oleh karena itu desain yang baik harus memposisikan pengguna sebgaia inisiator ketimbang sepagai responder dari sistem.

h) Kurangi aktifitas mengingat : Keterbatasan manusia yang berkaitan dengan memori jangka pendek menuntut desainer sistem membuat antarmuka yang sederhana. Sedapat mungkin pengguna tidak perlu mengingat terlalu banyak.

2.6.6. Prinsip Umum Merancang Antarmuka

Deborah J. Mayhew, memperkenalkan *General Principles Of UI Design*, atau Prinsip umum desain antarmuka. Ada 17 prinsip yang harus dipahami para perancang sistem, terutama untuk mendapatkan hasil maksimal dari tampilan yang dibuat.

a) *User Compatibility : User Compatibility* artinya kesesuaian tampilan dengan tipikal dari pengguna. karena berbeda pengguna bisa jadi kebutuhan tampilannya berbeda. misalnya, jika aplikasi diperuntukkan bagi anak-anak, maka jangan menggunakan istilah atau tampilan orang dewasa.

b) *Product Compatibility :* Istilah ini mengartikan bahwa produk aplikasi yang dihasilkan juga harus sesuai. memiliki tampilan yang sama/serupa. baik untuk pengguna yang awam maupun yang ahli.

c) *Task Compatibility* : *Task Compatibility* berarti fungsional dari task/tugas yang ada harus sesuai dengan tampilannya. misal untuk pilihan *report*, orang akan langsung mengartikan akan ditampilkan laporan. sehingga tampilan yang ada bukanlah tipe data (dari sisi pemrogram).

d) *Work Flow Compatibility* : Aplikasi bisa dalam satu tampilan untuk berbagai pekerjaan. Jika tampilan yang ada hanya untuk satu pekerjaan saja. misal untuk kirim pesan, maka kita harus membuka tampilan tersendiri untuk daftar alamat.

e) *Consistency* : Konsisten. Contohnya, jika anda menggunakan istilah *save* yang berarti simpan, maka gunakan terus istilah tersebut.

f) *Familiarity* : *Icon* disket akan lebih familiar jika digunakan untuk perintah menyimpan.

g) *Simplicity* : Aplikasi harus menyediakan pilihan *default* untuk suatu pekerjaan.

h) *Direct Manipulation* : Manipulasi secara langsung, misalnya untuk mempertebal huruf, cukup dengan ctrl+B.

i) Kontrol : Berikan kontrol penuh pada pengguna, tipikal pengguna biasanya tidak mau terlalu banyak aturan.

j) WYSIWYG : *What You See Is What You Get*, buatlah tampilan mirip seperti kehidupan nyata pengguna dan pastikan fungsionalitas yang ada berjalan sesuai tujuan.

k) *Flexibility Tool* : atau alat yang bisa digunakan pengguna. jangan hanya terpaku pada *keyboard* atau *mouse* saja.

l) *Responsiveness :* Tampilan yang dibuat harus ada responnya. misal, yang sering kita lihat ketika ada tampilan "*please wait...* 68%".

m) *Invisible Technology* : Pengguna tidak penting mengetahui algoritma apa yang digunakan. Contohnya untuk mengurutkan pengguna tidak perlu mengetahui algoritma yang digunakan programmer (*max sort, bubble sort, quick sort*, dst)

n) *Robustness* : Handal, yang berarti dapat mengakomodir kesalahan pengguna.

o) *Protection* : Melindungi pengguna dari kesalahan yang umum dilakukan. misalnya dengan memberikan *fitur back* atau *undo.*

p) *Ease of Learning* : Aplikasi yang dibuat harus mudah dipelajari.

q) *Easy of use :* Aplikasi yang dibuat harus mudah digunakan.

