INTERAKSI ARDUINO DAN LABVIEW

DIAN ARTANTO

Catatan: Buku ini pernah diterbitkan oleh PT Elex Media Komputindo (2012), dan sudah tidak dicetak ulang.

KATA PENGANTAR

Puji syukur kepada Tuhan, hanya atas pertolonganNya saja buku ini bisa selesai dan berada di depan pembaca.

Buku ini mengambil judul **Interaksi Arduino dengan LabVIEW**. Dalam bukunya yang menarik, berjudul *Programming Interactivity*, Joshua Noble mendefinisikan kata interaksi sebagai pertukaran informasi antara 2 atau lebih partisipan aktif. Lebih jauh, interaksi bisa menghasilkan keuntungan yang luar biasa, seperti yang akan diuraikan berikut ini.

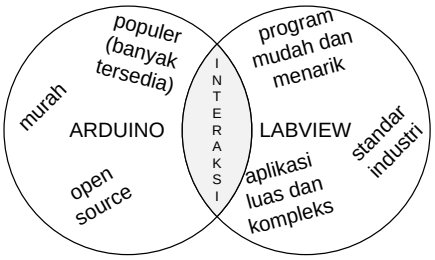
Arduino adalah sebuah papan rangkaian yang kecil, seukuran kartu nama, yang dapat diprogram dan mampu bekerja seperti layaknya sebuah komputer. Arduino ini disebut sebagai hardware mikrokontroler paling populer di dunia saat ini. Apa yang membuat Arduino ini sedemikian populer? Silahkan pembaca menemukannya di Bab 1.

LabVIEW merupakan software yang luar biasa mudah dan menarik. Tidak dibutuhkan keahlian khusus untuk memprogramnya, karena hanya tinggal menarik gambar yang tersedia dan menghubungkannya. Software ini telah diakui dan menjadi software standar untuk instrumentasi dan kontrol di industri, dengan tingkat ketelitian dan kehandalan yang tinggi. Bahkan di beberapa negara, software ini merupakan software wajib untuk pembelajaran pemrograman.

Namun demikian, di Indonesia nampaknya LabVIEW belum banyak digunakan. Hal ini mungkin disebabkan karena mahalnya biaya hardware dan softwarenya. Padahal, apabila dilihat dari kemampuan dan hasilnya, maka harga tersebut sangatlah wajar dan mungkin bisa dikatakan "murah", bila dibandingkan dengan software dan hardware yang lain.

Buku ini menawarkan solusi atas kendala biaya tersebut. Dengan melakukan interaksi antara Arduino dan LabVIEW, akan dapat dihasilkan sebuah perpaduan yang menguntungkan, di mana dari sisi hardware, harga Arduino sangat murah, sedangkan dari sisi software, dengan LabVIEW akan dihasilkan aplikasi yang tak terbatas dalam berbagai bidang, termasuk bidang otomotif, biomedis, komunikasi, energi, kontrol, akustik, mekatronik, vision, dll. Lebih jelasnya, perhatikan gambar ilustrasi berikut ini:

Gambar 1. Wilayah kelebihan dan kelemahan Arduino dan LabVIEW



Gambar 2. Dengan interaksi, membuat kelebihan Arduino dengan LabVIEW saling menambahkan dan menghilangkan kelemahan

Gambar 1 menunjukkan wilayah kelebihan dan kelemahan Arduino dan LabVIEW secara terpisah. Gambar 2 menunjukkan ketika interaksi dibuat antara Arduino dan LabVIEW, maka terlihat bahwa keduanya saling melengkapi, yaitu saling menambahkan kelebihan masing-masing, dan menghilangkan kekurangannya.

Interaksi yang dimaksud di sini, adalah menghubungkan Arduino dan LabVIEW melalui komunikasi serial di antara keduanya. Mengapa tidak menggunakan komunikasi paralel? Karena komunikasi serial ini diyakini lebih aman, mudah, dan sederhana dibanding dengan komunikasi paralel, dan kebanyakan komputer (laptop) sekarang ini sudah tidak memiliki port paralel, hanya port USB serial saja. Bagaimana membuat komunikasi serial di antara keduanya, menjadi topik utama dari buku ini.

Buku ini disusun secara sistematis, dengan semangat "*Learning by doing*" yaitu belajar melalui praktek.

Bab 1-4 berisi tentang pengenalan dan pemrograman Arduino dan bagaimana membuat komunikasi Arduino dengan komputer.

Bab 5-9 berisi tentang pengenalan dan pemrograman LabVIEW dan bagaimana membuat komunikasi antara LabVIEW dengan Arduino.

Bab 10 berisi tentang kemampuan Firmata Arduino, yaitu kemampuan yang membuat pemrograman tidak lagi perlu dilakukan di kedua sisi (Arduino dan LabVIEW), cukup hanya di satu sisi, yaitu di LabVIEW saja. Kemampuan ini membuat Arduino mirip seperti kartu akuisisi data, yang langsung bisa dikendalikan dari LabVIEW.

Bab 11 berisi tentang aplikasi komunikasi serial untuk membuat sebuah Ultrasonik Radar, yang mampu memetakan jarak objek di sekitarnya, melakukan penyimpanan data (data logging), dan menghasilkan indikator suara mengikuti jarak objeknya.

Bab 12 berisi tentang aplikasi komunikasi serial Firmata untuk membuat sebuah Infrared Radar, dengan kemampuan yang sama seperti Ultrasonik Radar. Bab 12 ini dimaksudkan untuk membandingkan komunikasi serial biasa dengan komunikasi serial Firmata.

Untuk informasi lebih lanjut mengenai isi buku ini, pembaca dapat mengunjungi blog ini: **interaksiarduinodanlabview.blogspot.com**.

Buku ini tidak lepas dari banyak kekurangan dan kesalahan akibat keterbatasan pengetahuan penulis. Untuk itu kritik, saran atau pertanyaan, silahkan dikirimkan ke alamat email berikut ini: **dian.artanto@gmail.com**. Atas kritik dan sarannya itu, penulis mengucapkan terimakasih.

Penulis juga merasa perlu untuk mengucapkan terimakasih kepada orang-orang yang berada di balik pembuatan buku ini, pertama kepada Bp. Whindy yang telah membantu penerbitan buku ini, kepada tim pembuat software dan hardware Arduino, LabVIEW dan Fritzing yang menjadi isi dari buku ini, kepada rekan-rekanku di PMSD (saat ini telah bergabung kembali menjadi USD - Universitas Sanata Dharma) atas sumbang saran dan idenya, serta kepada anak dan istriku yang setia mendampingi. Tidak lupa ucapan terimakasih buat pembaca buku ini, salam hangat dan sukses selalu.

DAFTAR ISI

BAB 1
PENGENALAN ARDUINO

1.1 Apa itu Arduino?

Dalam buku "*Getting Started with Arduino*", Arduino dituliskan sebagai sebuah *platform Physical Computing* yang *open source* pada *board* input output sederhana. Yang dimaksud dengan *platform* Physical Computing di sini adalah sebuah sistem fisik yang interaktif dengan penggunaan software dan hardware yang dapat mendeteksi dan merespon situasi dan kondisi yang ada di dunia nyata.

Sedangkan dari situs resminya di www.arduino.cc, Arduino didefinisikan sebagai sebuah *platform* elektronik yang *open source*, berbasis pada software dan hardware yang fleksibel dan mudah digunakan, yang ditujukan untuk para seniman, desainer, hobyist dan setiap orang yang tertarik dalam membuat objek atau lingkungan yang interaktif.

Nama Arduino di sini tidak hanya dipakai untuk menamai *board* rangkaiannya saja, tetapi juga untuk menamai bahasa dan software pemrogramannya, serta lingkungan pemrogramannya atau IDE-nya (IDE = *Integrated Development Environment*).

Kelebihan Arduino dari platform hardware mikrokontroler lain adalah:

1. IDE Arduino merupakan multi*platform*, yang dapat dijalankan di berbagai sistem operasi, seperti Windows, Macintosh dan Linux.
2. IDE Arduino dibuat berdasarkan pada IDE Processing, yang sederhana sehingga mudah digunakan.
3. Pemrograman Arduino menggunakan kabel yang terhubung dengan port USB, bukan port serial. Fitur ini berguna karena banyak komputer yang sekarang ini tidak memiliki port serial.
4. Arduino adalah hardware dan software *open source* – pembaca bisa mendownload software dan Gambar rangkaian tanpa harus membayar ke pembuat Arduino.
5. Biaya hardware cukup murah, sehingga tidak terlalu menakutkan untuk berbuat kesalahan.
6. Proyek Arduino ini dikembangkan dalam lingkungan pendidikan, sehingga bagi pemula akan lebih cepat dan mudah mempelajarinya.
7. Memiliki begitu banyak pengguna dan komunitas di internet yang dapat membantu setiap kesulitan yang dihadapi.

1.2 Instalasi IDE Arduino di Windows

Catatan: sekalipun Arduino bisa digunakan di sistem operasi yang lain, dalam buku ini penulis hanya menggunakan sistem operasi Windows. Apabila pembaca membutuhkan informasi instalasi Arduino di sistem operasi yang lain, silahkan melihatnya di **www.arduino.cc**.

Instalasi IDE Arduino di Windows ini sangat mudah, Anda hanya tinggal meng-ekstrak file software Arduino (**arduino-1.0-windows**) dan menempatkannya ke lokasi yang Anda inginkan, bisa di Desktop, di Program Files, di MyDocuments, atau tempat lainnya.

Kemudian untuk menjalankan software IDE ini, Anda tinggal meng-klik 2 kali pada icon Arduino (bertipe **Application**) yang ada di dalam folder hasil ekstraksi di atas.

Selanjutnya untuk membuat software IDE Arduino ini bisa berkomunikasi dengan *board* Arduino melalui kabel USB, maka Anda harus menginstal driver USB-nya.

1.3 Instalasi Driver USB Arduino di Windows

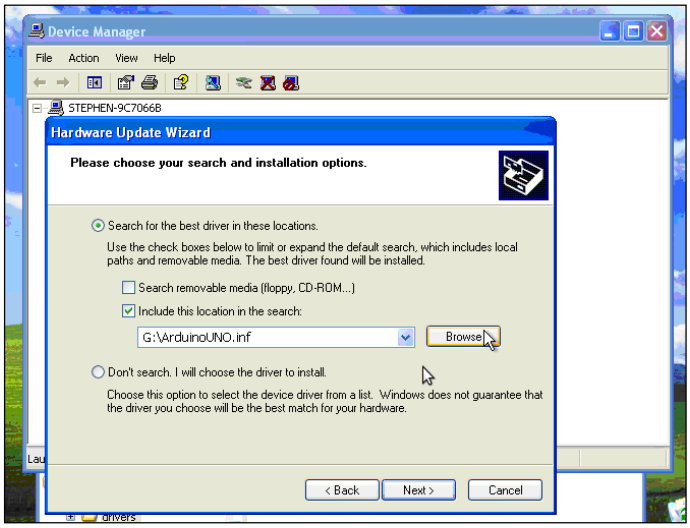
Berikut langkah-langkah instalasi driver USB Arduino:

1 Hubungkan *board* Arduino ke port USB komputer melalui kabel USB, maka pada layar komputer akan muncul jendela ***Found New Hardware Wizard***. Untuk pertanyaan terhubung dengan Windows Update, pilih pilihan ***No, not this time***, dan kemudian klik ***Next***.

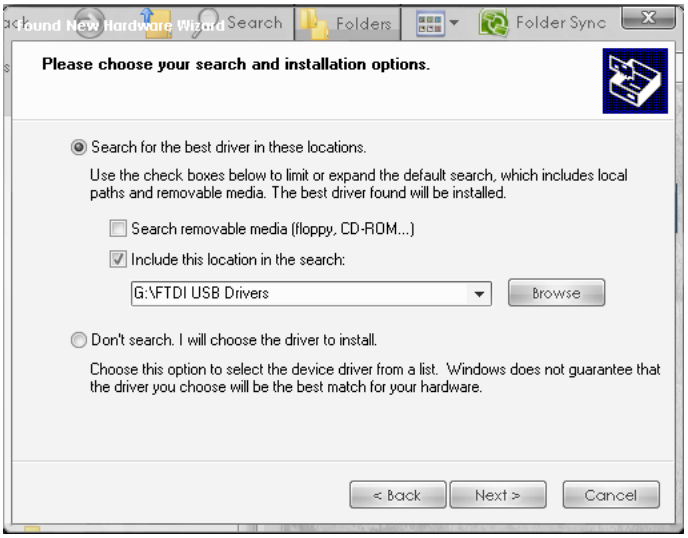
2 Pada jendela berikutnya, pilih pilihan ***Install from a list or specific location***, dan kemudian tekan tombol ***Next***.

3 Pada jendela berikutnya, apabila pembaca menggunakan *board* Arduino Uno, maka arahkan lokasi pencarian file pada folder **Drivers**, di dalam folder hasil ekstraksi arduino-1.0-windows, sampai diperoleh file **ArduinoUNO.inf** (lihat Gambar 1.1). Sedangkan apabila pembaca menggunakan *board* Arduino Duemilanove, maka arahkan lokasi pencarian file pada folder **Drivers**, sampai diperoleh **FTDI USB Drivers** (lihat Gambar 1.2).

4 Selanjutnya, tekan tombol ***Next*** hingga proses instalasi selesai. Khusus untuk Arduino Duemilanove, ada 2 file driver yang akan diinstalasi, yaitu driver **USB Serial Converter** dan **USB Serial Port**, yang semuanya ada di folder **FTDI USB Drivers**. Sedangkan untuk Arduino Uno hanya satu file driver saja, yaitu **ArduinoUno.inf**.



Gambar 1.1 Lokasi file driver untuk Arduino Uno

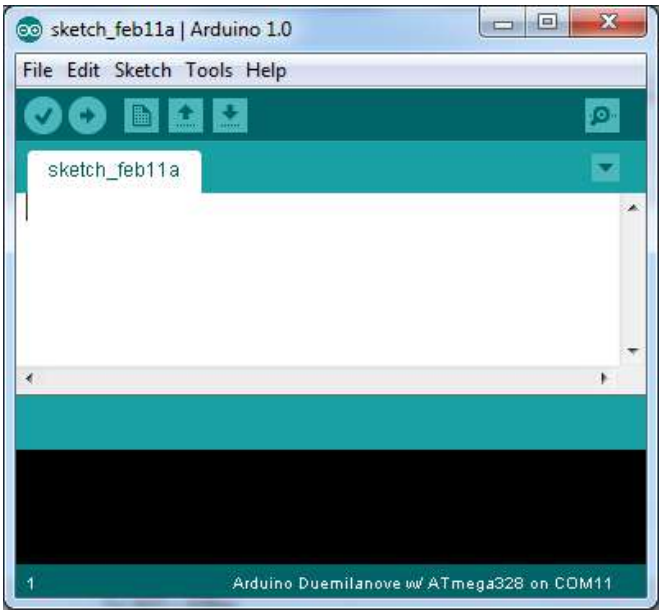


Gambar 1.2 Lokasi file driver untuk Arduino Duemilanove

Setelah instalasi IDE dan driver Arduino selesai dilakukan, maka langkah berikutnya adalah memahami cara penggunaan software IDE tersebut.

1.4 Lingkungan Pemrograman Arduino

Lingkungan pemrograman ini juga disebut IDE, atau *Integrated Development Environment*. Seperti telah disebutkan di awal, bahwa kelebihan Arduino adalah pada penggunaan IDE-nya yang mudah dikarenakan kesederhanaannya, seperti terlihat pada tampilan IDE Arduino versi 1.0 berikut ini:



Gambar 1.3 Tampilan IDE Arduino

Tampak ada 6 buah tombol pada Toolbar, dengan fungsi masing-masing seperti berikut:

No.	Tombol	Nama	Fungsi
1.		Verify	Menguji apakah ada kesalahan pada program atau sketch. Apabila sketch sudah benar, maka sketch tersebut akan

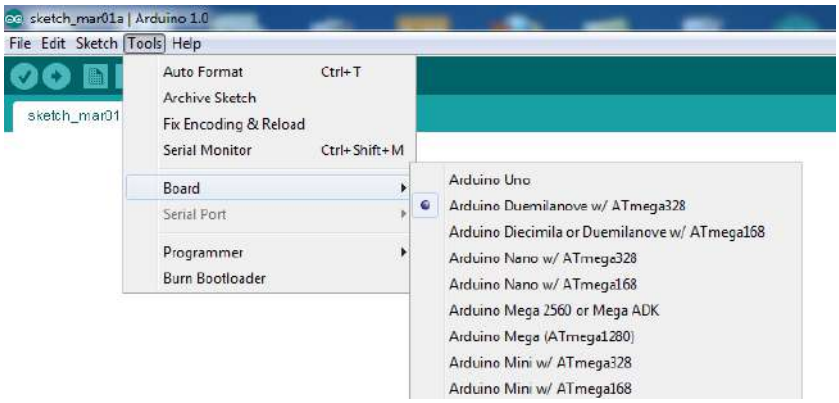
No.	Tombol	Nama	Fungsi
			dikompilasi. Kompilasi adalah proses mengubah kode program ke dalam kode mesin.
2.		Upload	Mengirimkan kode mesin hasil kompilasi ke *board* Arduino.
3.		New	Membuat sketch yang baru.
4.		Open	Membuka sketch yang sudah ada.
5.		Save	Menyimpan sketch.
6.		Serial Monitor	Menampilkan data yang dikirim dan diterima melalui komunikasi serial.

Keterangan: program di software Arduino sering disebut sebagai sketch.

Berikut ini langkah-langkah penggunaan IDE Arduino:

1. Hubungkan Arduino dengan port USB komputer melalui kabel USB.
2. Jalankan software IDE Arduino dengan meng-klik 2 kali icon Arduino.
3. Buat program atau sketch di jendela Editor Arduino.
4. Tekan tombol **Verify**. Apabila tidak terdapat kesalahan pada sketch maka sketch tersebut akan dikompilasi untuk menghasilkan kode mesin. Proses kompilasi ini selesai ketika muncul tulisan "***Done Compiling***" di bawah jendela Editor.
5. Setelah kode mesin dihasilkan, langkah berikutnya adalah meng-upload kode mesin tersebut ke *board* Arduino. Namun sebelum menekan tombol **Upload**, yang berada di samping kanan tombol

Verify, pastikan 2 hal berikut ini sudah benar, yaitu yang pertama adalah tipe *board* Arduino dan kedua adalah saluran serial port yang digunakan. Kedua hal ini dapat diatur dengan membuka menu **Tools**, dan pilih ***Board*** untuk mengatur tipe *board*nya, dan pilih **Serial Port** untuk mengatur saluran port COM yang digunakan.



Gambar 1.4 Atur tipe board dan saluran port di menu Tools

Catatan: Apabila pilihan Serial Port-nya belum dapat dibuka, maka ada 2 kemungkinan penyebabnya, yaitu karena *board* Arduinonya belum terhubung dengan komputer, atau yang kedua karena driver Arduino belum ter-instal (lihat bab 1.3 untuk meng-instalnya).

6. Setelah tipe *board* Arduino dan saluran serial port dipilih secara benar, maka tekan tombol **Upload**. Maka proses pengiriman kode mesin ke *board* Arduino akan berlangsung. Proses Upload ini selesai ketika muncul tulisan "***Done Uploading***" di bawah jendela Editor Arduino.