2.6.7. Orientasi Objek Antarmuka Pengguna IBM

IBM memiliki prinsip-prinsip desain, yang diturunkan dari prinsip desain klasik dengan perluasan yang mengkaitkan dengan antrmuka modern berdasarkan pengalamannya pada *object-oriented user interface* (OOUI. Prinsip-prinsip tersebut adalah sebagai berikut.

a) Kesederhanaan: Tanpa mengabaikan kegunaan demi fungsionalitas tertentu, antarmuka yang baik harus tetap menjaga kesederhanaan. Sebuah fungsionalitas yang handal namun sangat sulit untuk digunakan tetap saja merupakan hal yang tidak disukai pengguna. Untuk fungsi-fungsi dasar, harus segera terlihat oleh pengguna sedangkan fungsi lanjutan dapat diberikan menu pilihan tersendiri. Namun harus senantiasa diusahakan bahwa akses terhadap suatu fungsi memerlukan langkah seminimum mungkin.

b) *Support:* pengguna tetap terkendali melalui panduan proaktif. Pengguna senantiasa tetap dalam kendali. Keadaan sistem senantiasa terkendali dan stabil. Apabila interkasi dihentikan, pengguna dapat memeperoleh sistem seperti keadaan saat dihaentikan, misalnya pengguna meninggalkan pekerjaanya sejam atau seharian.

c) *Familiarity:* Bangun pemahaman pengguna. Interaksi yang didasarkan pengalaman pengguna akan suatu sistem akan sangat membantu dalam interaksi pada suatu sistem baru. Pengalaman ini dapat berasal dari pengalaman sehari-harinya maupun pengalaman menggunakan sistem yang lain.

d) *Obviousness:* Buat Objek fungsinya dapat terlihat dan intuitif. Berbagai macam representasi objek dapat diambil dari dunia nyata. Misalnya lamabang disket, mengindikasikan bahwa ikon tersebut digunakan untuk menyimpan data. Contoh lainnya ikon telepon, artinya pengguna dapat melakukan panggilan dengan menggunakan menu tersebut.

e) *Encouragement:* Aksi dapat diperkirakan hasilnya dan dapat dibatalkan. Berbagai aksi pengguna harus menghasilkan hasil sebagaimana yang diinginkan pengguna. Untuk itu desainer harus memahami benar benar pekerjaan pengguna, tujuannya, dan model mentalnya. Pengguna dapat memperkirakan hasil atau akibat dari berbagai macam tindakan yang dilakukannya pada sistem. Apabila hasilnya tidak seperti yang diharapkan, pengguna harus diberi kesempatan untuk membatalkan aksi tersebut, dan keadaan dipulihkan seperti sedia kala.

f) *Satisfaction:* Berikan pencapaian kemajuan setiap aksi yang dilakukan pengguna, berikan informasi yang cukup kepada pengguna mengenai aksi yang telah ia lakukan. Misalnya saat pengguna mengubah jenis huruf, maka jenis huruf yang diubah harus segera di tampilkan agar pengguna dapat segera mengevaluasi apakah huruf yang dipilih cocok atau tidak. Apabila pengguna tidak cocok dengan huruf baru tersebut, maka dapat segera di ambil keputusan untuk mengubah huruf kembali. Contoh lain, apabila pengguna menghapus berkas, maka setelah operasi penghapusan, berkas

tersebut harus hilang dari daftar berkas dengan segera. Contoh penampilan progres adalah *progress bar* saat mengunduh berkas atau menginstall suatu aplikasi.

g) *Accessibility:* Buat semua object dapat di akses setiap saat. Pengguna harus dapat mengakses semua objek setiap saat. Hindari sedapat mungkin adanya '*mode*' dalam penggunaan aplikasi yang menyebabkan perbedaan state pada antarmuka. Misalnya adanya menu yang tidak berjalan karena masalah hak akses, atau pesan "*no longer available*".

h) *Safety:* Pastikan pengguna terbebas dari masalah : Pengguna harus dilindungi dari kemungkinan melakukan kesalahan. Pencegahan kesalahan ini sangat dintentukan oleh desain dari sistem yang dibangun.

i) *Versatility:* Berikan alternatif teknik interaksi : Berikan kesempatan kepada pengguna untuk menentukan teknik interkasi yang paling sesuai dengan keadaanya. Antarmuka yang fleksibel akan sangat membantu pengguna pada berbagai macam tingkatan keahlian, sering tidaknya berinteraksi, atau lingkungan penggunaan. Misalnya seorang pengguna yang sudah mahir dalam operasi *file,* lebih menyukai *command line* dari pada interkasi melalui *window*. Dalam beberapa kasus, operasi *command line* lebih cepat dari pada operasi dengan menggunakan *window*.

j) *Personalization:* Berikan kesempatan pengguna untuk kustomisasi antarmuka yang baik memberikan kepada pengguna melakukan kustomisasi dengan antarmukanya. Tidak ada seorang pengguna pun yang memiliki kesamaan selera dalam semua hal. Melalui kustomisasi akan membuat pengguna merasa nyaman. Melalui personalisasi dapat meningkatkan produktifitas. Misalnya adanya pengaturan nilai awal dapat mempercepat proses *input*.

k) *Affinity:* Sesuaikan objek dengan kehidupan nyata melalui desain visual. Dengan desain visual pada antarmuka, pengguna dapat meningkatkan kualitas interaksi bersama sama dengan prinsip desain yang lain. Desain visual yang dibangun harus mencerminkan model pengguna dan juga fungsi yang diberikan tanpa menimbulkan ambiguitas. (Pratondo, 2009).

2.7. *Data Logger*

Data logger dapat didefinisikan sebagai suatu sistem yang berfungsi untuk mengambil, mengumpulkan dan menyiapkan data, hingga memprosesnya untuk menghasilkan data yang dikehendaki. Pengembangan sistem akuisisi data atau disebut juga sebagai *data logger* ini melibatkan dua sub sistem yaitu sub sistem *hardware* dalam hal ini sensor sebagai pengambil data dari obyek yang diukur dan sub sistem *software* yang merupakan sub sistem untuk mengumpulkan dan memproses data yang kemudian dapat ditampilkan sesuai dengan kebutuhan. Menurut Lilik (2011) Sensor adalah jenis tranduser yang digunakan untuk mengubah ariasi mekanis, magnetis, panas, sinar dan kimia menjadi tegangan dan arus. Sensor sering digunakan untuk pendeteksian pada saat melakukan pengukuran atau pengendalian. *Data logger* berkembang pesat sejalan dengan kemajuan dibidang teknologi digital dan komputer. Pengolahan data pengontrolan yang dilakukan oleh komputer memungkinkan penginterpretasian data dapat dengan mudah dilakukan. Dengan komputer data yang sudah diambil dapat disimpan dan sewaktu–waktu dapat dimanfaatkan kembali. (Pribadi & Ananta, 2011).

2.8. LabVIEW

LabVIEW merupakan perangkat lunak yang khusus digunakan untuk pemrosesan dan visualisasi data dalam bidang akuisisi data, kendali dan instrumentasi, serta otomasi industri. Perangkat lunak ini pertama kali dikembangkan oleh perusahaan National Instruments (NI) pada tahun 1986. LabVIEW merupakan singkatan dari *Laboratory Virtual Instrument Engineering Workbench.* (Artanto, 2012)

Virtual instrument (VI) merupakan elemen pemrograman LabVIEW . Sebuah VI terdiri dari panel depan, diagram blok, dan ikon yang mewakili program. Panel depan digunakan untuk menampilkan kontrol dan indikator bagi pengguna, dan diagram blok berisi kode untuk VI. Ikon yang merupakan

representasi visual dari VI, memiliki konektor untuk masukan dan keluaran program. (Bitter, Mohiudin, & Nawocki, 2007).

Beberapa kelebihan LabVIEW dibandingkan dengan bahasa pemrograman lainnya adalah:

a) Bahasa pemrograman LabVIEW jelas dan mudah dipahami, karena berbentuk grafis, dengan instruksi berbentuk ikon-ikon, yang dihubungkn dengan garis/kawat untuk menunjukkan aliran data, mirip seperti *flowchart.*
b) Pembuatan program mudah, yaitu hanya dengan menarik keluar ikon instruksi yang sudah tersedia di palet (kotak instruksi), dan menghubungkannya dengan kawat ke ikon yang lain. Kawat ini sama seperti variabel pada bahas pemrograman teks. Dengan cara ini, LabVIEW menyederhanakan pemrograman, karena kawat hanya akan terhubung apabila tipe datanya sesuai sehingga menghilangkan kebutuhan manajemen memori dan deklasi tipe data setiap variabel seperti dalam bahasa pemrograman teks. Juga tidak perlu mengingat nama-nama instruksi karena semua ditampilkan pada palet. Jadi, pembaca hanya perlu mencarinya dari kategori yang disediakan, atau dengan menggunakan bantuan tombol *Search* untuk menemukannya.
c) Karena mudah dipahami dan mudah dibuat, maka mempersingkat waktu pembuatan program. Begitu pula untuk perbaikan programnya, karena dibuat dalam bentuk gambar yang bersifat interaktif, maka perbaikan programnya menjadi lebih cepat yang memungkinkan pengembangan program menjadi lebih baik.
d) Dari tahun 1986 hingga sekarang, LabVIEW telah memiliki integrasi dengan ribuan *hardware* dan ratusan *library* yang siap digunakan untuk aplikasi di bidang instrumentasi, pengolahan sinyal, analisa dan visualisasi data serta koneksi ke internet.
e) Telah terbukti di industry sebagai *software* yang andal,powerful, dan fleksibel yang dapat digunakan dengan *library* eksternal dari *sotware* lain, seperti MATLAB, Simulink, SPICE, ADAMS, SolidWorks, dan Lego Mindstorms.