7. Terakhir, simpan sketch dengan menekan tombol **Save**.

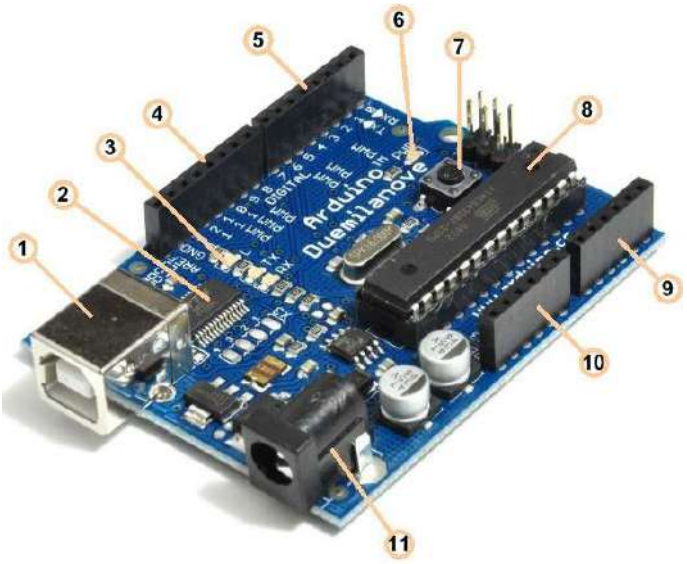
BAB 2
HARDWARE ARDUINO

2.1 Bagian-bagian Arduino

Sebenarnya Arduino memiliki berbagai macam versi. Pembaca dapat melihat versi-versi tersebut di **www.arduino.cc/en/Main/Hardware**.

Beberapa versi yang cukup banyak tersedia di pasaran lokal di antaranya adalah versi Arduino Duemilanove, yang disebut sebagai Arduino 2009, versi Arduino Uno, yang disebut sebagai Arduino 2010 dan Arduino Mega, yang memiliki kapasitas memori dan kaki I/O yang lebih banyak. Untuk membatasi halaman, maka buku ini tidak membahas Arduino Mega, sekalipun sebenarnya tidak jauh beda dan bisa juga digunakan untuk membuat aplikasi di buku ini. Lebih jauh mengenai Arduino Mega, silahkan melihat informasinya di alamat web di atas.

Pada dasarnya, bagian-bagian komponen dari Arduino Duemilanove dan Arduino Uno sama, hanya berbeda pada IC konverter USB ke serialnya. Apabila Arduino Duemilanove menggunakan IC FTDI, maka Arduino Uno menggunakan IC ATMega8U2 sebagai konverter USB ke serialnya. Gambar 2.1 berikut ini menunjukkan bagian-bagian komponen Arduino Duemilanove, beserta keterangannya ditunjukkan dalam Tabel 2.1.



Gambar 2.1 Bagian-bagian Arduino Duemilanove

Tabel 2.1 Keterangan Bagian Arduino Duemilanove

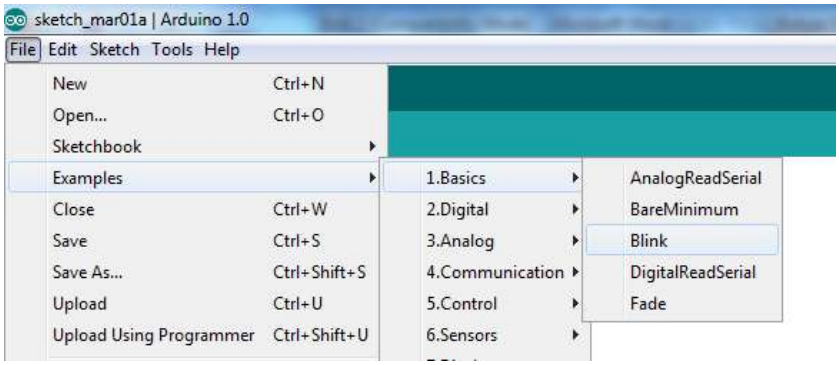
No	Keterangan
1.	Port USB
2.	IC Konverter Serial-USB (FTDI)
3.	LED untuk test output kaki D13
4.	Kaki-kaki I/O Digital (D8 – D13)
5.	Kaki-kaki I/O Digital (D0 – D7)
6.	LED Indikator catu daya
7.	Tombol Reset
8.	Mikrokontroler ATmega 328
9.	Kaki-kaki Input Analog (A0 – A5)
10.	Kaki-kaki catu daya (5V, GND)
11.	Terminal catu daya (6 - 9V)

2.2 Contoh Pertama: LED berkedip

Cara belajar yang paling cepat adalah dengan mempelajari contoh yang diberikan. Software IDE Arduino telah menyediakan begitu banyak contoh program atau sketch, sehingga memudahkan bagi pemula untuk memahami bahasa pemrograman Arduino dengan lebih cepat.

Salah satu contoh sketch yang populer adalah membuat LED berkedip atau **Blink**. Berikut langkah-langkah pembuatannya:

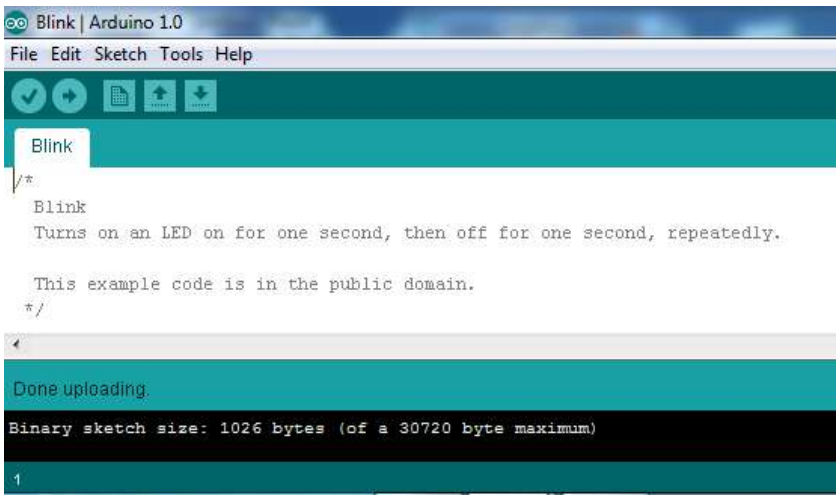
1. Hubungkan *board* Arduino dengan port USB komputer.
2. Jalankan software IDE Arduino dengan meng-klik 2 kali icon Arduino.
3. Buka menu **File**, dan pilih **Examples**, **Basics**, kemudian **Blink**.



Gambar 2.2 Dari menu File, pilih Examples, Basics kemudian Blink

4. Tekan tombol **Verify** untuk mengkompilasi sketch menjadi kode mesin, dan tunggu hingga muncul tulisan "***Done Compiling***".
5. Setelah kompilasi selesai, pastikan tipe ***Board*** dan **Serial Port** di menu **Tools**, sesuai dengan yang digunakan.
6. Tekan tombol **Upload** untuk memasukkan kode mesin ke *board* Arduino, dan tunggu hingga muncul tulisan "***Done Uploading***".

Gambar 2.3 Proses kompilasi telah selesai



Gambar 2.4 Proses Upload telah selesai

7. Setelah proses upload selesai, maka *board* Arduino akan menjalankan program. Perhatikan bahwa LED di kaki 13 (berlabel L) sekarang berkedip, yaitu menyala dan padam secara bergantian.

Catatan: ada 2 hal yang menyenangkan ketika menggunakan Arduino. Pertama, pembaca tidak perlu menyediakan sumber tegangan lagi seperti adaptor atau batere untuk menghidupkan Arduino. Cukup dengan menghubungkan Arduino ke komputer melalui kabel USB. Untuk kebutuhan arus-arus rangkaian yang kecil, seperti rangkaian dalam buku ini, maka suplai melalui kabel USB ini cukup memadai. Namun untuk rangkaian yang membutuhkan arus yang besar, disarankan untuk menggunakan Adaptor 9-12V. Secara otomatis, ketika ada sumber tegangan dimasukkan, maka Arduino akan menggunakan sumber tegangan tersebut sebagai ganti suplai dari komputer.

Kedua, pembaca tidak lagi perlu menambahkan rangkaian programmer eksternal untuk menanamkan program ke memori Arduino. Dengan adanya bootloader yang dimiliki Arduino, maka Arduino mampu menanam program ke memorinya sendiri saat berkomunikasi secara serial dengan komputer melalui kabel USB.

2.3 Kaki I/O Arduino

Program LED berkedip di atas menggunakan kaki D13 sebagai kaki output yang bisa menyalakan dan memadamkan LED. Output seperti ini disebut output tipe digital, karena hanya memiliki 2 kemungkinan kondisi, yaitu nyala atau padam, atau dalam bentuk sinyal berarti 5V atau 0V, High atau Low.

Kebalikan dari output adalah input. Apabila kaki output bertugas mengeluarkan sinyal, maka kaki input bertugas memasukkan sinyal.

Sedangkan lawan dari digital adalah analog. Analog artinya memiliki lebih dari 2 kondisi; dalam bentuk sinyal bisa berarti semua sinyal dengan tegangan antara 0 sampai 5V.

Baik Arduino Duemilanove maupun Arduino Uno memiliki kaki input output atau kaki I/O digital sebanyak 14 kaki, yaitu di kaki D0 sampai D13, dan memiliki 6 kaki I/O analog, yaitu di kaki A0 sampai A5 (lihat gambar 2.1). Khusus untuk kaki-kaki I/O digital, setiap kakinya bisa dijadikan sebagai input atau output. Namun untuk kaki I/O analog, hanya bisa dijadikan sebagai input saja.

Sebagai input analog, keenam kaki ini memiliki tambahan rangkaian ADC di dalamnya, dengan resolusi ADC sebesar 10 bit. ADC atau *analog to digital converter* ini adalah pengubah sinyal analog menjadi digital. Resolusi 10 bit artinya untuk tegangan masukan 0-5V, nilai digital yang dihasilkan memiliki jangkauan nilai dari 0-1023.

Arduino tidak memiliki kaki output analog. Untuk menghasilkan sinyal analog, digunakan sebuah metode PWM (*pulse width modulation*), yaitu sebuah cara untuk menghasilkan sinyal analog dari sinyal digital, dengan cara mengatur lebar pulsa sinyal digital pada frekuensi yang tinggi, sehingga rata-rata tegangannya menghasilkan nilai yang bervariasi antara 0-5V, untuk nilai PWM antara 0-255. Karena dibuat dari sinyal digital, maka kaki output analog dari PWM ini berada di kaki I/O digital, yang jumlahnya sebanyak 6 kaki, yaitu kaki D3, D5, D6, D9, D10 dan D11.

Catatan: apabila dibutuhkan, keenam kaki input analog, yaitu kaki A0 hingga A5 dapat dijadikan sebagai kaki I/O digital, sehingga kaki I/O digital bisa menjadi 20 buah. Pengubahan tipe kaki ini dilakukan dengan hanya mengganti nama kaki I/O analog dengan nomor lanjutan dari kaki I/O digital yang sudah ada, yaitu dari angka 14 hingga 19 untuk kaki A0 hingga A5. Setelah menjadi digital, maka selanjutnya perlu pengaturan apakah kaki tersebut sebagai input atau sebagai output. Pengaturan ini dilakukan dengan instruksi **pinMode**, yang akan dijelaskan di bab 3.

Ada 2 buah kaki I/O Arduino yang dikhususkan untuk komunikasi dengan komputer, yaitu kaki D0 dan kaki D1. Kaki D0 digunakan untuk menerima data atau RX (*receive*), sedangkan kaki D1 digunakan untuk mengirimkan data atau TX (*transmit*). Komunikasi dilakukan secara serial, dengan kecepatan komunikasi harus diatur baik di sisi Arduino maupun di sisi komputer. Instruksi untuk komunikasi serial ini akan dijelaskan di bab 4.

Untuk lebih mengenal kaki-kaki I/O Arduino ini, maka berikut ini diberikan contoh-contoh aplikasi input output kaki I/O Arduino, baik untuk sinyal digital maupun analog. Berikut daftar komponen yang diperlukan:

Tabel 2.2 Daftar Komponen Aplikasi I/O Arduino

No	Komponen	Jumlah
1.	Arduino dan kabel USB	1 buah
2.	*Breadboard*	1 buah
3.	Tombol	1 buah
4.	LDR	1 buah
5.	LED	2 buah
6.	Resistor 330 ohm	2 buah
7.	Resistor 10k ohm	2 buah
8.	Kabel Jumper	10 potong

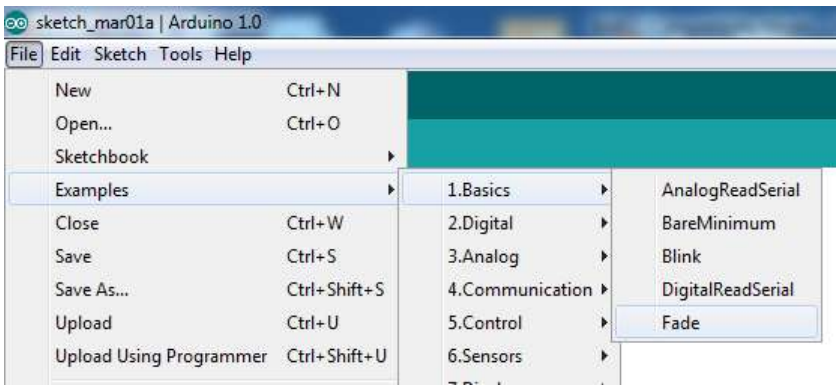
Keterangan: *Breadboard* adalah sebuah media penyambungan rangkaian yang memiliki pola sambungan tertentu. Pola sambungan tersebut menghubungkan lubang-lubang di bagian tengah *breadboard* secara vertikal, dan menghubungkan lubang-lubang di bagian tepi *breadboard* secara horizontal, seperti ditunjukkan dalam gambar 2.5 berikut ini:

Gambar 2.5 Breadboard dan pola sambungannya

2.4 Contoh Kedua: LED terang dan redup

Apabila contoh pertama menggunakan kaki D13 untuk menghasilkan Output Digital, maka contoh kedua ini menggunakan kaki D9, untuk menghasilkan Output Analog dengan metode PWM. Dengan kaki PWM tersebut akan dibuat aplikasi untuk membuat LED semakin terang dan kemudian semakin redup secara terus-menerus. Berikut langkah-langkah pembuatannya:

1. Hubungkan *board* Arduino dengan port USB komputer.
2. Jalankan software IDE Arduino dengan meng-klik 2 kali icon Arduino.
3. Buka menu **File**, dan pilih **Examples**, **Basics**, kemudian **Fade**.
4. Tekan tombol **Verify** untuk mengkompilasi sketch menjadi kode mesin, dan tunggu hingga muncul tulisan "***Done Compiling***".

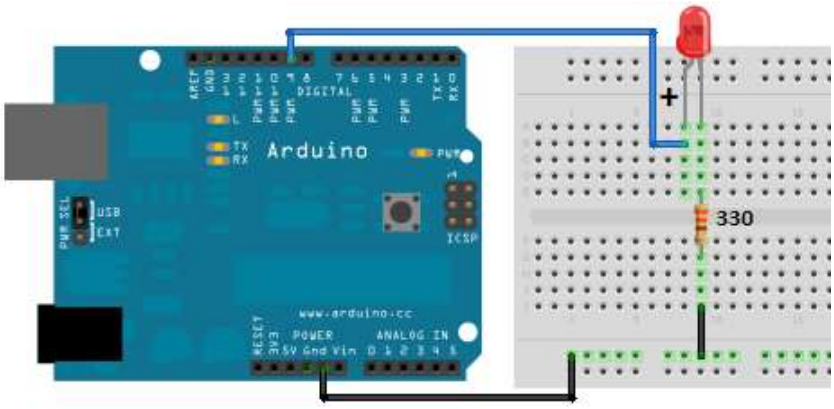


Gambar 2.6 Dari menu File, pilih Examples, Basics kemudian Fade

5. Setelah kompilasi selesai, pastikan tipe ***Board*** dan **Serial Port** di menu **Tools**, sesuai dengan yang digunakan.

6. Tekan tombol **Upload** untuk memasukkan kode mesin ke *board* Arduino, dan tunggu hingga muncul tulisan "***Done Uploading***".

7. Setelah proses upload selesai, maka *board* Arduino akan menjalankan program. Untuk bisa melihat hasil program tersebut, susun rangkaian dengan komponen sebuah LED dan sebuah resistor 330 Ω pada *breadboard*, seperti gambar berikut ini:



Gambar 2.7 Rangkaian Output Analog

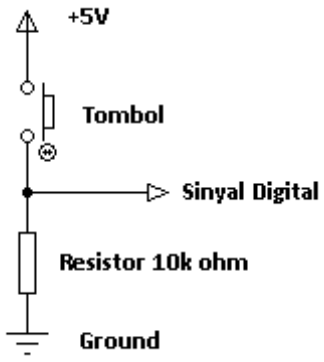
Dari gambar rangkaian di atas, perhatikan bahwa kaki LED yang panjang (bertanda positif) dihubungkan dengan kaki D9, sedangkan kaki LED yang pendek dihubungkan dengan resistor 330 Ω. Kaki LED yang panjang biasa disebut kaki Anode dan kaki LED yang pendek disebut kaki Katode.

Saran: sebaiknya saat menyusun rangkaian ini, kabel USB Arduino ke komputer dilepas. Setelah rangkaian selesai disusun secara benar, hubungkan kembali kabel USB Arduino ke komputer.

Amati apakah LED bisa menyala semakin terang dan kemudian semakin redup, dan kembali lagi terus-menerus demikian secara berulang?

2.5 Contoh Ketiga: Input Digital Tombol

Apabila contoh pertama dan kedua menunjukkan aplikasi kaki I/O sebagai Output Digital dan Output Analog, maka contoh ketiga ini menunjukkan aplikasi kaki I/O Arduino sebagai Input Digital, yang akan membaca sinyal digital dari luar. Sinyal digital tersebut dihasilkan oleh rangkaian tombol dan resistor 10kΩ yang disusun secara seri sebagai pembagi tegangan 5V seperti gambar skematik berikut ini:

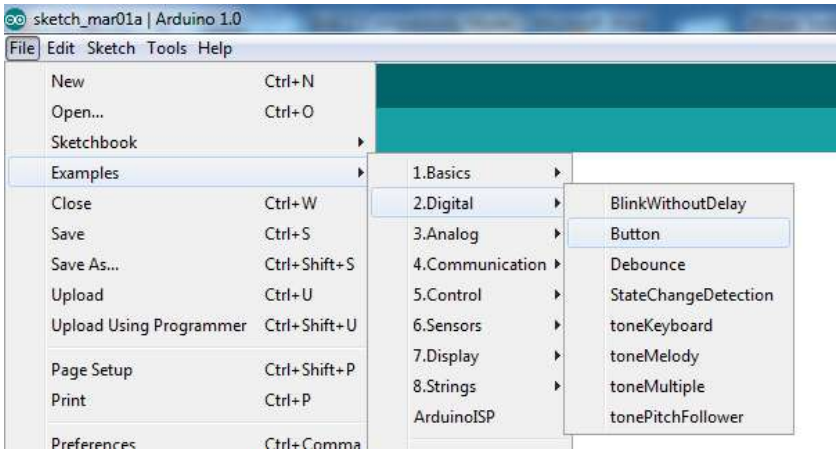


Gambar 2.8 Skematik rangkaian tombol dan resistor 10kΩ

Dengan rangkaian di atas, maka tegangan pada sinyal digital akan sebesar 5V (High) ketika tombol ditekan, dan sebesar 0V (Low) ketika tombol dilepas.

Sebagai indikator pembacaan sinyal digital oleh kaki input Arduino ini, digunakan LED di kaki 3, yang akan menyala apabila tombol ditekan, dan padam ketika tombol dilepas. Berikut langkah-langkah pembuatannya:

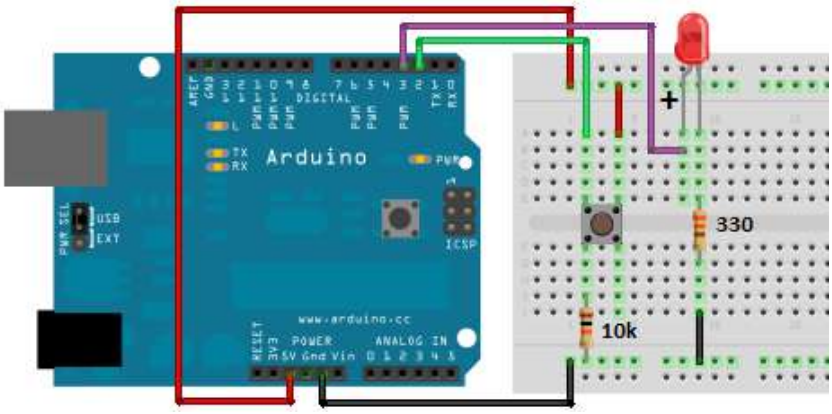
1. Hubungkan *board* Arduino dengan port USB komputer.
2. Jalankan software IDE Arduino dengan meng-klik 2 kali icon Arduino.
3. Buka menu **File**, dan pilih **Examples**, **Digital**, kemudian **Button**.