f) Menjembatani dunia pendidikan dengan industri, karena *software* yang digunakan sama, sehingga transisi dan transfer teknologi dari dunia pendidikan ke industri menjadi lebih mudah.

g) LabVIEW didesain sebagai sebuah bahasa pemrograman paralel (*multicore*) yang mampu menangani beberapa instruksi sekaligus dalam waktu bersamaan. Hal ini sangat sulit dilakukan dalm bahasa pemrograman teks, karena biasanya bahasa perograman teks mengeksekusi instruksinya secara berurutan per baris satu. Dengan LabVIEW, pengguna dapat membuat aplikasi eksekusi paralel ini secara mudah dengan menempatkan beberapa struktur *loop* secara terpisah dalam *block digram.*

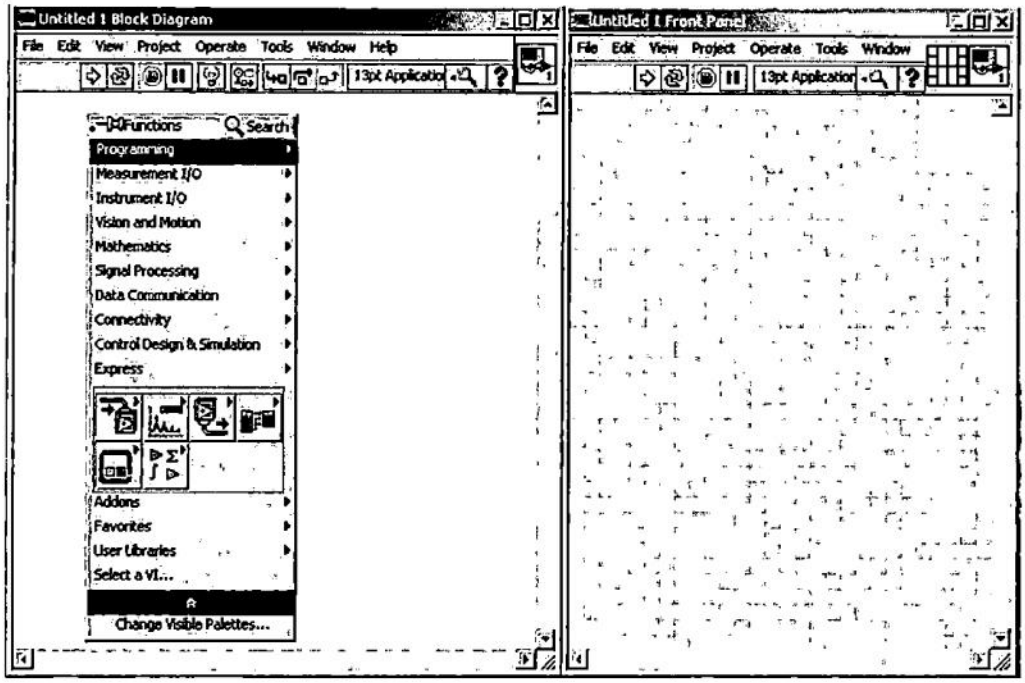
h) Sifat modular LabVIEW memungkinkan pengguna untuk membuat program yang kompleks dan rumit menjadi sederhana, yaitu dengan cara membuat sub program, atau pada LabVIEW disebut sebagai subVI. Beberapa subVI dapat digabungkan menjadi sebuah subVI. Beberapa subVI gabungan tersebut dapat digabungkan lagi menjadi sebuah subVI lain, demikian seterusnya dengan tingkat hierarki yang tidak terbatas.