Gambar 2.9 Dari menu File, pilih Examples, Digital kemudian Button

4. Tekan tombol **Verify** untuk mengkompilasi sketch menjadi kode mesin, dan tunggu hingga muncul tulisan "***Done Compiling***".
5. Setelah kompilasi selesai, pastikan tipe ***Board*** dan **Serial Port** di menu **Tools**, sesuai dengan yang digunakan.
6. Tekan tombol **Upload** untuk memasukkan kode mesin ke *board* Arduino, dan tunggu hingga muncul tulisan "***Done Uploading***".

7. Setelah proses upload selesai, maka *board* Arduino akan menjalankan program. Untuk bisa melihat hasil program tersebut, susun rangkaian dengan komponen berupa sebuah tombol, sebuah resistor 10kΩ, sebuah LED dan sebuah resistor 330 ohm pada *breadboard* seperti gambar 2.10 berikut ini.



Gambar 2.10 Rangkaian Input Digital

Perhatikan pada gambar rangkaian di atas, bahwa jumlah kaki tombol ada 4. Kaki kiri atas tombol dihubungkan dengan kaki D2 Arduino, dan kaki kiri bawah dihubungkan dengan resistor 10k ohm. Kemudian kaki kanan atas dihubungkan dengan tegangan +5V. Perlu diketahui, bahwa saat tombol tidak ditekan, kaki kiri atas sudah terhubung dengan kaki kiri bawah, namun tidak terhubung dengan kaki kanan atas. Begitu tombol ditekan, maka semua kaki tersebut akan saling terhubung.

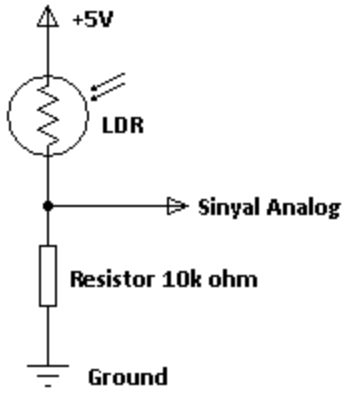
Setelah rangkaian di atas selesai disusun, tekanlah tombol tersebut. Apakah LED di D3 bisa menyala ketika tombol ditekan, dan padam ketika tombol dilepas?

2.6 Contoh Keempat: Input Analog LDR

Apabila contoh ketiga menggunakan kaki I/O Arduino sebagai Input Digital, dengan tombol sebagai penghasil sinyal digitalnya, maka contoh keempat ini menggunakan kaki I/O Arduino sebagai Input Analog, dengan LDR sebagai penghasil sinyal analognya.

LDR atau *Light Dependent Resistor* merupakan komponen yang peka terhadap cahaya. Ketika mendapat cahaya, maka hambatan pada LDR menjadi kecil, sebaliknya ketika tidak mendapat cahaya, maka hambatan LDR semakin besar.

Untuk menghasilkan sinyal analog, maka LDR disusun seri dengan resistor 10k ohm untuk dijadikan rangkaian pembagi tegangan 5V seperti gambar skematik berikut ini:

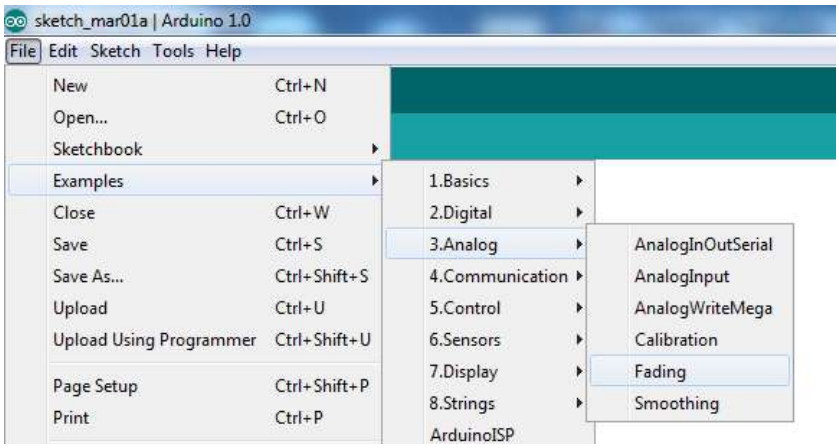


Gambar 2.11 Skematik rangkaian LDR dan resistor 10kΩ

Dengan rangkaian di atas, maka tegangan pada sinyal analog akan bervariasi tergantung pada nilai LDR. Untuk membaca sinyal analog ini digunakan kaki Input Analog Arduino, yaitu A0.

Sebagai indikator pembacaan sinyal analog ini, digunakan LED di kaki D9, yang akan menyala semakin terang ketika tegangan sinyal analog semakin besar, dan akan semakin redup ketika tegangan sinyal analog semakin kecil. Berikut langkah-langkah pembuatannya:

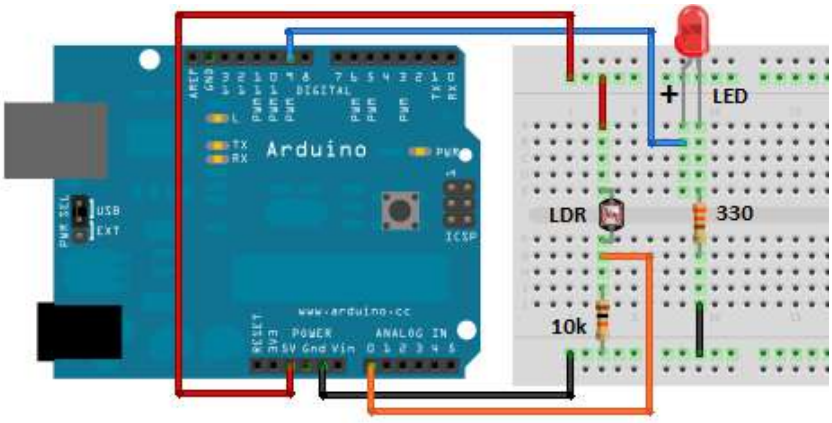
1. Hubungkan *board* Arduino dengan port USB komputer.
2. Jalankan software IDE Arduino dengan meng-klik 2 kali icon Arduino.
3. Buka menu **File**, dan pilih **Examples**, **Analog**, kemudian **Fading**.
4. Tekan tombol **Verify** untuk mengkompilasi sketch menjadi kode mesin, dan tunggu hingga muncul tulisan "***Done Compiling***".
5. Setelah kompilasi selesai, pastikan tipe ***Board*** dan **Serial Port** di menu **Tools**, sesuai dengan yang digunakan.
6. Tekan tombol **Upload** untuk memasukkan kode mesin ke *board* Arduino, dan tunggu hingga muncul tulisan "***Done Uploading***".



Gambar 2.12 Dari menu File, pilih Examples, Analog kemudian Fading

7. Setelah proses upload selesai, maka *board* Arduino akan menjalankan program. Untuk bisa melihat hasil program tersebut, susun rangkaian dengan komponen berupa sebuah LDR, sebuah resistor 10k ohm,

sebuah LED dan sebuah resistor 330 ohm pada *breadboard* seperti gambar berikut ini:



Gambar 2.13 Rangkaian Input Analog

Setelah rangkaian di atas selesai disusun, buka tutup LDR dengan tangan dan perhatikan nyala LED. Apakah LED semakin terang ketika LDR dibuka, dan semakin redup ketika LDR ditutup dengan tangan?

2.7 Contoh Kelima: Input Output Digital dan Analog

Sebagai penutup pada Bab 2 ini, maka semua rangkaian input output pada contoh-contoh sebelumnya akan digabungkan menjadi satu rangkaian. Berikut langkah-langkah pembuatannya:

1. Hubungkan *board* Arduino dengan port USB komputer.
2. Jalankan software IDE Arduino dengan meng-klik 2 kali icon Arduino.
3. Ketikkan sketch program Arduino seperti berikut:

contoh_kelima

```
void setup()
{
  pinMode(2,INPUT);
  pinMode(3,OUTPUT);
  pinMode(9,OUTPUT);
}
void loop()
{
  int TOMBOL=digitalRead(2);
  if (TOMBOL==HIGH)
  {digitalWrite(3,HIGH);}
  else
  {digitalWrite(3,LOW);}
  int LDR=analogRead(0);
  int LED2=map(LDR,0,1023,0,255);
  analogWrite(9,LED2);
}
```

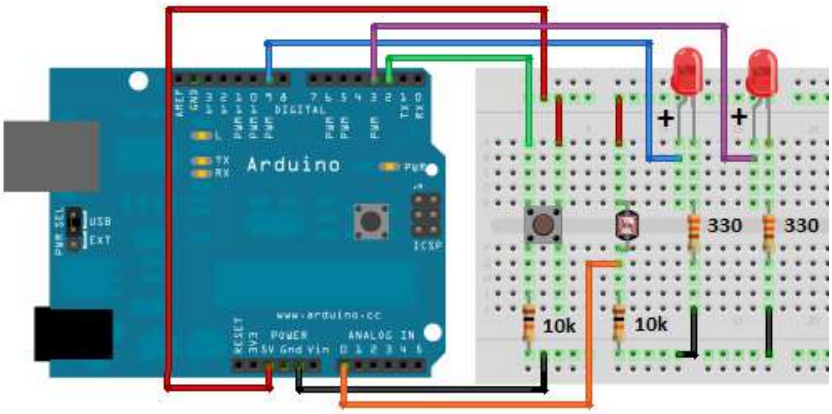
Gambar 2.14 Sketch Arduino untuk contoh kelima

4. Tekan tombol **Verify** untuk mengkompilasi sketch tersebut menjadi kode mesin, dan tunggu hingga muncul tulisan "***Done Compiling***".

5. Setelah kompilasi selesai, pastikan tipe ***Board*** dan **Serial Port** di menu **Tools**, sesuai dengan yang digunakan.

6. Tekan tombol **Upload** untuk memasukkan kode mesin ke *board* Arduino, dan tunggu hingga muncul tulisan "***Done Uploading***".

7. Setelah proses upload selesai, maka *board* Arduino akan menjalankan program. Untuk bisa melihat hasil program tersebut, susun rangkaian dengan komponen berupa sebuah tombol, sebuah LDR, 2 buah LED, 2 buah resistor 330 ohm dan 2 buah resistor 10k ohm pada *breadboard* seperti gambar 2.15 berikut:



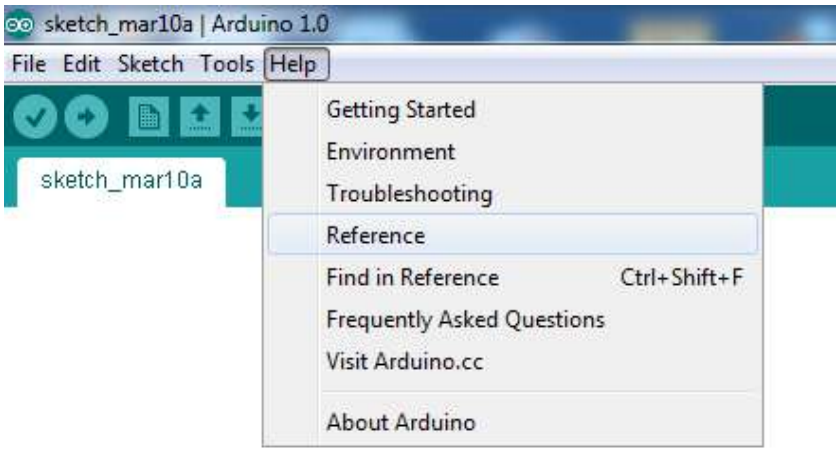
Gambar 2.15 Rangkaian input output analog dan digital

Setelah rangkaian di atas selesai disusun, amati LED pertama (yang kiri), apakah LED pertama tersebut menyala ketika tombol ditekan dan padam ketika tombol dilepas? Buka tutup LDR dengan tangan dan amati LED kedua (yang kanan), apakah LED tersebut redup ketika LDR ditutup, dan terang ketika LDR dibuka?

BAB 3
SOFTWARE ARDUINO

3.1 Bahasa Pemrograman Arduino

Untuk mempelajari bahasa pemrograman Arduino, pembaca dapat membuka **Reference** di menu **Help**.



Gambar 3.1 Membuka Reference di menu Help

Ada 3 bagian utama dalam bahasa pemrograman Arduino, yaitu Struktur, Variabel dan Fungsi. Berikut penjelasan masing-masing bagian.

3.2 Struktur Program Arduino

Bagian struktur program Arduino ini meliputi kerangka program, sintaks program, kontrol aliran program, dan operator. Berikut uraian singkat dari keempat hal tersebut.

3.2.1 Kerangka Program

Kerangka program Arduino sangat sederhana, yaitu terdiri dari 2 blok. Blok pertama adalah **void setup()**, dan blok kedua adalah **void loop()**. Berikut keterangan masing-masing:

- Blok **void setup()**: berisi kode program yang hanya dijalankan sekali sesaat setelah Arduino dihidupkan atau di-reset. Merupakan bagian persiapan atau inisialisasi program.
- Blok **void loop()**: berisi kode program yang akan dijalankan terus-menerus. Merupakan tempat untuk program utama.

Catatan: Selain kedua blok tersebut, apabila diperlukan bisa ditambahkan blok **Function**. Blok **Function** adalah bagian dari program yang dibuat terpisah untuk melaksanakan tugas khusus atau fungsi tertentu. Apabila blok **Function** ini menghasilkan data yang digunakan di blok lain, maka di depan nama blok **Function** tersebut harus diberi tipe data. Namun apabila blok **Function** ini tidak menghasilkan data, maka di depan nama **Function** tersebut diberi tulisan “void”.

3.2.2 Sintaks Program

- Baik blok **void setup()**, **void loop()** maupun blok **function** harus diberi tanda kurung kurawal buka “**{**“ sebagai tanda awal program di blok itu dan kurung kurawal tutup “**}**” sebagai tanda akhir program.

- Tanda kurung kurawal buka dan kurung kurawal tutup tersebut juga digunakan pada blok kontrol program, seperti **if**, **if-else**, **for-loop**, **while-loop** dan **do-while-loop**.
- Untuk menandai akhir dari sebuah baris kode program digunakan tanda titik koma ";".

Catatan: kurangnya tanda kurung kurawal buka, kurung kurawal tutup, dan titik koma ini pada program akan menyebabkan *compiler error*.

- Agar program lebih jelas, biasanya diberi komentar keterangan. Sebagai tanda komentar, apabila lebih dari satu baris, digunakan tanda "**/***" sebagai awal dan tanda "***/**" sebagai akhir komentar.
- Namun apabila komentarnya hanya satu baris, maka digunakan tanda "**//**" sebagai pemisah komentar dari kode program.

Sebagai contoh kerangka dan sintaks, perhatikan program berikut ini:

```
/* Berikut ini program Blink yang sedikit
   diubah untuk contoh struktur Arduino */
void setup()
{
  pinMode(13, OUTPUT); // kaki 13 terhubung dengan LED
}
void loop()
{
  digitalWrite(13, HIGH);   // nyalakan LED
  delay(1000);              // tunggu 1 detik
  padam();                  // memanggil function
}
void padam()                // function padam
{
  digitalWrite(13, LOW);    // padamkan LED
  delay(1000);              // tunggu 1 detik
}
```

3.2.3 Kontrol Aliran Program

Kontrol aliran program ini meliputi instruksi-instruksi yang digunakan untuk membuat percabangan dan perulangan. Instruksi percabangan di antaranya adalah **if**, **if-else**, **switch case**, **break**, **continue**, **return** dan **goto**. Sedangkan instruksi perulangan di antaranya adalah **for-loop**, **while-loop**, **do-while-loop**. Berikut penjelasannya:

- Instruksi **if** dan **if-else** akan menguji apakah kondisi tertentu dipenuhi atau tidak. Jika tidak dipenuhi, maka instruksi berikutnya akan dilompati, tetapi jika dipenuhi, maka instruksi berikutnya akan dijalankan. Instruksi ini memiliki format penulisan sebagai berikut:

```
if(x==100)                // jika x sama dengan 100
{digitalWrite(13,HIGH);}  // nyalakan LED
else                      // jika tidak sama
{digitalWrite(13,LOW);}   // padamkan LED
```

Catatan: perhatikan bahwa untuk menguji nilai x apakah sama dengan 100, digunakan operator "==" sebagai ganti "=". Operator "=" digunakan untuk mengisi nilai x dengan nilai 100, sedangkan, operator "==" digunakan untuk membandingkan apakah nilai x sama dengan 100.

- Instruksi **switch case** selalu diikuti dengan **break**, memiliki format penulisan sebagai berikut:

```
switch(x)                 // uji nilai x
{ case 100:               // jika nilainya 100
    analogWrite(9,100);   // beri pwm=100 di kaki 9
    break;                // keluar dari switch case
  case 200:               // jika nilainya 200
    analogWrite(9,200);   // beri pwm=200 di kaki 9
    break;                // keluar dari switch case
  default:                // jika tidak ada yang cocok
    analogWrite(9,0);}    // beri pwm=0 di kaki 9
```

- Selain digunakan pada instruksi **switch case**, instruksi **break** juga digunakan untuk melompat keluar dari blok perulangan.
- Berbeda dengan instruksi **break**, instruksi **continue** digunakan untuk melanjutkan ke perintah berikutnya tanpa keluar dari blok perulangan.
- Instruksi **return** digunakan untuk menghentikan proses sebuah blok **Function** dan kembali ke program utamanya, sambil membawa hasil proses apabila blok **Function** tersebut menghasilkan data.
- Instruksi **goto** sama seperti instruksi **break**, yaitu digunakan untuk melompat keluar dari perulangan, hanya bedanya, lokasi lompatan **goto** bisa diatur dengan cara menempatkan labelnya pada lokasi yang diinginkan.
- Instruksi perulangan **for-loop** akan membuat perulangan pada bloknya dalam jumlah tertentu, yaitu sebanyak nilai counternya. Format penulisan instruksi **for-loop** ini adalah sebagai berikut:

```
for (int i=0;i<256,i++)// deklarasi i, uji apakah i<256?
{analogWrite(9,i);     // jika ya, naikkan i, dan ulangi
 delay(100);}          // blok ini, jika tidak, berhenti
```

- Instruksi perulangan **while-loop** akan membuat perulangan pada bloknya selama persyaratannya dipenuhi, dan berhenti bila persyaratannya tidak dipenuhi. Berikut format **while-loop**:

```
int x=0;              // deklarasi x
while(x<200)          // persyaratan while
{// lakukan sesuatu  // bila dipenuhi, ulangi blok ini
 x++;}                // jika tidak, berhenti
```

- Instruksi perulangan **do-while-loop** sama seperti while-loop, yaitu mengulangi bloknya selama persyaratannya dipenuhi. Hanya

bedanya, karena persyaratan diletakkan di akhir blok, maka perulangan do-while loop ini akan menjalankan bloknya sekali sekalipun persyaratan tidak dipenuhi. Berikut formatnya:

```
do                  // do-while-loop, menjalankan
{delay(100);        // bloknya paling tidak sekali,
 x=analogRead(0);   // menguji apakah syarat dipenuhi
}while (x<500);     // jika tidak, maka tidak diulangi
```

3.2.4 Operator

Berikut ini beberapa operator pada program Arduino:

1 Operator Aritmetika di Arduino meliputi perkalian (*), pembagian (/), penjumlahan (+), pengurangan (-) dan modulo (%). Modulo adalah perhitungan untuk mendapatkan sisa dari hasil pembagian.