i) Satu alat untuk berbagai bidang, meliputi otomotif, biomedis, komunikasi data, energi, kontrol, vision, dengan penggunaan mulai dari perencanaan, pengukuran, *prototype*, pengujian, hingga implementasi dan pengembangannya.

j) Komunikasi LabVIEW dan dukungan dari pihak National Instruments yang begitu besar untuk dunia pendidikan

2.9. Lingkungan Pemrograman LabVIEW

Lingkungan pemrograman LabVIEW terdiri atas dua jendela, yaitu jendela *Front Panel* dan jendela *Block Diagram*, seperti gambar berikut ini:



Gambar 2.7 Jendela *Front panel* dan *Block Diagram*

Masing-masing jendela tersebut memiliki *Toolbar* dan *Palet* sendiri-sendiri, berikut ini uraiannya.

2.9.1. *Toolbar Front Panel*

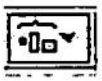
Toolbar Front Panel tampak di bagian atas di bawah menu. Berikut beberapa tombol yang tersedia di *Toolbar Front Panel.*

Gambar 2.8 Tombol-tombo pada *Toolbar Front Panel*

Dengan fungsi masing masing tombol sebagai berikut:

Tabel 2.1 Ikon tombol dan fungsi pada *toolbar Front Panel*

NO	Ikon Tombol	Fungsi
1		Tombol ***Run*** ini untuk menjalankan VI. Saat VI berjalan, tombol ***Run*** tersebut akan berubah menjadi seperti berikut : 

2		Jika suatu ketika tombol ***Run*** terlihat patah seperti gambar disamping, hal ini menandakan ada *error*. Klik tombol tersebut untuk mengetahui error yang terjadi.
3		Klik tombol ***Run Continuosly*** ini untuk menjalankan VI terus-menerus. Setelah berjalan, VI hanya akan berhenti bila tombol ***stop*** atau ***pause*** ditekan.Saat VI berjalan, tombol ini akan berubah menjadi 
4		Tombol ***Abort Execution*** ini untuk menghentikan VI.
5		Tombol ***Pause*** ini untuk menghentikan VI sesaat.
6		Tombol ***Text Setting*** ini untuk mengubah bentuk huruf, ukuran, perataan dan warna pada teks.
7		Tombol ***Align Objects*** ini untuk menata posisi beberapa objek supaya rata, termasuk rata kanan, rata kiri, rata atas, dan rata bawah.
8		Tombol ***Distribute Object*** ini untuk menata posisi antar objek dengan spasi yang sama.
9		Tombol ***Resize Object*** ini untuk mengubah ukuran objek
10		Tombol ***Reorder*** ini unuk menempatkan objek di depan atau di belakang objek lain.

2.9.2. *Toolbar Block Diagram*

Toolbar di *Block Diagram* hampir sama dengan *Toolbar di Front Panel*, hanya sedikit berbeda, yaitu pada *Block Diagram* tidak ada fungsi *Resize Object* dan ada 6 tombol tambahan, yaitu :

Gambar 2.9 Tombol-tombol tambahan pada *Block Diagram*

Tabel 2.2 Ikon tombol dan fungsi pada *toolbar block diagram*

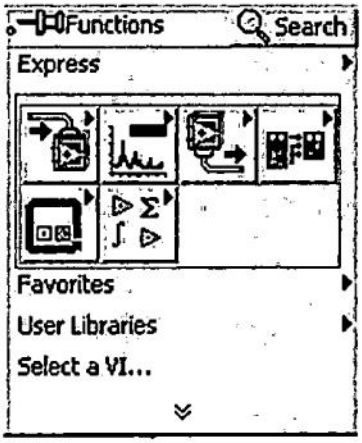
No	Tombol	Fugsi
1		Tombol ***Highlight Execution*** ini berfungsi untuk melihat jalannya aliran data pada *Block Diagram*.
2		Tombol ***Retain Wire Values*** ini berfungsi untuk menyimpan nilai data di setiap titik ketika program dijalankan.
3		Tombol ***Step Info*** ini berfungsi untuk menjalankan program setiap *step* (langkah) dari *node* ke *node*, sebuah *loop* atau subVI.
4		Tombol ***Step Over*** ini berfungsi untuk melompati step sebuah *loop* atau subVI.
5		Tombol ***Step Out*** ini berfungsi untuk keluar dari suatu *node* atau *loop* dan masuk ke node berikutnya.
6		Tombol ***Clean Up Diagram*** ini befungsi untuk merapikan garis dan menata ikon-ikon sehingga lebih ringkas dan menghemat ruang.

2.9.3. *Palet Control*

Gambar 2.10 *Palet Controls*

Palet Control hanya terdapat pada jendela *Front Panel*. Panel ini berfungsi untuk mengambil objek-objek yang akan terlihat oleh pengguna ketika program dijalankan. Pembaca dapat mencari objek dengan melihat pada katgori yang disediakan atau menggunakan bantuan *Search*.

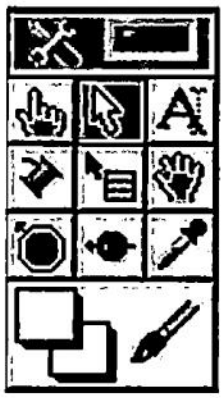
2.9.4. *Palet Function*



Gambar 2.11 *Palet Function*

Palet Function terdapat pada jendela *Block Diagram*. Palet ini digunakan untuk mengambil ikon-ikon guna pembuatan program di *Block Diagram*. Dalam palet ini tersedia tombol *Search* untuk membantu pencarian.

2.9.5. *Palet Tool*



Gambar 2.12 *Palet Tool*

Berbeda dengan kedua palet sebelumnya yang hanya bisa dimunculkan pada satu jendela tertentu, *palet tool* ini dapat dimunculkan di kedua jendela, baik di *Front Panel* maupun di *Block Diagram. Palet tool* dapat dimunculkan dengan cara menekan tombol *Shift* diikuti dengan klik kanan.

Isi *palet tool* ini berhubungan dengan mode pada *pointer mouse*, apakah *pointer mouse* akan digunakan untuk menempatkan teks, memberi warna, memilih objek, menggulung layar dan sebagainya. Masing-masing memiliki gambar *pointer* yang berbeda. Untuk lebih jelasnya, berikut keterangan fungsi masing-masing mode/*tool* pada *palet tool* ini.

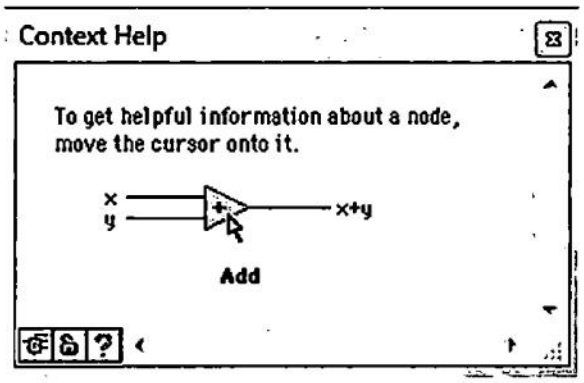
Tabel 2.3 Ikon tombol dan fungsi pada *palet tool*

No	Tool	Fungsi
1		*Operating Tool*, Berfungsi untuk mengubah data dari sebuah kontrol atau teks.
2		*Positioning Tool*, berfungsi untuk memilih, menggerakkan atau mengubah ukuran objek
3		*Labeling Tool*, berfungsi untuk mengedit teks dan menciptakan label.

4		*Wiring Tool*, berfungsi untuk menghubungkan objek-objek pada *block diagram.*
5		*Object Shortcut Menu Tool*, berfungsi untuk mengakses sebuah menu shortcut objek dengan tombol kiri *mouse.*
6		*Scrolling Tool*, berfungsi untuk menggulung layar.
7		*Breakpoint Tool*, berfungsi untuk menempatkan brakpoint pada VI, fungsi, *node*, kawat dan struktur untuk menghentikan eksekusi pada lokasi tersebut.
8		*Probe Tool*, berfungsi untuk mngukur nilai di suatu kawat pada *Block Diagram.*
9		*Color Copy Tool*, berfungsi untuk menyalin warna dan menempelkan warnanya pada dengan *Coloring tool.*
10		*Coloring Tool*, berfungsi untuk memberi warna ke sebuah objek.