2 Hasil operasi Aritmetika ini tergantung pada tipe data operand. Contoh, untuk pembagian operand 9 dengan operand 4 akan menghasilkan nilai 2, apabila tipe data kedua operand adalah **int**, atau menghasilkan nilai 2,25 apabila tipe data kedua operand adalah **float**.

3 Jika operandnya berbeda tipe datanya, maka tipe data yang paling besar yang digunakan. Sebagai contoh, apabila operand 9 bertipe data **int**, dan operand 4 bertipe data **float**, maka hasil perhitungan akan bertipe **float**, atau 2,25.

4 Karena operasi Aritmetika tergantung pada tipe data operandnya, maka pilihlah tipe data operand yang cukup besar untuk menampung hasil operasinya. Jika tipe data tidak cukup besar, maka hasil operasi bisa menjadi salah karena terjadi *rollover* (nilai datanya berputar kembali), seperti contoh berikut ini:

```
int x=32767; // 37267 adalah nilai maks tipe int
x = x + 1;   // ketika ditambah 1, hasilnya -32768
```

5 Operator Perbandingan di Arduino meliputi berikut ini:

```
x == y;  // x sama dengan y
x != y;  // x tidak sama dengan y
x < y;   // x lebih kecil dari y
x > y;   // x lebih besar dari y
x <= y;  // x lebih kecil sama dengan y
x >= y;  // x lebih besar sama dengan y
```

6 Operator Boolean di Arduino meliputi logika AND (&&), OR (||) dan NOT (!), dengan penggunaan seperti contoh berikut ini:

```
if (x > 0 && x < 5)  // logika AND
if (x > 0 || y > 0)  // logika OR
if (!x > 0)          // logika NOT
```

7 Untuk melakukan operasi perhitungan pada level bit (biner) dari sebuah variabel, gunakan operator-operator Bitwise berikut ini:

```
& (bitwise and)
| (bitwise or)
^ (bitwise xor)
~ (bitwise not)
<< (bitshift left)
>> (bitshift right)
```

8 Untuk mempersingkat penulisan instruksi, pembaca dapat menggunakan operator Compound berikut ini:

```
x++;     // sama seperti x=x+1
x--;     // sama seperti x=x-1
x += y;  // sama seperti x=x+y
x -= y;  // sama seperti x=x-y
x *= y;  // sama seperti x=x*y
x /= y;  // sama seperti x=x/y
x &= y;  // sama seperti x=x&y
x |= y;  // sama seperti x=x|y
```

3.3 Variabel dan Tipe Data

Berikut ini hal-hal yang berkaitan dengan variabel dan tipe data pada program Arduino:

- Sebuah variabel digunakan untuk menyimpan sebuah nilai.
- Sebuah variabel bisa diberi nama apa saja asalkan tidak sama dengan salah satu instruksi atau kode program di Arduino.
- Sebuah variabel harus dideklarasikan terlebih dulu sebelum digunakan. Mendeklarasikan variabel ini berarti menentukan tipe data variabel dan juga memberi nilai awal (opsional) pada variabel.
- Tempat di mana variabel tersebut dideklarasikan menentukan lingkup variabel. Variabel yang dideklarasikan sebelum void setup() disebut sebagai variabel global. Variabel global dapat digunakan oleh kode program di semua bagian. Sedangkan variabel yang dideklarasikan di dalam sebuah bagian, maka hanya kode-kode program di bagian itu saja yang dapat menggunakan variabel tersebut. Variabel terakhir ini disebut sebagai variabel lokal.
- Berikut ini beberapa tipe data pada program Arduino:

Tabel 3.1 Keterangan tipe data variabel

Tipe data	Ukuran data	Jangkauan nilai
Boolean	1 bit	True / False
Char	8 bit	-128 s/d 127
Byte	8 bit	0 s/d 255
Int	16 bit	-32768 s/d 32767
Word	16-bit	0 s/d 65535
Long	32 bit	-2147483648 s/d 2147483647
Float	32 bit	$-3{,}4028235 \times 10^{38}$ s/d $3{,}4028235 \times 10^{38}$

- Apabila sebuah variabel hanya menyimpan sebuah nilai, maka untuk menyimpan kumpulan nilai, digunakan sebuah array.
- Setiap nilai di dalam array dapat dipanggil dengan nama array dan nomor indeks nilai tersebut, di mana nomor indeks dimulai dari 0.
- Sama seperti sebuah variabel, sebuah array harus dideklarasikan dulu tipe datanya dan diberi nilai awal (opsional).
- Contoh deklarasi array:

```
int arrayku[]={nilail, nilai2, nilai3,...}
```

- Atau bisa juga:

```
int arrayku[5];  // tipe data int, sebanyak 6
arrayku[3] = 8;  // diisi 8 pada indeks ke 4
```

- Mengambil sebuah nilai di dalam array:

```
x=arrayku[3];     // nilai x sebesar 8
```

Sebagai contoh penggunaan array, perhatikan program berikut ini:

```
byte atur[]={0,50,100,150,200,255}; // array 6 nilai
void setup()
{pinMode(9,OUTPUT);}           // menugasi kaki 9
void loop()
{for(int i=0;i<6;i++)          // dari i=0 sampai i<6
  {analogWrite(9,atur[i]);     // mengambil nilai array
   delay(200);})               // tunda waktu 200 ms
```

- Apabila array menyimpan nilai berupa angka, maka untuk menyimpan teks atau kumpulan huruf digunakan sebuah string. Berikut ini contoh sebuah string:

```
char stringku[]="arduino";
```

9 Untuk mengubah tipe data, pembaca dapat menggunakan penulisan sebagai berikut:

```
char(x);   // membuat nilai x bertipe data char
int(y);    // membuat nilai y bertipe data int
float(z);  // membuat nilai z bertipe data float
```

10 Konstanta adalah sebuah nilai yang telah ditetapkan dan dimasukkan sebagai kode program Arduino. Beberapa konstanta Arduino yang sering digunakan adalah INPUT/OUTPUT, HIGH/LOW, true/false. Berikut keterangannya:

INPUT/OUTPUT: konstanta ini digunakan pada instruksi **pinMode()** untuk menugasi sebuah kaki I/O digital sebagai input atau output.

HIGH/LOW: konstanta ini digunakan untuk menentukan level tegangan pada sebuah kaki I/O digital. HIGH = 5V dan LOW = 0V.

true/false: perhatikan bahwa konstanta ini dituliskan dengan huruf kecil. Nilai false selalu identik dengan 0. Sedangkan nilai true adalah semua nilai selain 0, bisa berarti 1, -1, 2, 200 dll.

3.4 Fungsi

Bagian Fungsi ini meliputi fungsi input output digital, input output analog, *advanced* I/O, fungsi waktu, fungsi matematika (termasuk random, instruksi byte dan bit), serta fungsi komunikasi. Berikut uraian singkat dari keenam fungsi tersebut:

3.4.1 Input Output Digital

Ada 3 instruksi yang digunakan dalam Input Output Digital, yaitu **pinMode()**, **digitalRead()** dan **digitalWrite()**. Berikut penjelasannya:

- Instruksi pinMode() ditempatkan di void setup(), digunakan untuk mengatur sebuah kaki I/O digital, untuk dijadikan INPUT atau OUTPUT, dengan format penulisan sebagai berikut:

```
pinMode(3,OUTPUT); // menjadikan D3 sebagai OUTPUT
```

Catatan: Bila tidak ada pengaturan oleh **pinMode()**, maka kaki-kaki I/O digital Arduino secara default bekerja sebagai kaki INPUT.

- Chip ATmega pada Arduino memiliki resistor internal pullup 20 kΩ di setiap kakinya, yang dapat diaktifkan secara software. Dengan mengaktifkan resistor ini pada sebuah kaki yang dijadikan INPUT, maka akan menyederhanakan rangkaian masukan, yaitu rangkaian masukan bisa hanya berupa sebuah tombol yang terhubung dengan ground. Dengan rangkaian seperti ini, ketika tombol tidak ditekan, maka sinyal yang dibaca oleh kaki INPUT tersebut adalah HIGH, sedangkan ketika tombol ditekan, maka sinyal yang dibaca adalah LOW. Berikut instruksi untuk mengaktifkannya:

```
void setup()
{pinMode(2,INPUT);      // menjadikan D2 sebagai INPUT
 digitalWrite(2,HIGH);}//mengaktifkan resistor pullup
```

Catatan: instruksi digitalWrite() di atas tidak membuat kaki D2 sebagai OUTPUT, tetapi mengaktifkan resistor internal pullup di kaki tersebut.

- Kaki I/O yang dijadikan INPUT memiliki nilai impedansi yang tinggi, sedangkan kaki I/O yang dijadikan OUTPUT memiliki impedansi yang rendah, yang dapat menghantarkan arus maksimum sebesar 40mA. Arus sebesar ini dapat menyalakan LED dengan cukup terang (jangan lupa menambahkan resistor agar LED tidak putus), namun tidak cukup kuat untuk menggerakkan rele dan motor (kecuali motor servo, karena di motor servo sudah ada penguatnya).

- Setelah kaki I/O dikonfigurasikan menjadi INPUT, maka untuk membaca sinyal digital yang masuk, digunakan instruksi **digitalRead()**. Berikut format penulisan instruksinya:

```
int tombol=digitalRead(2);//membaca sinyalmasuk di D2
```

- Instruksi di atas berarti membaca sinyal masuk di kaki D2 dan kemudian menyimpannya di variabel tombol dengan tipe data int.
- Untuk kaki I/O yang dikonfigurasikan menjadi OUTPUT, maka untuk mengeluarkan sinyal digital, digunakan instruksi **digitalWrite()**. Berikut format penulisan instruksinya:

```
digitalWrite(3,HIGH); //mengeluarkan sinyal HIGH di D3
```

- Instruksi di atas berarti menghasilkan sinyal digital High (+5V) pada kaki D3.

Sebagai contoh, berikut ini program untuk membaca sinyal digital dari sebuah tombol yang dihubungkan dengan D2 dan ground, dan memadamkan sebuah LED yang terhubung dengan D3 setiap kali tombol ditekan, dan menyalakan LED setiap kali tombol dilepas:

```
void setup()
{pinMode(2,INPUT);     // menjadikan D2 sebagai INPUT
 digitalWrite(2,HIGH);//mengaktifkan resistor pullup
 pinMode(3,OUTPUT);}   //menjadikan D3 sebagai OUTPUT
void loop()
{int tombol=digitalRead(2);//membaca sinyal di D2
digitalWrite(3,tombol);}//mengeluarkan nilainya di D3
```

3.4.2 Input Output Analog

Secara umum hanya ada 2 instruksi yang digunakan, yaitu **analogRead()** dan **analogWrite()**. Berikut penjelasannya:

- Seperti telah diuraikan di Sub Bab 2.3, ada 6 buah kaki input analog Arduino, yaitu A0, A1, A2, A3, A4 dan A5.
- Sedangkan untuk kaki output analog, sebenarnya Arduino tidak memiliki kaki output analog. Namun dengan metode PWM, maka beberapa kaki I/O digital Arduino dapat digunakan untuk menghasilkan sinyal analog. Ada 6 buah kaki yang bisa menghasilkan PWM, yaitu kaki D3, D5, D6, D9, D10 dan D11.
- Berbeda dengan input output digital yang harus diatur fungsinya sebagai INPUT atau sebagai OUTPUT dengan instruksi pinMode(), maka pada input output analog ini tidak diperlukan pengaturan fungsi seperti itu , karena kaki input dan outputnya terpisah.
- Untuk membaca sinyal analog yang masuk, digunakan instruksi **analogRead()**. Berikut format penulisan instruksinya:

```
int baca=analogRead(0); // membaca analog di kaki A0
```

- Instruksi di atas berarti membaca sinyal masuk di kaki A0 dan kemudian menyimpannya di variabel baca dengan tipe data int. Nilai input analog tersebut memiliki jangkauan antara 0 hingga 1023.
- Untuk mengeluarkan sinyal analog PWM, digunakan instruksi **analogWrite()**. Berikut format penulisan instruksinya:

```
analogWrite(3,255); // menghasilkan PWM 255 di kaki D3
```

- Instruksi di atas berarti menghasilkan sinyal analog sebesar 255 (+5V) pada kaki D3. Nilai jangkauan PWM antara 0 hingga 255.

3.4.3 Advanced I/O

Ada 5 instruksi yang digunakan dalam fungsi advanced I/O ini, yaitu **tone()**, **notone()**, **shiftOut()**, **shiftIn()**, **pulseIn()**. Berikut penjelasannya:

- Instruksi **tone()** digunakan untuk membangkitkan gelombang kotak pada sebuah kaki dengan frekuensi dan durasi yang dapat ditentukan. Kaki tersebut dapat dihubungkan dengan buzzer atau speaker untuk menghasilkan nada. Hanya satu kaki saja yang dapat digunakan untuk instruksi **tone()** pada satu waktu.
- Instruksi **notone()** digunakan untuk menghentikan instruksi **tone()**.
- Instruksi **shiftOut()** digunakan untuk menggeser keluar (mengeluarkan) satu byte data ke salah satu kaki, satu bit pada satu waktu, mengikuti nilai clocknya.
- Instruksi **shiftIn()** digunakan untuk menggeser kedalam (memasukkan) satu byte data di salah satu kaki, satu bit pada satu waktu, mengikuti nilai clocknya.
- Instruksi **pulseIn()** digunakan untuk membaca lebar pulsa (bisa pulsa High atau Low) yang diterima pada sebuah kaki.

3.4.4 Waktu

Ada 4 instruksi yang digunakan dalam fungsi waktu, yaitu **millis()**, **micros()**, **delay()**, **delayMicroseconds()**. Berikut penjelasannya:

- Instruksi **millis()** digunakan untuk menghasilkan banyaknya milidetik sejak program Arduino dijalankan. Angka hasil instruksi tersebut akan *overflow* (kembali ke 0) setelah kira-kira 50 hari.
- Instruksi **micros()** digunakan untuk menghasilkan banyaknya mikrodetik sejak program Arduino dijalankan. Angka hasil instruksi tersebut akan *overflow* (kembali ke 0) setelah kira-kira 70 menit.
- Instruksi **delay()** digunakan untuk menghentikan program untuk sejumlah waktu tertentu yang dinyatakan dalam parameternya, dengan satuan waktu dalam milidetik.

- Instruksi **delayMicroseconds()** digunakan untuk menghentikan program untuk sejumlah waktu tertentu yang dinyatakan dalam parameternya, dengan satuan waktu dalam mikrodetik.

3.4.5 Matematika

Ada beberapa instruksi yang digunakan dalam fungsi matematika, yaitu **min()**, **max()**, **abs()**, **constrain()**, **map()**, **pow()**, **sqrt()**, dan 3 instruksi dalam fungsi trigonometri, yaitu **sin()**, **cos()**, **tan()**, serta instruksi **random()**, **byte()** dan **bit()**. Berikut penjelasannya:

- Instruksi **min()** digunakan untuk mendapatkan nilai yang paling kecil dari 2 nilai yang dinyatakan dalam parameternya.
- Instruksi **max()** digunakan untuk mendapatkan nilai yang paling besar dari 2 nilai yang dinyatakan dalam parameternya.
- Instruksi **abs()** digunakan untuk menghasilkan nilai mutlak atau nilai yang lebih besar dari 0 untuk setiap nilai dalam parameternya.
- Instruksi **constrain()** digunakan untuk mendapatkan semua nilai yang masuk dalam jangkauan daerahnya, dengan batas bawah dan batas atas dinyatakan dalam parameternya. Berikut contohnya:

```
int sensorku=constrain(sensorku,10,100);
//nilai sensorku dibatasi antara 10 sampai 100 saja
```

- Instruksi **map()** digunakan untuk memetakan nilai dari daerah jangkauannya yang lama ke dalam daerah jangkauan yang baru. Berikut contohnya:

```
void setup() {}
void loop() {
int dataku=analogRead(0);// membaca sinyal analog 0-1023
dataku=map(dataku,0,1023,0,255);//memetakan ke 0-255
analogWrite(9,dataku);}//mengeluarkan nilainya ke kaki 9
```

- Instruksi **pow()** digunakan untuk menghasilkan pangkat tertentu dari sebuah nilai dalam parameternya.
- Instruksi **sqrt()** digunakan untuk menghasilkan nilai akar pangkat dua dari sebuah nilai dalam parameternya.
- Instruksi **sin()** digunakan untuk menghasilkan fungsi sinus dari sebuah nilai dalam parameternya (dalam radian).
- Instruksi **cos()** digunakan untuk menghasilkan fungsi cosinus dari sebuah nilai dalam parameternya (dalam radian).
- Instruksi **tan()**digunakan untuk menghasilkan fungsi tangen dari sebuah nilai dalam parameternya (dalam radian).
- Instruksi **random()** digunakan untuk membangkitkan nilai acak dengan daerah nilai yang dapat diatur.
- Instruksi **lowByte()** digunakan untuk menghasilkan 4 bit terkecil (LSB) dari data 1 byte, atau 1 byte terkecil dari data 1 word.
- Instruksi **highByte()** digunakan untuk menghasilkan 4 bit terbesar (MSB) dari data 1 byte, atau 1 byte terbesar dari data 1 word.
- Instruksi **bitRead()** digunakan untuk membaca nilai bit pada urutan tertentu dari sebuah bilangan.
- Instruksi **bitWrite()** digunakan untuk mengubah nilai bit pada urutan tertentu dari sebuah bilangan.
- Instruksi **bitSet()** digunakan untuk memberi nilai 1 pada bit di urutan tertentu dari sebuah bilangan.
- Instruksi **bitClear()** digunakan untuk memberi nilai 0 pada bit di urutan tertentu dari sebuah bilangan.
- Instruksi **bit()** digunakan untuk menghitung nilai dari bit tertentu.

3.4.6 Komunikasi

Fungsi ini digunakan untuk berkomunikasi dengan komputer melalui port serial. Kaki Arduino yang digunakan untuk fungsi ini adalah kaki D0 (RX) dan kaki D1 (TX). Beberapa instruksi yang digunakan adalah **begin()**, **available()**, **read()**, **print()**, **println()** dan w**rite()**. Berikut penjelasannya:

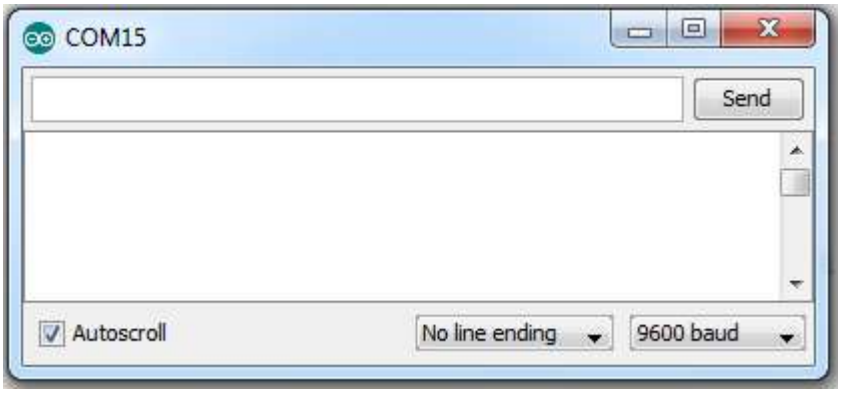
- Instruksi **Serial.begin()** digunakan untuk mengatur baudrate atau kecepatan komunikasi, umumnya nilainya adalah 9600.
- Instruksi **Serial.available()** digunakan untuk mendapatkan jumlah karakter atau byte yang telah diterima di serial port.
- Instruksi **Serial.read()** digunakan untuk membaca data yang telah diterima di serial port.
- Instruksi **Serial.print()** digunakan untuk mencetak data ke serial port dalam bentuk teks ASCII. Untuk data angka, setiap digitnya akan menggunakan satu karakter ASCII. Untuk data teks, setiap hurufnya akan menggunakan satu karakter ASCII.
- Instruksi **Serial.println()** sama seperti instruksi **Serial.print()** dengan penambahan **Enter**.
- Instruksi **Serial.write()** digunakan untuk mengirimkan data dalam bentuk biner, satu byte data setiap pengiriman.