2.10. Jendela *Context Help*

Jendela *Context Help* berfungsi untuk menampilkan informasi mengenai objek yang disentuh oleh *pointer mouse*. Untuk menampilkan fungsi *Context Help* dapat dilakukan dengan cara menekan tombol tanda tanya yang berada di pojok kanan atas.



Gambar 2.13 Jendela *Context Help*

2.11. Istilah istilah penting pada pemrograman LabVIEW

Dalam pemrograman LabVIEW terdapat banyak istilah istilah yang digunakan untuk merancang sebuah proyek, berikut ini beberapa istilah-istilah penting yang terdapat pada pemrograman LabVIEW:

a) G : diperoleh dari kata "*graphical*" yang merupakan sebutan untuk bahasa pemrograman LabVIEW yang menggunakan kode berbentuk grafis.
b) VI : diperoleh dari kata "*Virtual Instruments*" yang merupakan sebutan utuk program yang dibuat dengan menggunakan LabVIEW.
c) SbVI : sebagian besar kode program di LabVIEW adalah subVI. subVI ini sama seperti subrutin dalam bahasa pemrograman teks, yaitu sebuah VI di dalam VI. SubVI ini berbentuk ikon, atau kotak kecil dengan gambar yang unik di dalamnya, dengan kaki *input* di sebelah kiri dan kaki *output* berada di sebelah kanan.
d) *Front Panel*: adalah tampilan program. Objek-objek pada jendela ini akan terlihat oleh pengguna saat program dijalankan. Objek-objek pada *Front Panel* ini akan secara otomatis memiliki reprentasi ikonnya di *Block Diagram*, kususnya untuk objek-objek yang membawa data, baik data yang masuk dari pengguna ke program maupun data yang keluar dari program ke pengguna.
e) *Block Diagram* : adalah tempat pembutan program. Jendela ini tidak akan terlihat oleh pengguna saat program dijalankan. Pembuatan program disini

dilakukan dengan cara menempatkan beberapa *node* dan menghubungkannya satu sama lain.

f) *Node* : adalah semua objek di jendela *Block Diagram* yang memiliki *input*/output dan melakukan operasi tertentu ketika dijalankan, termasuk didalamnya subVI, terminal, struktur, dan fungsi.

g) *Terminal* : adalah ikon-ikon di *Block Diagram* yang mewakili objek-objek di *Front Panel*, dimana objek-objek tersebut membawa data, baik data yang masuk dari pengguna ke program, maupu data yang keluar dari program ke pengguna. Contoh dari *terminal* adalah *Control* dan *Indicator*.

h) *Control* : adalah semua objek di *Front Panel* yang memasukan data dari pengguna ke program. Disebut juga *Terminal Input*. Contoh dari *Control* adalah *knob*, tombol, sakelar, dan alat *input* lainnya.

i) *Indicator* : adalah semua objek di *Front Panel* yang mengeluarkan atau menampilkan data dari program ke pengguna. Disebut juga Terminal *Output*. Contoh *Indicator* adalah grafik dan LED.

j) Struktur : adalah semua bentuk alur pemrograman seperti perulangan, percabangan, urutan, dan lain-lain. Contoh struktur ini adalah *For Loop*, *While Loop*, *Sekunsial*, maupun *Case*. Struktur hanya terdapat pada jendela *Block Diagram*, berbentuk blok yang dapat diatur luasannya, dan hanya bekerja untuk ikon-ikon yang ditempatkan di dalamnya.

k) Fungsi : adalah semua kode-kode dasar yang telah disediakan untuk membuat subVI. Contoh fungsi seperti *Add*, *Substract*.

l) *Wire* : sering disebut dengan kawat, digunakan untuk menghubungkan ikon-ikon, sekaligus untuk menunjukkan aliran data dan tipe data. Beberapa warna kawat dan tipe datanya dapat dilihat dalam tabel berikut :

Tabel 2.4 Tipe data pada LabVIEW

Tipe Data	Skalar	1D Array	2D Array	Warna
Numerik- Float				Jingga
Numerik- Integer				Biru
Boolean				Hijau
String				Merah muda

m) Pemrograman *Dataflow* (aliran data) : yaitu konsep pemrograman yang akan mengeksekusi node ketika semua *input*nya telah tersedia. Ketika *node* ini telah selesai dieksekusi, maka data akan diteruskan dari *output node* tersebut ke *node* berikutnya.