Agar lebih jelas, bab berikut ini akan membahas aplikasi komunikasi ini.

BAB 4
KOMUNIKASI SERIAL ARDUINO

4.1 Serial Monitor

Komunikasi Serial merupakan inti dari buku ini, di mana dengan komunikasi ini, Arduino dapat berinteraksi dengan komputer melalui berbagai macam software, yang salah satunya adalah software LabVIEW. Bab ini akan membahas contoh dan aplikasi komunikasi serial Arduino ini dan juga penggunaan Serial Monitor yang ada di software Arduino untuk menampilkan data yang diterima dan dikirimkan dari komputer.



Gambar 4.1 Jendela Serial Monitor

Untuk bisa membuka jendela Serial Monitor, ada 2 syarat, yaitu:

1. Arduino harus terhubung dengan komputer.
2. Menekan tombol bergambar kaca pembesar yang terletak di pojok paling kanan pada Toolbar.

Agar Serial Monitor bisa menampilkan data yang diterima dan dikirimkan dari komputer, ada 2 syarat, yaitu:

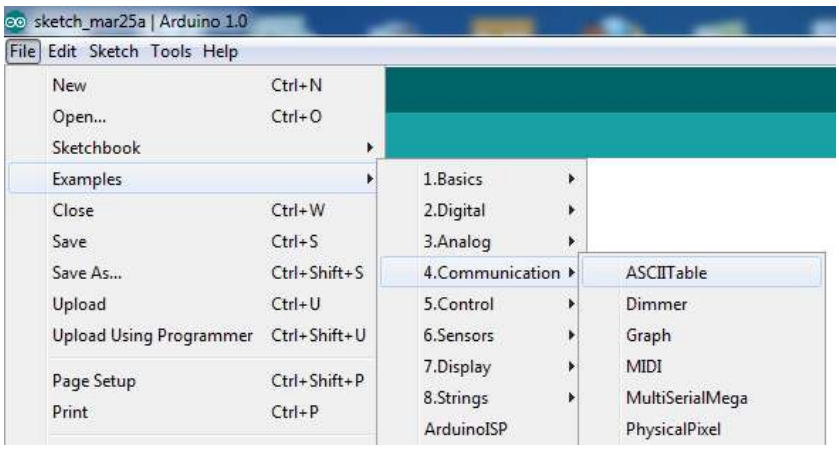
1. Saluran serial port yang digunakan Arduino harus benar. Pemilihan saluran ini dapat dilakukan dengan membuka menu **Tools**, dan kemudian **Serial Port**.
2. Kecepatan komunikasi atau baud rate pada Serial Monitor harus sama dengan baud rate pada program Arduino.

Berikut contoh-contoh aplikasi komunikasi serial dan penggunaan Serial Monitor dalam setiap aplikasi tersebut.

4.2 Contoh #1: ASCII Table

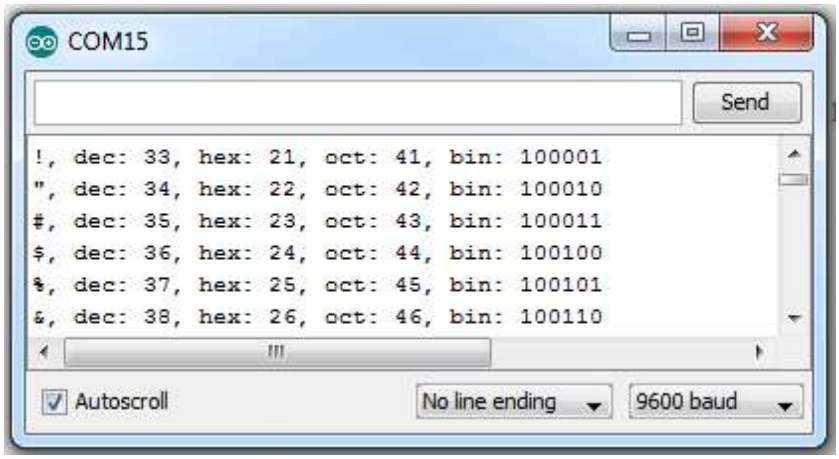
Contoh pertama aplikasi komunikasi serial ini adalah pengiriman karakter ASCII dari Arduino ke komputer, mulai dari karakter ASCII ke 33 hingga ke 126, yang merupakan karakter-karakter ASCII yang dapat ditampilkan. Berikut langkah-langkah pembuatannya:

1. Hubungkan Arduino ke komputer dengan kabel USB.
2. Buka menu **File**, **Examples**, **Communication**, kemudian **ASCII Table**.
3. Pilih **Board** dan **Serial Port** yang digunakan pada menu **Tools**.
4. Tekan tombol **Verify** bergambar centang di pojok kiri Toolbar.
5. Setelah muncul pesan "*Done Compiling*", berikutnya tekan tombol **Upload** yang berada di sebelah kanan tombol **Verify**.



Gambar 4.2 Membuka program ASCII Table

6. Setelah muncul pesan "*Done Uploading*", berikutnya buka Serial Monitor dengan meng-klik tombol bergambar kaca pembesar di pojok kanan Toolbar. Secara default, baudrate pada Serial Monitor adalah 9600, yang mana telah sesuai dengan baudrate pada program ASCII Table ini, sehingga seharusnya Serial Monitor akan menampilkan data seperti gambar berikut ini:



Gambar 4.3 Serial Monitor menampilkan data ASCII Table

4.3 Lebih Jauh tentang ASCII

ASCII atau *American Standard Code for Information Interchange* adalah sebuah kode alfanumerik yang digunakan untuk mengirimkan huruf, sImbol, angka dan karakter-karakter special yang tidak tercetak, antara komputer dan pirantinya, contoh piranti seperti printer, keyboard, dll. Kode ASCII ini terdiri dari 128 buah kode berukuran 7 bit yang berbeda.

Kode dari 000 0000 hingga 001 1111 digunakan untuk karakter-karakter tidak tercetak atau instruksi mesin khusus, seperti BS (Backspace), LF (Line feed), CR (Carriage return), ESC (Escape), dll. Kode dari 010 0000 hingga 111 1111 digunakan untuk karakter-karakter tercetak seperti huruf A-Z, angka 0-9, simbol-simbol seperti &, #, *, dll. Untuk lebih jelasnya, silahkan melihat tabel berikut:

Tabel 4.1 Kode ASCII tidak tercetak

ASCII	DEC	CHAR	ASCII	DEC	CHAR	ASCII	DEC	CHAR
0000000	0	NUL	0001011	11	VT	0010110	22	SYN
0000001	1	SOH	0001100	12	FF	0010111	23	ETB
0000010	2	STX	0001101	13	CR	0011000	24	CAN
0000011	3	ETX	0001110	14	SO	0011001	25	EM
0000100	4	EOT	0001111	15	SI	0011010	26	SUB
0000101	5	ENQ	0010000	16	DLE	0011011	27	ESC
0000110	6	ACK	0010001	17	DC1	0011100	28	FS
0000111	7	BEL	0010010	18	DC2	0011101	29	GS
0001000	8	BS	0010011	19	DC3	0011110	30	RS
0001001	9	HT	0010100	20	DC4	0011111	31	US
0001010	10	LF	0010101	21	NAK			

Tabel 4.2 Kode ASCII tercetak

ASCII	DEC	CHAR	ASCII	DEC	CHAR	ASCII	DEC	CHAR
0100000	32	SP	1000000	64	@	1100000	96	‘
0100001	33	!	1000001	65	A	1100001	97	a
0100010	34	“	1000010	66	B	1100010	98	b

0100011	35	#	1000011	67	C	1100011	99	C
0100100	36	$	1000100	68	D	1100100	100	d
0100101	37	%	1000101	69	E	1100101	101	e
0100110	38	&	1000110	70	F	1100110	102	f
0100111	39	`	1000111	71	G	1100111	103	g
0101000	40	(	1001000	72	H	1101000	104	h
0101001	41	)	1001001	73	I	1101001	105	i
0101010	42	*	1001010	74	J	1101010	106	j
0101011	43	+	1001011	75	K	1101011	107	k
0101100	44	,	1001100	76	L	1101100	108	l
0101101	45	-	1001101	77	M	1101101	109	m
0101110	46	.	1001110	78	N	1101110	110	n
0101111	47	/	1001111	79	O	1101111	111	o
0110000	48	0	1010000	80	P	1110000	112	p
0110001	49	1	1010001	81	Q	1110001	113	q
0110010	50	2	1010010	82	R	1110010	114	r
0110011	51	3	1010011	83	S	1110011	115	s
0110100	52	4	1010100	84	T	1110100	116	t
0110101	53	5	1010101	85	U	1110101	117	u
0110110	54	6	1010110	86	V	1110110	118	v
0110111	55	7	1010111	87	W	1110111	119	w
0111000	56	8	1011000	88	X	1111000	120	x
0111001	57	9	1011001	89	Y	1111001	121	y
0111010	58	:	1011010	90	Z	1111010	122	z
0111011	59	;	1011011	91	[	1111011	123	{
0111100	60	<	1011100	92	\	1111100	124	|
0111101	61	=	1011101	93	]	1111101	125	}
0111110	62	>	1011110	94	^	1111110	126	~
0111111	63	?	1011111	95	_	1111111	127	DEL

Dalam prakteknya, ketika kode ASCII dikirimkan, sebuah bit ditambahkan di bagian paling kiri atau di posisi MSB (*most significant bit*) kode ASCII untuk membuatnya kompatibel dengan sistem 8-bit. Kode ASCII 8-bit inilah yang dikirimkan melalui komunikasi serial Arduino. Pengiriman data dengan ukuran per byte atau 8 bit sekali kirim ini memiliki banyak

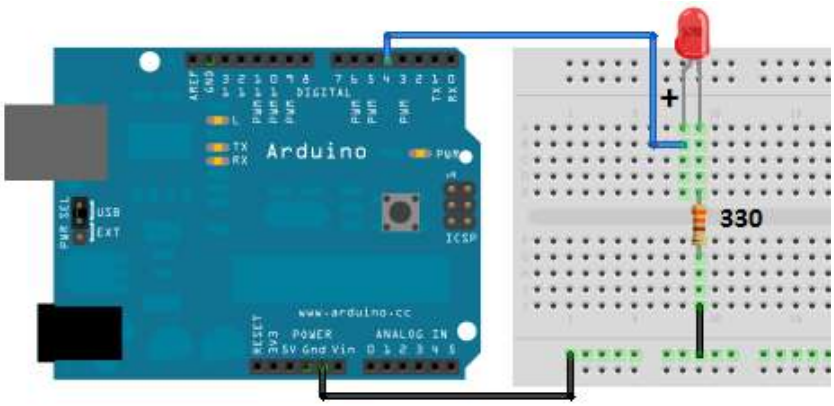
keuntungan. Sub bab 4.7 yang membahas tentang Konsep Byte akan menjelaskan keuntungan ini.

4.4 Contoh #2: Input Serial Output Digital

Pada contoh kedua ini akan dibuat aplikasi di mana sebuah LED yang terhubung dengan kaki D4 Arduino, akan menyala, bila angka 1 dikirimkan dari komputer ke Arduino, dan akan padam, bila selain angka 1 dikirimkan. Pengiriman angka ini dilakukan dengan mengetikkan sebuah angka di kolom **Send** jendela Serial Monitor, diikuti **Enter**.

Berikut langkah-langkah pembuatan aplikasi ini:

1. Susun rangkaian seperti gambar berikut ini:



Gambar 4.4 Rangkaian output digital LED terhubung dengan kaki D4

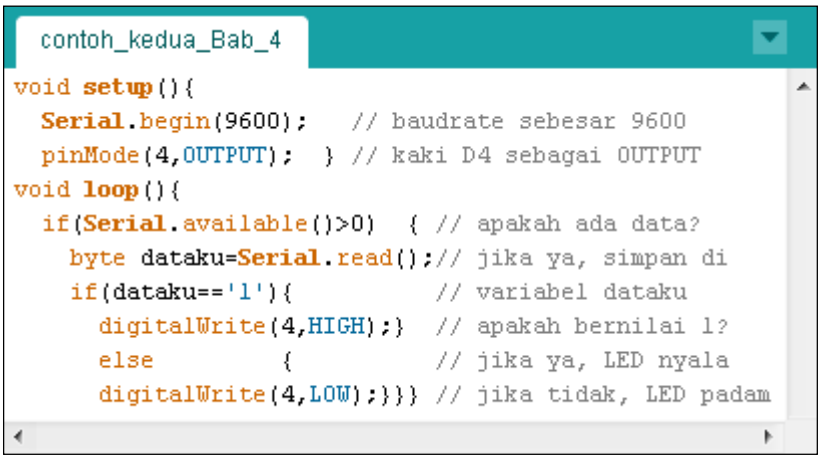
Keterangan rangkaian: Tanda + pada kaki LED menunjukkan kaki Anoda (kaki yang panjang), dihubungkan dengan kaki Arduino D4. Kaki Katoda LED diseri dengan resistor 330 Ω yang terhubung dengan Ground.

2. Hubungkan Arduino ke komputer dengan kabel USB.

3. Buka software Arduino, dan ketik program pada gambar 4.5.

4. Pilih **Board** dan **Serial Port** yang digunakan di menu **Tools**.

5. Tekan tombol **Verify** yang bergambar centang di pojok kiri Toolbar.



Gambar 4.5 Program contoh kedua

6. Setelah muncul pesan "*Done Compiling*", berikutnya tekan tombol **Upload** yang berada di sebelah kanan tombol **Verify**.

7. Setelah muncul pesan "*Done Uploading*", berikutnya buka Serial Monitor dengan meng-klik tombol bergambar kaca pembesar di pojok kanan Toolbar.

8. Karena baudrate pada program Arduino sama seperti di Serial Monitor, maka langkah selanjutnya adalah ketik angka 1 diikuti Enter, apakah LED dapat menyala? Lanjutkan dengan mengetik angka lain, apakah LED sekarang padam?

   ⇨ *Jawaban keduanya seharusnya adalah Ya.*

Hal-hal penting pada program:

Perhatikan bahwa angka 1 pada instruksi **if(dataku == '1')** diapit dengan tanda petik tunggal.

Dengan menambahkan tanda petik, maka angka 1 tersebut telah berubah menjadi karakter ASCII, yang sama dengan angka desimal sebesar 49 (lihat **Tabel 4.2**). Jadi pembaca juga bisa mengganti instruksi **if(dataku=='1')** menjadi **if(dataku==49)**.

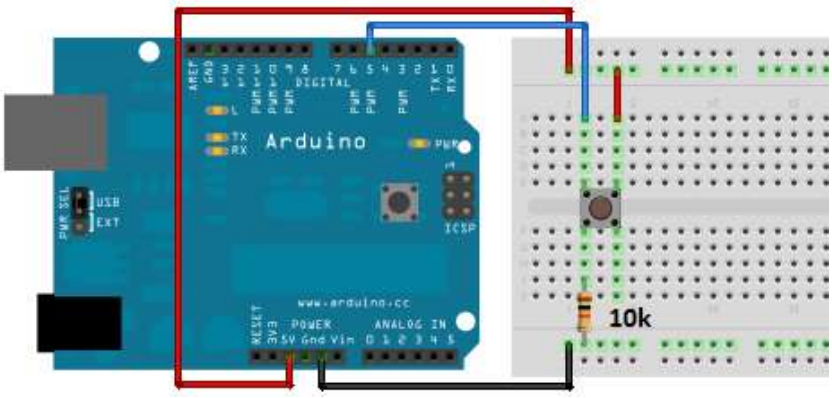
4.5 Contoh #3: Input Digital Output Serial

Aplikasi pada contoh ketiga ini merupakan kebalikan dari contoh kedua. Apabila pada contoh kedua komputer mengirimkan data ke Arduino, maka pada contoh ketiga ini, Arduino mengirimkan data ke komputer, di mana data yang dikirimkan nilainya akan berbeda tergantung pada kondisi tombol di kaki D5 Arduino, apakah ditekan atau dilepas.

Apabila tombol ditekan, maka angka 1 dikirimkan, dan apabila tombol dilepas, maka angka 0 dikirimkan. Data angka ini kemudian diterima oleh komputer dan ditampilkan di Serial Monitor.

Berikut langkah-langkah pembuatan aplikasi ini.

1. Susun rangkaian seperti gambar berikut ini:



Gambar 4.6 Rangkaian input digital tombol di kaki D5

Keterangan rangkaian: Tombol disusun seri dengan resistor 10 kΩ sebagai rangkaian pembagi tegangan 5V. Titik temu antara tombol dengan resistor 10 kΩ dihubungkan dengan kaki D5 Arduino.

2. Hubungkan Arduino ke komputer dengan kabel USB.
3. Buka software Arduino, dan ketik program pada gambar 4.7.
4. Pilih **Board** dan **Serial Port** yang digunakan di menu **Tools**.
5. Tekan tombol **Verify** bergambar centang di pojok kiri Toolbar.
6. Setelah muncul pesan "*Done Compiling*", berikutnya tekan tombol **Upload** yang berada di sebelah kanan tombol **Verify**.

contoh_ketiga_Bab_4

```
void setup(){
  Serial.begin(9600);   // baudrate sebesar 9600
  pinMode(5,INPUT);  }  // kaki D5 sebagai INPUT
void loop(){
  byte tombol=digitalRead(5); // baca tombol
  if(tombol==HIGH) {          // jika ditekan
    Serial.println('1');}     // kirim angka 1
  else  {                     // jika tidak,
    Serial.println('0');}}    // kirim angka 0
```

Gambar 4.7 Program contoh ketiga

7. Setelah muncul pesan "*Done Uploading*", berikutnya buka Serial Monitor dengan meng-klik tombol bergambar kaca pembesar di pojok kanan Toolbar. Apakah muncul angka 0 terus-menerus secara menurun? Kemudian tekan tombol yang terhubung dengan kaki D5 Arduino, apakah angka 0 berubah menjadi angka 1?

   ⇨ *Jawaban keduanya seharusnya adalah Ya.*

Hal-hal penting pada program:

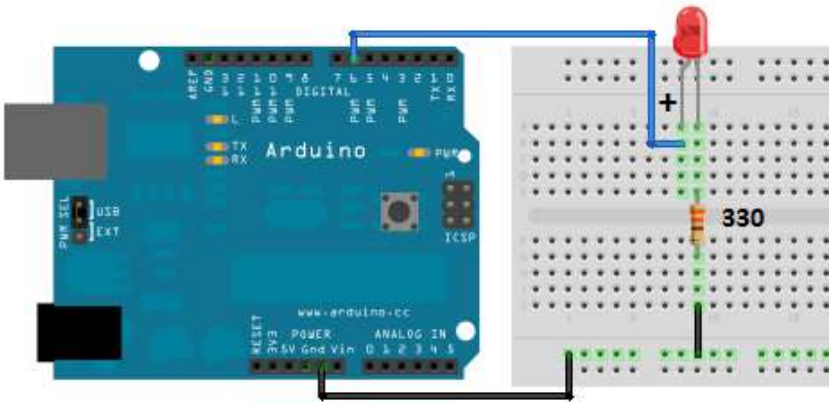
Instruksi **Serial.println('1')** di atas dapat diganti dengan **Serial.println(1)**, di mana angka 1 tidak perlu diberi tanda petik, karena instruksi **Serial.println** langsung membuat angka yang dikirimkan menjadi karakter ASCII untuk setiap digitnya, yang diikuti dengan karakter CR dan LF (karakter CR dan LF ini sama dengan Enter).