2.12. Struktur Pemrograman LabVIEW

LabVIEW menyediakan bermacam-macam struktur pemrograman, diantaranya adalah *While Loop, For Loop, Shift Register,* dan *Case Structure.*

2.12.1. *While Loop*

Struktur *While Loop* memiliki 3 komponen utama, yaitu sebuah blok yang dapat diatur luasannya, sebuah terminal *input conditional (Stop if True)* dan sebuah terminal *output counter* (i).

Struktur *While Loop* ini akan terus mengeksekusi ikon-ikon di dalam bloknya berulang kali hingga terminal *input conditional (Sop if True)* mendapat nilai *False,* sebaliknya perulangan akan berhenti ketika mendapat nilai *True.*

Terminal *input conditional* tersebut dapat diubh dari *Stop if True* menjadi *Continue if True,* yang mana akan membuat kondisi yang berlawanan, yaitu hanya akan melakukan perulangan ketika terminal *input conditional* tersebut mendapatt nilai *True,* dan berhenti ketika mendapat nilai *False.*

2.12.2. *For Loop*

Struktur *For Loop* memiliki 3 komponen utama, yaitu sebuah blok yang dapat diatur luasannya, sebuah terminal *input* N dan sebuah terminal *output counter* (i).

Struktur *For Loop* ini akan mengeksekusi ikon-ikon di dalam bloknya berulang kali, dengan jumlah perulangan sebanyak nilai yang dimasukkan ke terminal *input* N. apabila nilai yang dimasukkan ke terminal *input* N sebesr 0, maka *For Loop* tidak akan melakukan perulangan. Begitu pula apabila tidak ada nilai yang dimasukkan ke terminal N maka *For Loop* juga tidak akan melakukan perulangan, kecuali ada sebuah data *array* yang dimasukkan melalui dending blok *For Loop* dan dibuat *auto-indexing.* Jumlah perulangan untuk cara *auto-indexing* ini adalah sebanyak jumlah data *array* tersebut.

Sama seperti pada *While Loop,* terminal *output counter* (i) akan memberikan nilai jumlah perulangan yang telah dilakukan. Pembaca juga dapat menambahkan terminal *input conditional* seperti pada struktur *While Loop,* dengan cara klik kanan pada dinding blok *For Loop,* dan pada menu *popup* yang muncul, pilih *Conditional Terminal.*

2.12.3. *Shift Register*

Struktur *Shift Register* ini tidak tersedia di *palet Functions,* karena penggunaannya hanya dapat diterapkan pada struktur *For Loop* dan *While Loop* saja. *Shift Register* ini digunakan untuk meneruskan data dari satu perulangan ke perulangan berikutnya. Bentuk *Shift Register* ini berupa pasangan terminal di dinding kiri dan kanan blok.

2.12.4. *Case Structure*

Struktur Case memiliki 2 atau lebih blok, namun hanya satu saja yang dieksekusi pada satu waktu. *Struktur Case* ini sama seperti instruksi *If, Then Else, Switch* atau *select case* pada pemrograman berbasis teks. Blok mana yang akan dieksekusi tergantung pada nilai *input* di *terminal selector* (?). Di bagian atas setiap blok terdapat label yang menunjukkan nilai *input* yang mengakktifkannya.

Tipe data nilai *input* ini bias berupa *Boolean* (*True* atau *False*), *Integer, String*, atau Enumerasi. Enumerasi dalah tipe data yang terdiri atas sekumpulan nilai.

2.12.5. *State Machine*

State machine adalah sebuah cara pemrograman yang memugkinkan program untuk merespon secara cerdas terhadap *input* yang diberikan. *State machine* yang baik adalah yang dapat memberikan semua kemungkinan kondisi yang akan terjadi untuk setiap keadaan awal dan *input* pemicuan tertentu. Pada LabVIEW, cara pemrograma *State Machine* dapat diimplementasikan secara mudah, yaitu dengan menggabungkan 3 buah struktur, yaitu *While Loop, Case Structure* dan *Shift Register.* (Artanto, 2012).