4.6 Contoh #4: Input Serial Output Analog

Apabila aplikasi pada contoh kedua dan ketiga menggunakan alat input output digital yang hanya memiliki 2 kondisi, maka pada contoh keempat ini akan digunakan alat output analog berupa LED yang dapat diatur intensitas cahayanya dari komputer, dengan nilai antara 0-255, di mana 0 untuk LED benar-benar padam dan 255 untuk LED benar-benar terang. LED ini terhubung dengan kaki PWM D6 Arduino.

Berikut langkah-langkah pembuatan aplikasi ini:

1. Susun rangkaian seperti gambar 4.8.



Gambar 4.8 Rangkaian output analog LED di kaki PWM D6

Keterangan rangkaian: Tanda + pada kaki LED menunjukkan kaki Anoda (kaki yang panjang), terhubung ke kaki PWM Arduino D6. Kaki Katoda LED diseri dengan resistor 330 Ω yang terhubung dengan Ground.

2. Hubungkan Arduino ke komputer dengan kabel USB.

3. Buka software Arduino, dan ketik program berikut.

```
contoh_keempat_Bab_4 §
void setup() {
  Serial.begin(9600);   // baudrate 9600
  pinMode(6, OUTPUT);}  // kaki 6 sebagai OUTPUT
void loop() {
  if (Serial.available()>0) {  // apakah ada data?
    byte terang=Serial.read(); // jika ya, simpan
    analogWrite(6, terang);}}  // keluarkan di D6
```

Gambar 4.9 Program contoh keempat

4. Pilih **Board** dan **Serial Port** yang digunakan di menu **Tools**.

5. Tekan tombol **Verify** bergambar centang di pojok kiri Toolbar.

6. Setelah muncul pesan "*Done Compiling*", berikutnya tekan tombol **Upload** yang berada di sebelah kanan tombol **Verify**.

7. Setelah muncul pesan "*Done Uploading*", berikutnya buka Serial Monitor dengan meng-klik tombol bergambar kaca pembesar.

8. Karena baudrate pada program Arduino sama seperti di Serial Monitor, maka langkah selanjutnya adalah ketik angka 0 pada kolom Send di Serial Monitor diikuti Enter. Kemudian ulangi ketik angka 255 diikuti Enter. Apakah LED padam ketika angka 0 dikirimkan, dan menyala terang ketika angka 255 dikirimkan?

   ⇨ *Jawabannya seharusnya adalah Tidak. Mengapa demikian?*

*Instruksi **Serial.read()** pada program di atas hanya membaca data 1 byte saja setiap kali pembacaan. Data 1 byte ini berarti 1 digit angka apabila dikirimkan melalui Serial Monitor. Jadi untuk pengiriman angka 255, maka ini dianggap 3 byte. Itulah sebabnya, LED tidak menyala terang ketika dikirimkan angka 255. Lebih jelasnya, baca Konsep Byte berikut ini.*

4.7 Konsep Byte

1 byte adalah 8 bit. Sekalipun sebenarnya jumlah data bisa diatur kurang dari 8 bit, namun secara default, pengiriman data melalui komunikasi serial adalah 8 bit data setiap kali kirim. Seperti terlihat pada seting default komunikasi serial: **9600, 8, n, 1,** yang artinya kecepatan baudrate sebesar 9600, 8 bit data, tanpa parity check (n) dan 1 bit stop.

Pengiriman data per 8 bit ini memiliki banyak keuntungan, yaitu:

1. Dengan 8 bit data ini, maka dapat diperoleh 256 buah kemungkinan data yang berbeda.
2. Dengan ditetapkannya kode ASCII, maka 8 bit data tersebut sama dengan sebuah karakter, sehingga menyederhanakan komunikasi. Artinya, untuk data yang berukuran 8 bit atau kurang, dapat dikirimkan sebagai satu karakter, sedangkan untuk data berukuran lebih dari 8 bit hingga 16 bit, dapat dikirimkan dalam 2 karakter, dan seterusnya.
3. Dengan sistem paket 8 bit data ini, memudahkan pemisahan variabel. Anggap bahwa ada 3 data variabel yang akan dikirimkan, di mana data variabel pertama memiliki tipe data Boolean yang berukuran 2 bit, data variabel kedua memiliki tipe data Byte yang berukuran 8 bit, dan data variabel ketiga memiliki tipe data Integer yang berukuran 16 bit. Maka pengiriman ketiga data tersebut dapat diurutkan dan dibentuk

menjadi 4 paket data. Paket data pertama untuk variabel Boolean, paket data kedua untuk variabel Byte, dan paket data ketiga dan keempat untuk variabel Integer.

Berikutnya, menjawab pertanyaan pada **Contoh Keempat**, di mana LED tidak menyala terang ketika angka 255 dikirimkan dari kolom **Send** di jendela Serial Monitor, dan tidak padam ketika dikirimkan angka 0, maka berikut ini penjelasannya:

Seperti telah dituliskan, bahwa instruksi **Serial.read()** pada program hanya membaca data 1 byte saja setiap kali pembacaan, di mana data yang diinginkan adalah nilai desimal antara 0-255, di mana nilai 0 untuk padam dan nilai 255 untuk menyala terang. Dalam format biner 8 bit, nilai desimal 0 ini sama dengan angka 0000 0000, dan nilai desimal 255 ini sama dengan angka 1111 1111.

Namun demikian, pengiriman data melalui jendela Serial Monitor tidak memberikan angka desimal, melainkan karakter ASCII. Jadi untuk angka 0, yang sebenarnya dikirimkan adalah 0011 0000 atau angka desimal 48. Kemudian untuk angka 255, yang dikirimkan adalah 0011 0010 0011 0101 0011 0101 atau sama dengan angka desimal 50 53 53.

Lalu bagaimana bila diinginkan pengiriman angka desimal 0 dari jendela Serial Monitor? Jawabannya adalah tidak bisa. Begitu pula tidak bisa mengirimkan angka desimal 255. Karena jendela Serial Monitor hanya bisa mengirimkan karakter yang tercetak saja (lihat Tabel 4.2). Jadi angka desimal yang paling rendah bisa dikirimkan adalah angka 33, yaitu dengan mengetikkan karakter tanda seru (**!**) di kolom Send diikuti Enter. Sedangkan angka desimal yang paling tinggi adalah angka 127, yaitu dengan mengetikkan karakter tak terhingga (**~**) di kolom Send diikuti Enter. Agar lebih paham, silahkan pembaca mengulangi langkah ke-8, tetapi sebagai ganti angka 0 dan 255, gunakan karakter di Tabel 4.2, yaitu

dimulai dengan mengetikkan karakter tanda seru (!), tanda petik dua (“), … dst. nya hingga karakter tanda tak terhingga (~), maka seharusnya LED akan menyala semakin terang.

Untungnya, keterbatasan pengiriman data seperti pada jendela Serial Monitor yang merupakan emulator (tiruan) komunikasi serial ini tidak dijumpai pada komunikasi serial yang sebenarnya, di mana untuk komunikasi serial yang sebenarnya, menggunakan kode mesin atau kode biner, sehingga bisa mencakup semua karakter, baik karakter yang tercetak maupun tidak tercetak, dari 0000 0000 hingga 1111 1111 untuk pengiriman data per 8 bit.

Instruksi pada Arduino untuk pengiriman data per byte atau per 8 bit ini adalah **Serial.write()**. Sekalipun bisa menggunakan instruksi **Serial.print()** ditambah dengan fungsi **char()**, namun tentu saja ukuran file hasil kompilasi dengan **Serial.write()** akan lebih kecil dan lebih efisien. Instruksi **Serial.print()** tanpa fungsi **char()** akan menghasilkan karakter ASCII untuk setiap digit datanya. Sedangkan instruksi **Serial.write()** akan menghasilkan satu saja karakter ASCII. Agar lebih paham, perhatikan contoh instruksi berikut ini dan hasilnya:

- **Serial.print(122)** → mengirimkan 3 byte kode ASCII, yaitu 0011 0001 0011 0010 0011 0010, atau 3 karakter, yaitu 122.
- **Serial.write(122)** → mengirimkan 1 byte kode ASCII, yaitu 0111 1010, atau karakter z.
- **Serial.print(char(122))** → mengirimkan 1 byte kode ASCII, yaitu 0111 1010, atau karakter z. Hasil instruksi ini sama dengan **Serial.write(122)**, hanya saja file hasil kompilasinya lebih besar.

Berikutnya adalah tentang tanda petik. Dengan menambahkan tanda petik pada data yang dikirimkan, akan membuat data tersebut menjadi karakter, yang akan ditampilkan apa adanya. Untuk data dalam bentuk 1

digit gunakan tanda petik tunggal, sedangkan lebih dari 1 digit, gunakan tanda petik ganda. Sebagai contoh, lihat instruksi berikut ini:

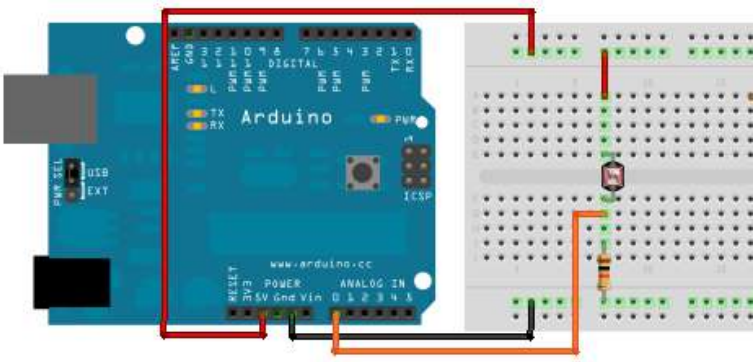
- **Serial.write("122")** → mengirimkan 3 byte kode ASCII, yaitu 0011 0001 0011 0010 0011 0010, atau 3 karakter, yaitu 122.

4.8 Contoh #5: Input Analog Output Serial

Apabila pada contoh keempat, data dikirimkan dari komputer ke Arduino, maka pada contoh kelima ini data dikirimkan dari Arduino ke komputer, di mana data yang dikirimkan akan bervariasi mengikuti besarnya cahaya yang diterima oleh LDR. LDR terhubung dengan kaki ADC Arduino A0, yang akan membaca sinyal analog dari LDR dengan nilai antara 0-1023 untuk tegangan 0-5V. Kemudian nilai angka tersebut dikirimkan ke komputer oleh Arduino dan ditampilkan di Serial Monitor.

Berikut langkah-langkah pembuatan aplikasi ini:

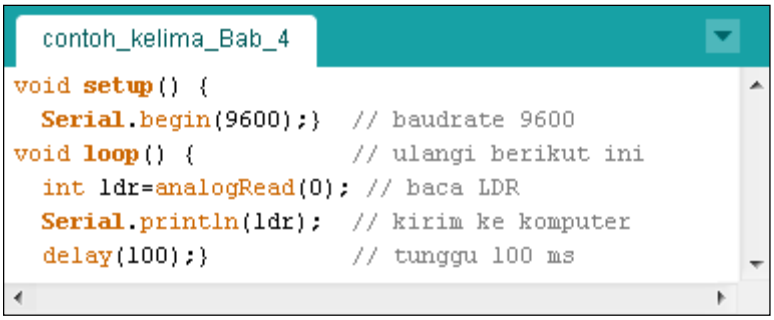
1. Susun rangkaian seperti gambar berikut.



Gambar 4.10 Rangkaian input analog LDR di kaki ADC A0

Keterangan rangkaian: LDR disusun seri dengan resistor 10 kΩ sebagai rangkaian pembagi tegangan 5V. Titik temu antara LDR dan resistor 10 kΩ dihubungkan dengan kaki ADC Arduino A0.

2. Hubungkan Arduino ke komputer dengan kabel USB.
3. Buka software Arduino, dan ketik program pada gambar 4.11.
4. Pilih **Board** dan **Serial Port** yang digunakan di menu **Tools**.



Gambar 4.11 Program contoh kelima

5. Tekan tombol **Verify** bergambar centang di pojok kiri Toolbar.
6. Setelah muncul pesan "*Done Compiling*", berikutnya tekan tombol **Upload** yang berada di sebelah kanan tombol **Verify**.
7. Setelah muncul pesan "*Done Uploading*", berikutnya buka Serial Monitor dengan meng-klik tombol bergambar kaca pembesar di pojok kanan Toolbar. Perhatikan pada Serial Monitor, apakah muncul nilai pada kolomnya? Tutuplah LDR dengan tangan dan perhatikan nilainya sekarang, apakah nilainya semakin besar ataukah semakin kecil?

   ⇨ *Jawaban keduanya seharusnya adalah Ya.*

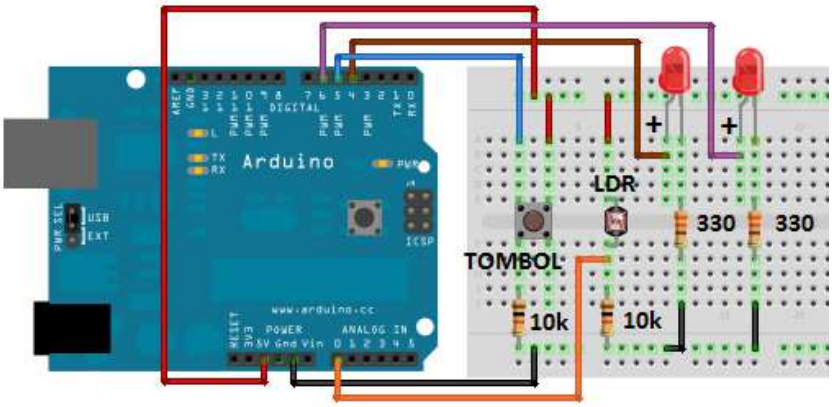
4.9 Contoh #6: Gabungan

Apabila contoh pertama hingga kelima, komunikasi dilakukan dalam satu arah saja, maka contoh keenam ini memperlihatkan bagaimana komunikasi Arduino dan komputer dilakukan dalam 2 arah, artinya

Arduino akan mengirimkan data ke komputer sekaligus menerima data dari komputer. Di samping itu, agar lebih lengkap, maka alat input output dari contoh-contoh sebelumnya akan digabungkan dalam aplikasi ini, baik tipe digital maupun analog.

Berikut langkah-langkah pembuatan aplikasi ini:

1. Susun rangkaian seperti gambar berikut.



Gambar 4.12 Rangkaian gabungan

Keterangan rangkaian: Tombol dan LDR, masing-masing disusun seri dengan resistor 10 kΩ sebagai rangkaian pembagi tegangan 5V. Titik temu antara tombol dan resistor 10 kΩ, dihubungkan dengan kaki D5, sedangkan titik temu LDR dan resistor 10 kΩ dihubungkan dengan kaki ADC Arduino A0. Untuk kaki Anoda LED kiri dihubungkan dengan kaki D4, sedangkan kaki Anoda LED kanan dihubungkan dengan kaki D6. Sedangkan kaki Katoda masing-masing LED dihubungkan dengan resistor pembatas arus 330 ohm yang terhubung dengan Ground.

2. Hubungkan Arduino ke komputer dengan kabel USB.

3. Buka software Arduino, dan ketik program seperti gambar 4.13.

4. Pilih **Board** dan **Serial Port** yang digunakan di menu **Tools**.
5. Tekan tombol **Verify** bergambar centang di pojok kiri Toolbar.
6. Setelah muncul pesan "*Done Compiling*", berikutnya tekan tombol **Upload** yang berada di sebelah kanan tombol **Verify**.

contoh_keenam_Bab_4

```
void setup() {
  Serial.begin(9600);  // baudrate 9600
  pinMode(5,INPUT);    // kaki D5 input tombol
  pinMode(4,OUTPUT);   // kaki D4 output LED
  pinMode(6,OUTPUT);}  // kaki D6 output LED
void loop() {
  if(Serial.available()) {
    int datakul=Serial.read();
    int dataku2=Serial.read();
    if (datakul=='1')
      {digitalWrite(4,HIGH);}
      else
      {digitalWrite(4,LOW);}
    analogWrite(6,dataku2);}
  int tombol=digitalRead(5); // baca tombol
  int ldr=analogRead(0);     // baca LDR
  Serial.print(tombol); // kirim nilai tombol
  Serial.print(',');    // beri koma
  Serial.println(ldr);  // kirim nilai LDR
  delay(100);}          // tunggu 100 ms
```

Gambar 4.13 Program contoh keenam

7. Setelah muncul pesan "*Done Uploading*", berikutnya buka Serial Monitor dengan meng-klik tombol bergambar kaca pembesar di pojok kanan Toolbar.

8. Perhatikan pada Serial Monitor, apakah muncul nilai pada kolomnya? Tekanlah tombol dan kemudian lepaskan tombol. Apakah nilai pada kolom untuk angka pertama sebelum koma, berubah dari 1 menjadi 0?

   ⇨ *Jawaban semuanya seharusnya adalah Ya.*

9. Kemudian tutuplah LDR dengan tangan dan kemudian buka secara perlahan. Apakah nilai pada kolom untuk angka setelah koma menunjukkan nilai yang semakin besar ketika LDR dibuka?

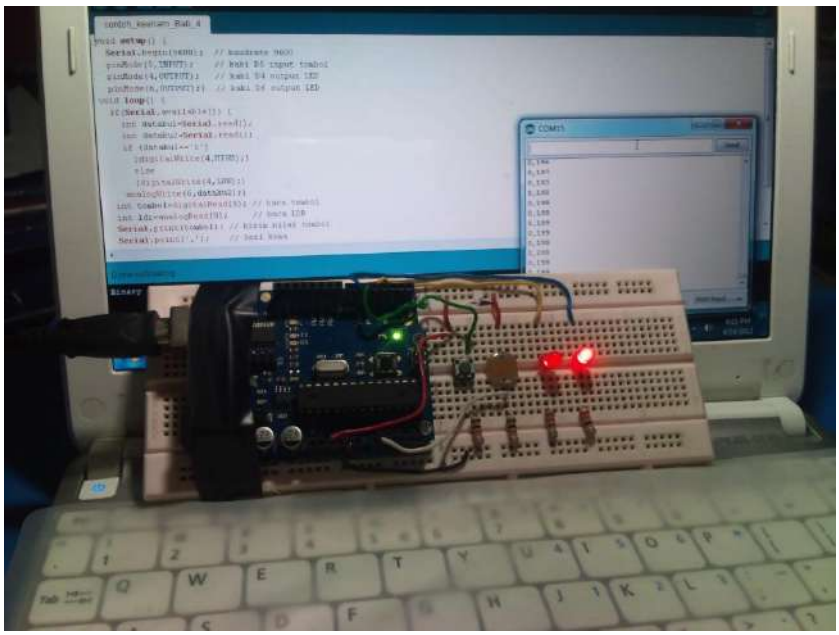
   ⇨ *Jawaban semuanya seharusnya adalah Ya.*

10. Ketik pada kolom Send angka 1 diikuti tanda seru (!), kemudian tekan Enter. Apakah LED pertama dan kedua menyala?

    ⇨ *Jawaban semuanya seharusnya adalah Ya.*

11. Ulangi dengan mengganti angka 1 dengan 0, dan tanda seru dengan tanda tak terhingga (~), dan kemudian tekan Enter. Apakah LED pertama padam dan LED kedua menyala semakin terang?

    ⇨ *Jawaban semuanya seharusnya adalah Ya.*



Gambar 4.14 Foto komunikasi serial Arduino dengan komputer

BAB 5
PENGENALAN LABVIEW

5.1 Apa itu LabVIEW?

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

Beberapa kelebihan LabVIEW dibandingkan dengan bahasa pemrograman lainnya adalah:

1. Bahasa pemrograman LabVIEW jelas dan mudah dipahami, karena berbentuk grafis, dengan instruksi berbentuk icon-icon, yang dihubungkan dengan garis/kawat untuk menunjukkan aliran data, mirip seperti flowchart.
2. Pembuatan program mudah, yaitu hanya dengan menarik keluar icon instruksi yang sudah tersedia di palet (kotak instruksi), dan menghubungkannya dengan kawat ke icon yang lain. Kawat ini sama seperti variabel pada bahasa pemrograman teks. Dengan cara ini

LabVIEW menyederhanakan pemrograman, karena kawat hanya akan terhubung apabila tipe datanya sesuai, sehingga menghilangkan kebutuhan manajemen memori dan deklarasi 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.

3. 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.
4. Dari tahun 1986 hingga sekarang, LabVIEW telah memiliki integrasi dengan ribuan hardware dan ratusan librari yang siap digunakan untuk aplikasi di bidang instrumentasi, pengolahan sinyal, analisis dan visualisasi data serta koneksi ke internet.
5. Telah terbukti di industri sebagai software yang handal, powerful, dan flexible, yang dapat digabungkan dengan librari eksternal dari software lain, seperti MATLAB, Simulink, SPICE, ADAMS, SolidWorks, Lego Mindstorms, dll.
6. Menjembatani dunia pendidikan dengan industri, karena software yang digunakan sama, sehingga transisi dan transfer teknologi dari dunia pendidikan ke industri menjadi lebih mudah.
7. LabVIEW didesain sebagai sebuah bahasa pemrograman paralel (multicore) yang mampu menangani beberapa instruksi sekaligus dalam waktu bersamaan. Hal ini sangat sulit dilakukan dalam bahasa pemrograman teks, karena biasanya bahasa pemrograman teks

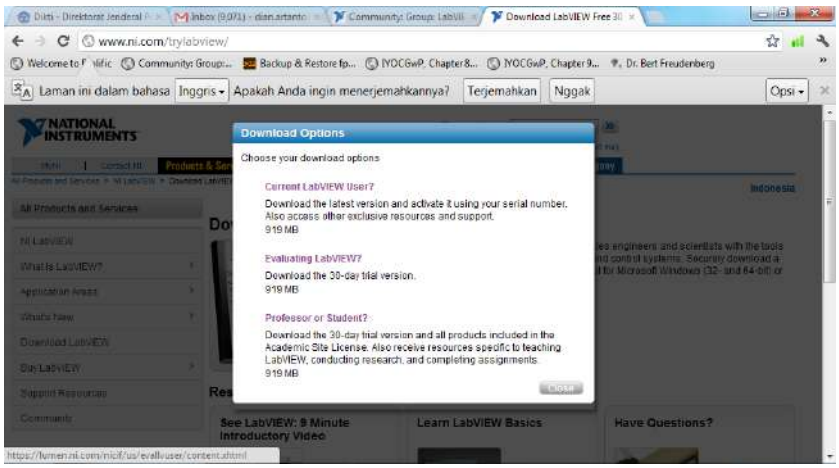
mengeksekusi instruksinya secara berurutan perbaris, satu demi satu. Dengan LabVIEW, pengguna dapat membuat aplikasi eksekusi paralel ini secara mudah dengan menempatkan beberapa struktur loop secara terpisah dalam block diagram.

8. Sifat modular LabVIEW memungkinkan pengguna untuk membuat program yang kompleks dan rumit menjadi sederhana, yaitu dengan cara membuat subprogram, atau di LabVIEW disebut subVI. Icon-icon di dalam LabVIEW sebenarnya merupakan subVI. Beberapa subVI dapat digabungkan menjadi sebuah subVI. SubVI-subVI gabungan tersebut dapat digabungkan lagi menjadi sebuah subVI lain, demikian seterusnya dengan tingkat hirarki yang tidak terbatas.
9. Satu alat untuk berbagai bidang, meliputi: otomotif, biomedis, komunikasi data, energi, kontrol, vision, dll, dengan penggunaan mulai dari perencanaan pengukuran, prototipe, pengujian, hingga implementasi dan pengembangannya.
10. Komunitas LabVIEW dan dukungan dari pihak National Instruments yang begitu besar untuk dunia pendidikan. Pembaca dapat mendownload secara gratis berbagai bahan pembelajaran di NI ZONE, yaitu di alamat: **zone.ni.com**.

5.2 Instalasi LabVIEW

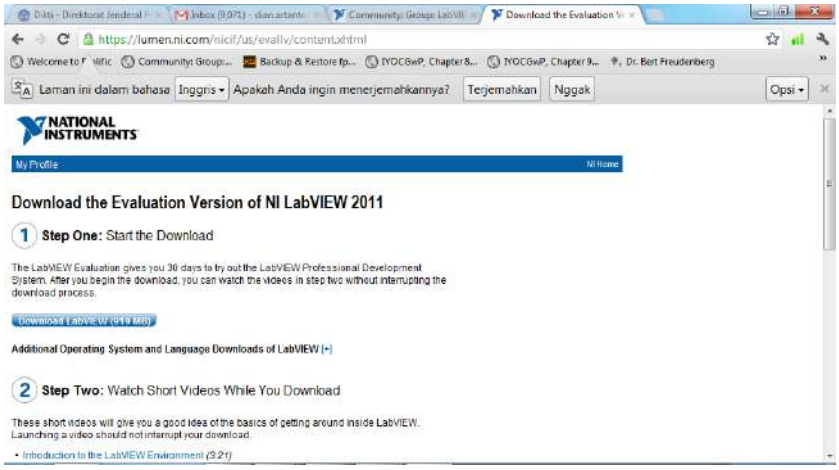
Berikut langkah-langkah instalasi LabVIEW:

1. Untuk bisa menginstal LabVIEW, pembaca perlu mendownload file LabVIEW di alamat **www.ni.com/trylabview/**, atau menggunakan bantuan mesin pencari Google dengan kata kunci “download LabVIEW”. Setelah halaman di alamat tersebut terbuka, klik pada tombol **See download options**, maka akan muncul 3 pilihan berikut:



Gambar 5.1 Tersedia 3 pilihan download

2. Pilih pilihan kedua, **Evaluating LabVIEW**, maka akan muncul halaman seperti berikut:



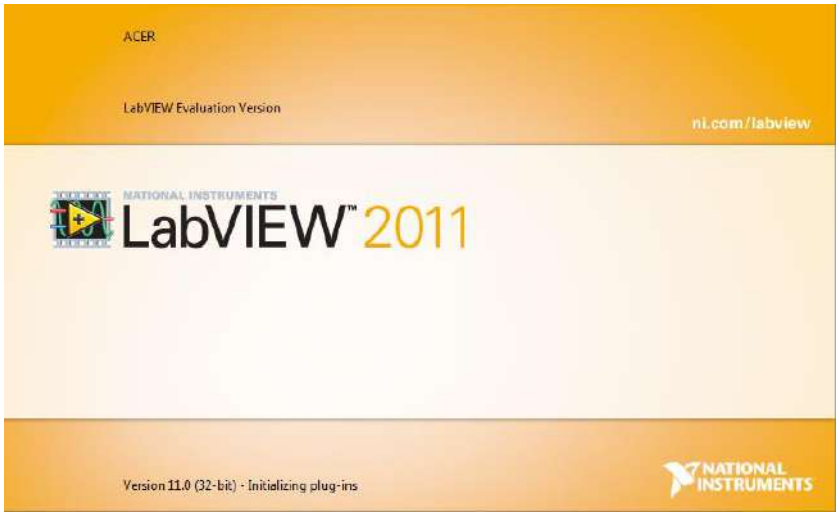
Gambar 5.2 Langkah-langkah download

3. Tekan tombol **Download LabVIEW**. Berhubung file yang didownload cukup besar, maka penekanan tombol tersebut tidak akan langsung mendownload file LabVIEW, tetapi akan mendownload sebuah file

aplikasi bernama **2011LV-WinEng_downloader** yang akan membantu melancarkan proses download. Setelah file tersebut terdownload, jalankan file tersebut dengan mengkliknya 2 kali, maka proses download file LabVIEW baru dimulai.

4. Setelah proses download file LabVIEW selesai, klik 2 kali pada file tersebut. Maka file tersebut akan dibuka dan ditempatkan di **C:\National Instruments Downloads\LabVIEW English\2011.**

5. Buka folder di atas, dan klik 2 kali pada file **Setup**. Maka proses instalasi dimulai. Ikuti proses instalasi hingga selesai. Maka LabVIEW versi evaluasi yang berlaku selama 30 hari siap digunakan.



Gambar 5.3 LabVIEW 2011 versi evaluasi

Catatan: sewaktu buku ini ditulis, versi yang terbaru adalah versi LabVIEW 2011. Ada kemungkinan sewaktu pembaca men-download, versi LabVIEW sudah tidak lagi 2011, karena secara berkala **National Instruments** selalu meng-upgrade LabVIEW ke versi yang lebih baru, dengan kemampuan yang lebih baik.

Sekalipun versi Evaluasi, namun kemampuannya sama dengan LabVIEW Profesional dan yang menyenangkan adalah versi ini gratis, hanya satu kekurangannya, yaitu waktu pemakaiannya hanya 30 hari.

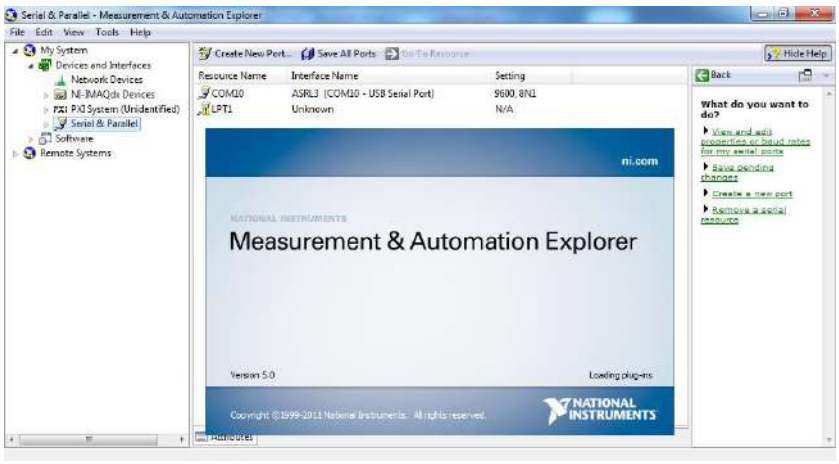
Tips: Untuk memperpanjang masa pemakaian versi evaluasi, pembaca dapat mengubah seting tanggal di komputer hingga mundur beberapa waktu ke belakang ke waktu di mana masa evaluasi masih berlaku. Kemudian jalankan LabVIEW. Setelah jendela **Getting Started LabVIEW** terbuka, pembaca dapat mengembalikan seting tanggal di komputer ke seting tanggal yang sekarang, dan LabVIEW dapat digunakan kembali sekalipun masa evaluasi telah habis.

Bantuan: apabila pembaca mengalami kesulitan dalam men-download file LabVIEW versi evaluasi, pembaca bisa menghubungi penulis melalui email di **dian.artanto@gmail.com**.

5.3 Instalasi Driver LabVIEW

Selanjutnya, agar LabVIEW bisa melakukan interaksi dengan Arduino, maka dibutuhkan instalasi driver **VISA** (*Virtual Instrument Software Architecture*), yang merupakan software aplikasi interface LabVIEW untuk komunikasi serial. Berikut langkah-langkah instalasi driver **VISA**:

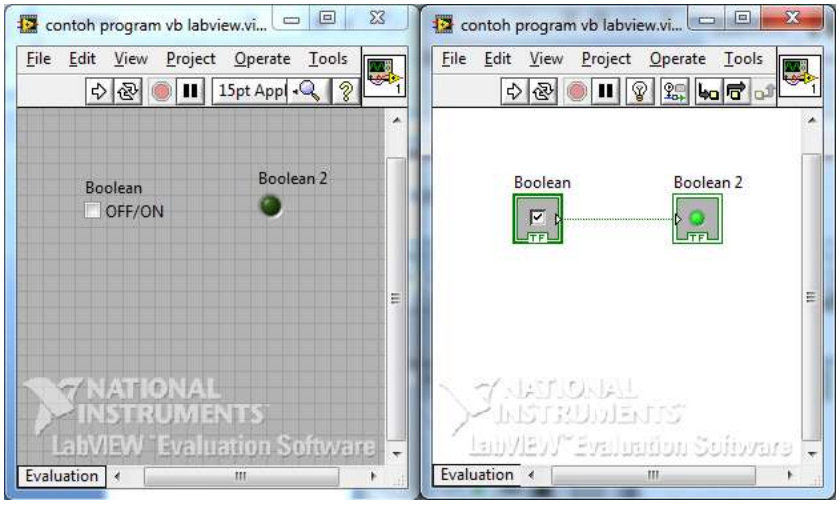
1. Klik 2 kali pada file bernama **visa503full** di dalam CD yang disertakan dalam buku ini, dan ikuti proses instalasi hingga selesai.
2. Apabila instalasi telah selesai, pembaca dapat melihat status driver untuk komunikasi serial tersebut dengan membuka software **MAX Measurement & Automation** (shortcut software ini ada di Desktop), dan perhatikan pada kolom kiri, di bagian bawah folder **Devices & Interfaces**, apabila di situ tertulis **Serial&Parallel**, maka instalasi driver **VISA** telah berhasil dilakukan.



Gambar 5.4 Tampak tulisan Serial&Parallel pada software MAX

5.4 Lingkungan Pemrograman LabVIEW

Lingkungan pemrograman LabVIEW terdiri dari 2 jendela, yaitu jendela Front Panel dan jendela Block Diagram, seperti gambar berikut ini.



Gambar 5.5 Jendela Front Panel dan jendela Block Diagram

Masing-masing jendela tersebut memiliki Toolbar dan Palet sendiri-sendiri. Berikut ini uraiannya:

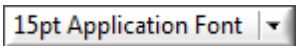
5.4.1 Toolbar Front Panel

Toolbar Front Panel tampak di bagian atas, di bawah Menu. Berikut beberapa tombol yang tersedia di Toolbar Front Panel:

Gambar 5.6 Tombol-tombol pada Toolbar Front Panel

Dengan fungsi masing-masing tombol sebagai berikut:

No	Tombol	Fungsi
1.		Klik tombol **Run** ini untuk menjalankan VI. Saat VI berjalan, tombol **Run** tersebut akan berubah menjadi seperti berikut: 
2.		Bila suatu ketika tombol **Run** terlihat patah seperti gambar di samping, maka hal itu 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:

No	Tombol	Fungsi
		
4.		Klik tombol **Abort Execution** ini untuk menghentikan VI.
5.		Klik tombol **Pause** ini untuk menghentikan VI sesaat.
6.		Klik tombol **Text Settings** ini untuk mengubah bentuk huruf, ukuran, perataan dan warna pada teks.
7.		Klik tombol **Align Objects** ini untuk menata posisi beberapa objek supaya rata, termasuk rata kanan, rata kiri, rata atas, dan rata bawah.
8.		Klik tombol **Distribute Objects** ini untuk menata posisi antar objek dengan spasi yang sama.
9.		Klik tombol **Resize Objects** ini untuk mem-besar/kecil-kan ukuran objek.
10.		Klik tombol **Reorder** ini untuk menempatkan objek di depan atau di belakang objek lain.

5.4.2 Toolbar Block diagram

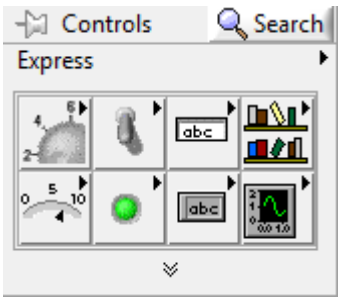
Toolbar di Block Diagram hampir sama dengan Toolbar di Front Panel, hanya sedikit berbeda, yaitu di Block Diagram tidak ada tool **Resize Objects** dan ada 6 tombol tambahan, yaitu:

Gambar 5.7 Tombol-tombol tambahan pada Toolbar Block Diagram

Berikut fungsi masing-masing tombol tersebut:

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

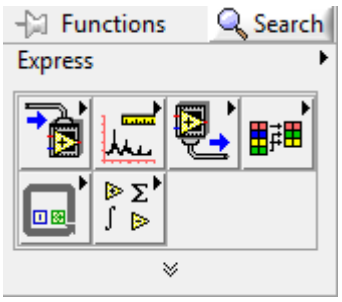
5.4.3 Palet Controls



Gambar 5.8 Palet Controls

Klik kanan pada jendela Front Panel, maka akan muncul **palet Controls**. **Palet Controls** ini hanya ada di jendela Front Panel. Palet ini digunakan untuk mengambil objek-objek yang akan terlihat oleh pengguna ketika program dijalankan. Pembaca dapat mencari objek dengan melihat pada kategori yang disediakan atau menggunakan bantuan tombol Search.

5.4.4 Palet Functions



Gambar 5.9 Palet Functions

Klik kanan pada jendela Block Diagram, maka akan muncul **palet Functions**. **Palet Functions** ini hanya ada di jendela Block diagram. Palet ini digunakan untuk mengambil icon-icon untuk pembuatan program di Block diagram. Tersedia tombol **Search** untuk membantu pencarian.

5.3.5 Palet Tool

Gambar 5.10 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. Tekan tombol **Shift** diikuti dengan **klik kanan** untuk memunculkan **palet Tool** ini.

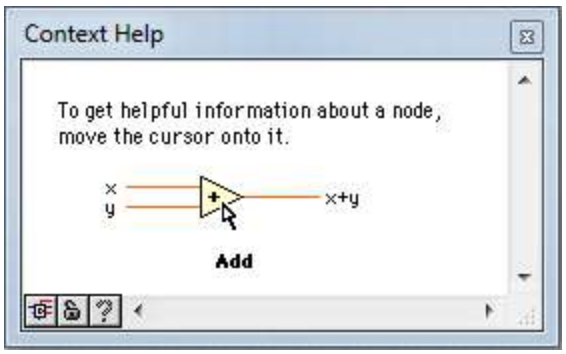
Isi palet Tool ini berhubungan dengan mode pada pointer mouse, apakah pointer mouse akan digunakan untuk menempatkan teks, memberi warna, memilih objek, menggulung layar, dll., masing-masing memiliki gambar pointer yang berbeda. Untuk lebih jelasnya, berikut keterangan fungsi masing-masing mode/tool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

No.	Tool	Fungsi
1.		Gunakan **Operating tool** ini untuk mengubah data dari sebuah kontrol atau teks.
2.		Gunakan **Positioning tool** ini untuk memilih, menggerakkan atau mengubah ukuran objek.
3.		Gunakan **Labeling tool** ini untuk mengedit teks dan menciptakan label.
4.		Gunakan **Wiring tool** ini untuk menghubungkan objek-objek pada Block diagram.

No.	Tool	Fungsi
5.		Gunakan **Objek Shortcut Menu tool** ini untuk mengakses sebuah menu shortcut objek dengan tombol kiri mouse.
6.		Gunakan **Scrolling tool** ini untuk menggulung layar.
7.		Gunakan **Breakpoint tool** ini untuk menempatkan breakpoint pada VI, fungsi, node, kawat dan struktur untuk menghentikan eksekusi pada lokasi tersebut.
8.		Gunakan **Probe tool** ini untuk mengukur nilai di suatu kawat pada Block diagram
9.		Gunakan **Color Copy tool** ini untuk meng-kopi warna dan menempelkan warnanya dengan Coloring tool.
10.		Gunakan **Coloring tool** untuk memberi warna ke sebuah objek.

Tool-tool di atas akan bekerja secara otomatis memilihkan mode yang tepat untuk pointer apabila LED indikator pada **palet Tool** menyala, seperti Gambar 5.10 di atas. Namun apabila pembaca menginginkan pengaturan tool pointer sendiri, pembaca dapat mematikan fungsi otomatis ini dengan meng-klik LED indikator **palet Tool** tersebut hingga padam. Pada kondisi ini, sekali sebuah tool pointer dipilih, tool tersebut tidak akan berubah, kecuali pembaca meng-klik pada pilihan tool yang lain di **palet Tool**.

5.3.6 Jendela Context Help



Gambar 5.11 Jendela Context Help

Tekan tombol tanda tanya di pojok kanan atas, maka jendela **Context Help** akan muncul. Jendela **Context Help** ini akan menampilkan informasi mengenai objek yang disentuh oleh pointer mouse.

BAB 6
PEMROGRAMAN LABVIEW

6.1 Istilah-istilah penting

Sebelum membahas mengenai pemrograman LabVIEW, sebaiknya pembaca mengenal istilah-istilah penting berikut ini:

1. **G:** dari kata "graphical", merupakan sebutan untuk bahasa pemrograman LabVIEW yang menggunakan kode berbentuk grafis.
2. **VI:** dari kata "virtual instruments", merupakan sebutan untuk program yang dibuat dengan LabVIEW.
3. **SubVI:** 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 icon, atau kotak kecil dengan gambar yang unik di dalamnya, dengan kaki input berada di sebelah kiri dan kaki output berada di sebelah kanan.
4. **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 representasi icon-nya di Block diagram, khususnya untuk objek-objek yang membawa

data, baik data yang masuk dari pengguna ke program, maupun data yang keluar dari program ke pengguna.

5. **Block Diagram:** adalah tempat pembuatan program, Jendela ini tidak akan terlihat oleh pengguna saat program dijalankan. Pembuatan program di sini dilakukan dengan cara menempatkan beberapa node dan menghubungkannya satu sama lain.
6. **Node:** adalah semua objek di jendela Block Diagram, yang memiliki input/output dan melakukan operasi tertentu ketika dijalankan, termasuk di dalamnya subVI, terminal, struktur dan fungsi.
7. **Terminal:** adalah icon-icon di Block diagram yang mewakili objek-objek di Front Panel, di mana objek-objek tersebut membawa data, baik data yang masuk dari pengguna ke program, maupun data yang keluar dari program ke pengguna. Contoh Terminal adalah Control dan Indicator.
8. **Control:** adalah semua objek di Front Panel yang memasukkan data dari pengguna ke program. Di sebut juga Terminal Input. Contoh Control adalah knob, tombol, saklar, dan alat input lainnya.
9. **Indicator:** adalah semua objek di Front Panel yang mengeluarkan atau menampilkan data dari program ke pengguna. Disebut juga Terminal Output. Contoh Indicator adalah grafik, LED, dll.
10. **Struktur:** adalah semua bentuk alur pemrograman, seperti perulangan, percabangan, urutan, dll. Contoh struktur ini adalah For Loop, While Loop, Sekuensial, Case, dll. Struktur ini hanya ada di jendela Block diagram, berbentuk blok yang dapat diatur luasannya, dan hanya bekerja untuk icon-icon yang ditempatkan di dalamnya.
11. **Fungsi:** adalah semua kode-kode dasar yang telah disediakan untuk membuat subVI. Contoh fungsi seperti Add, Subtract, dll.

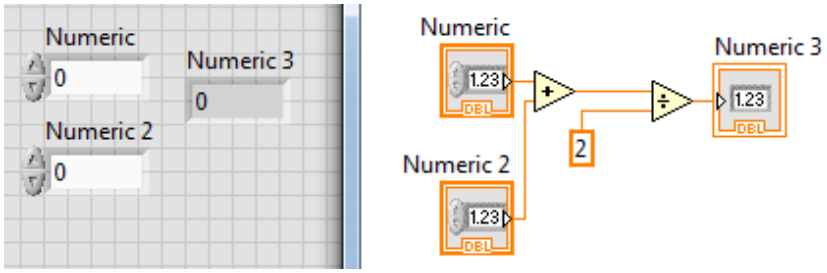
12. **Wire:** atau kawat, digunakan untuk menghubungkan icon-icon, sekaligus untuk menunjukkan aliran data dan tipe data. Beberapa warna kawat dan tipe datanya dapat dilihat dalam tabel berikut:

Tipe data	Skalar	Array 1D	Array 2D	Warna kawat
Numerik				Oranye/biru *)
Boolean				Hijau
String				Merah muda

Keterangan *)**:** warna kawat oranye untuk tipe data bilangan riil (*float*), sedangkan warna kawat biru untuk tipe data bilangan bulat (integer).

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

Untuk lebih jelas mengenai konsep pemrograman **Dataflow** ini, perhatikan program mencari rata-rata berikut ini, yang akan menjumlahkan 2 angka dan kemudian membaginya dengan 2. Dalam hal ini, program dieksekusi dari kiri ke kanan, bukan karena objek-objek tersebut ditempatkan dalam urutan demikian, namun karena fungsi **Divide (÷)** tidak akan bekerja sebelum fungsi **Add (+)** dieksekusi.

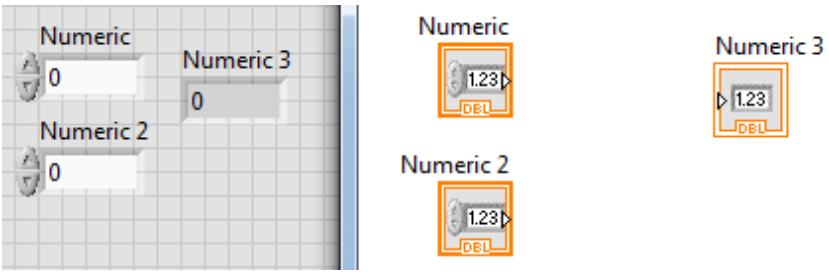


Gambar 6.1 Konsep Dataflow pada program mencari rata-rata

6.2 Membuat SubVI

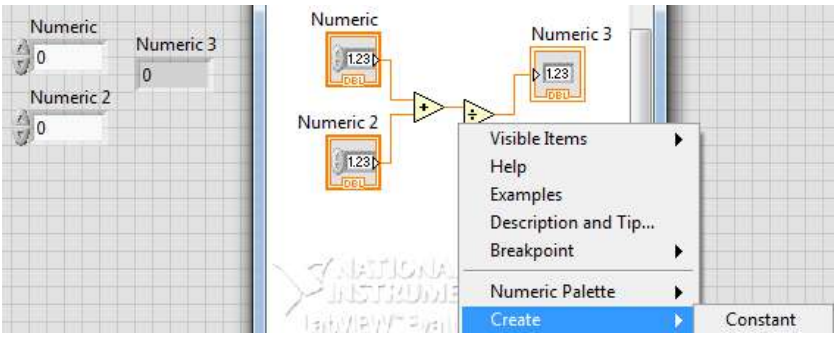
Berikut ini langkah-langkah untuk membuat sebuah subVI:

1. Buatlah program mencari rata-rata seperti gambar 6.1 diatas, yaitu dengan menempatkan 2 objek Num Ctrl dan sebuah objek Num Ind pada jendela Front Panel. Secara otomatis ketiga objek tersebut memunculkan 3 icon Terminal di Block diagram.



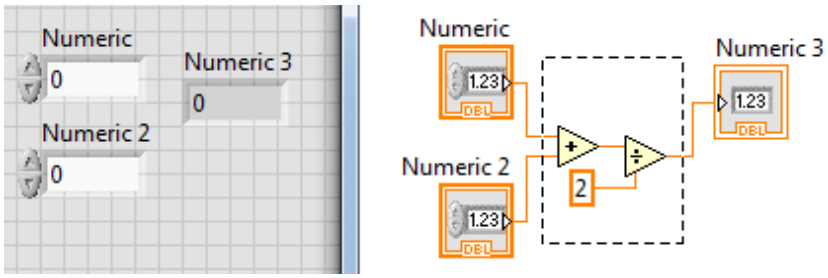
Gambar 6.2 Menempatkan 2 objek Num Ctrl dan sebuah objek Num Ind

2. Kemudian pada jendela Block diagram, tambahkan fungsi **Add**, **Divide** dan konstanta angka 2. Untuk memunculkan angka 2 ini, dapat dilakukan dengan menempatkan pointer mouse pada salah satu kaki input **Divide**, dan kemudian klik kanan untuk memunculkan menu popup, kemudian pilih **Create Constant**.



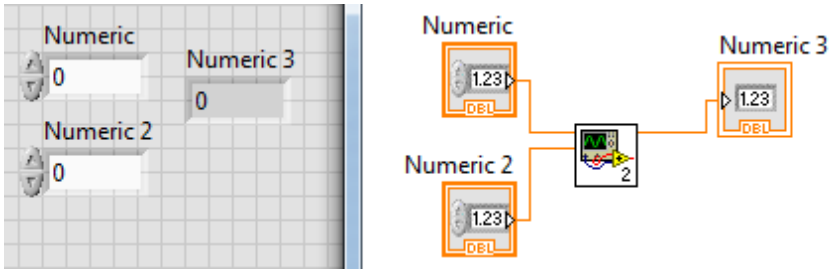
Gambar 6.3 Memunculkan konstanta (angka 2) dari menu popup

3. Setelah program selesai disusun seperti gambar 6.1, maka pada jendela Block diagram, klik dan gerakkan pointer mouse hingga melingkupi fungsi **Add**, **Divide** dan konstanta angka 2.



Gambar 6.4 Klik dan gerakkan mouse melingkupi fungsi dan konstanta

4. Pada menu **Edit** di jendela Block diagram, pilih **Create SubVI**, maka fungsi dan konstanta tersebut berubah menjadi sebuah icon (subVI).

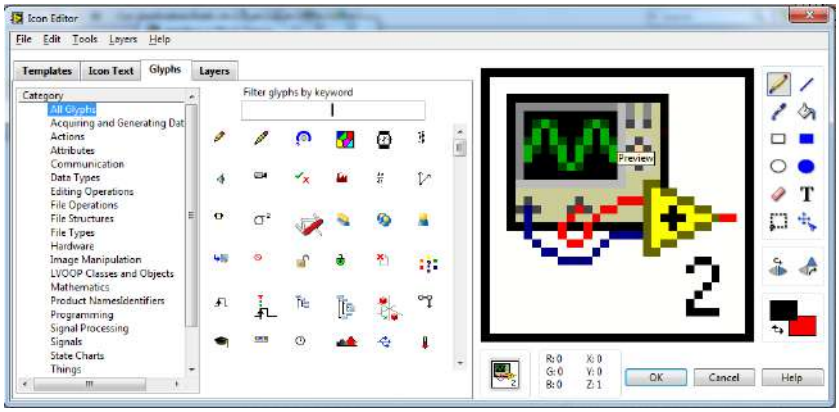


Gambar 6.5 Create subVI maka fungsi-fungsi menjadi sebuah icon

5. Klik 2 kali pada icon atau subVI tersebut, maka akan terbuka jendela Front Panel dari subVI tersebut.

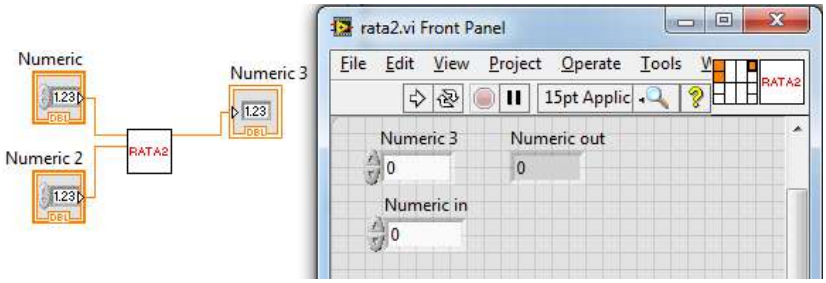
Gambar 6.6 Klik 2 kali subVI, maka muncul jendela Front Panel

6. Perhatikan pojok kanan atas jendela Front Panel tersebut, tampak kotak **Connector Pane** dan kotak **Icon subVI**. Kotak **Connector Pane** tersebut menunjukkan ada 2 buah kaki input di bagian kiri dan sebuah kaki output di bagian kanan subVI. Sedangkan kotak Icon menunjukkan tampilan subVI. Pembaca dapat mengubah tampilan subVI tersebut dengan meng-klik 2 kali pada kotak Icon, hingga muncul jendela Icon Editor seperti berikut ini.



Gambar 6.7 Jendela Icon Editor untuk mengubah tampilan subVI

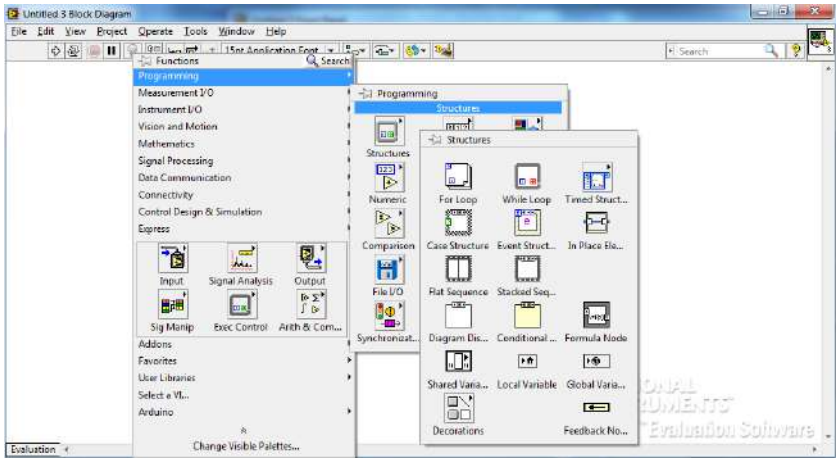
7. Ubah tampilan Icon dengan gambar yang unik dan kemudian simpan. Maka pembuatan subVI telah selesai.



Gambar 6.8 Tampilan subVI telah diubah

6.3 Struktur Pemrograman

LabVIEW menyediakan bermacam-macam struktur pemrograman. Untuk mengambilnya, klik kanan jendela Block diagram hingga muncul palet **Functions**, kemudian pilih kategori **Programming**, dilanjutkan dengan **Structures**, maka akan terlihat bermacam-macam struktur pemrograman, termasuk di dalamnya While Loop, For Loop, Shift Register, Case Structure, dll.



Gambar 6.9 Kategori Structures di palet Functions

6.3.1 While Loop

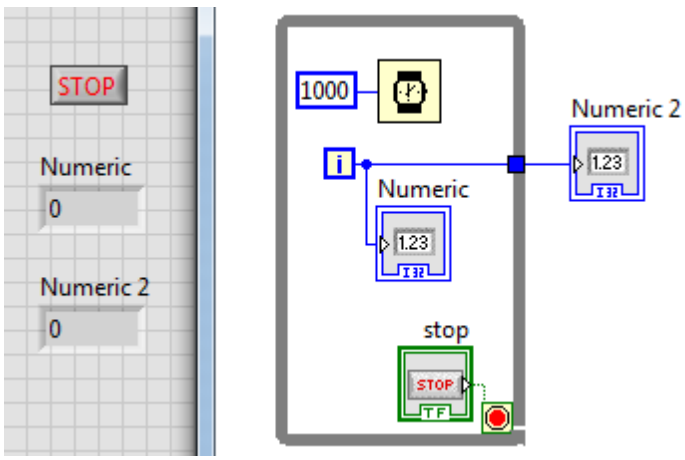
Struktur **While Loop** memiliki 3 komponen utama, yaitu sebuah blok yang dapat diatur luasannya, sebuah terminal input **conditional** (**Stop If True**) dan sebuah terminal output **counter** (**i**).

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

Terminal input **conditional** tersebut dapat diubah 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 mendapat nilai **True**, dan berhenti ketika mendapat nilai **False**.

Terminal output **counter (i)** akan memberikan nilai jumlah perulangan yang telah selesai dilakukan, yang dimulai dari 0.

Berikut contoh program dengan struktur **While Loop**. Tampak dalam gambar 2 buah objek Numeric Indicator, yaitu **Numeric** dan **Numeric2** sama-sama mendapat nilai **counter (i)**, hanya saja **Numeric** berada di dalam **While Loop**, sedangkan **Numeric2** berada di luar **While Loop**. Ketika program dijalankan, maka nilai **Numeric** akan bertambah mengikuti nilai **counter (i)**, sedangkan nilai **Numeric2** sama sekali tidak bertambah, karena nilai counter (i) tidak akan diteruskan ke Numeric2 sebelum **While Loop** berhenti. Ketika tombol Stop ditekan, maka **While Loop** berhenti, barulah nilai **counter (i)** diteruskan ke **Numeric2**.



Gambar 6.10 Numeric2 tidak akan berubah hingga While Loop selesai

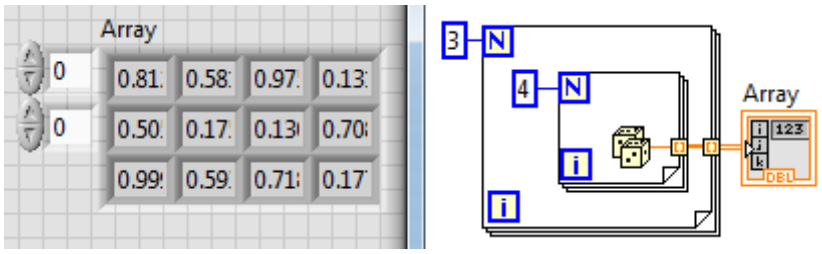
6.3.2 For Loop

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 icon-icon di dalam bloknya berulang-kali, dengan jumlah perulangan sebanyak nilai yang dimasukkan ke terminal input **N**. Apabila nilai yang dimasukkan ke terminal input **N** sebesar 0, maka **For Loop** tidak akan melakukan perulangan. Begitu pula apabila tidak ada nilai yang dimasukkan ke terminal input **N**, maka **For Loop** juga tidak akan melakukan perulangan, kecuali ada sebuah data array yang dimasukkan melalui dinding 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 meng-klik kanan dinding blok **For Loop**, dan pada menu popup yang muncul, pilih **Conditional Terminal**.

Berikut contoh program **For Loop** untuk membuat sebuah **Array 2D** atau disebut juga **Matriks** dengan ukuran **3x4** (3 baris 4 kolom) dan dengan nilai elemen-elemennya berupa angka **random**.



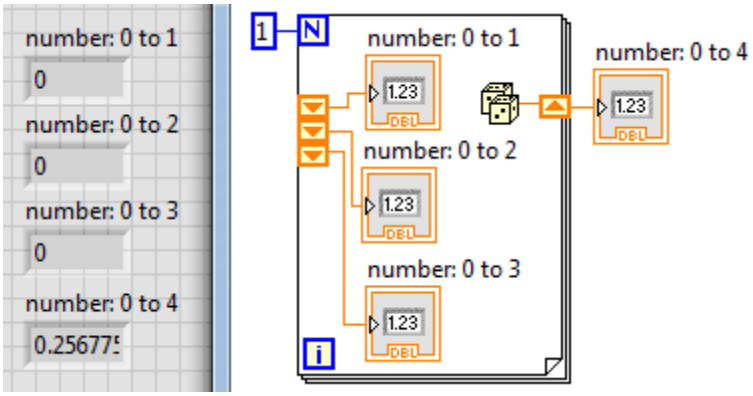
Gambar 6.11 Membuat Matriks 3x4 dengan For Loop

6.3.3 Shift Register

Struktur **Shift Register** ini tidak tersedia di palet **Functions**, karena penggunaannya hanya bisa diterapkan pada struktur **For Loop** dan **While Loop** saja. Untuk menggunakan struktur **Shift Register** ini, klik kanan dinding blok **For Loop** atau **While Loop**, dan kemudian pilih **Add Shift Register** dari menu popup yang muncul.

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, bergambar tanda panah ke bawah dan ke atas.

Untuk mengetahui secara jelas cara kerja **Shift Register** ini, tambahkan **Shift Register** untuk meneruskan data angka **random** pada blok **For Loop**, dengan jumlah perulangan N sebanyak 1. Tarik ke bawah terminal **Shift Register** di dinding kiri hingga muncul tambahan 2 terminal. Tambahkan **Indicator** untuk setiap terminal **Shift Register** tersebut, baik di dinding kiri maupun di kanan, dengan cara meng-klik kanan ujung terminal, sehingga muncul menu popup, dan pilih **Create Indicator**. Kemudian tekan tombol Run, dan perhatikan bagaimana data digeser.



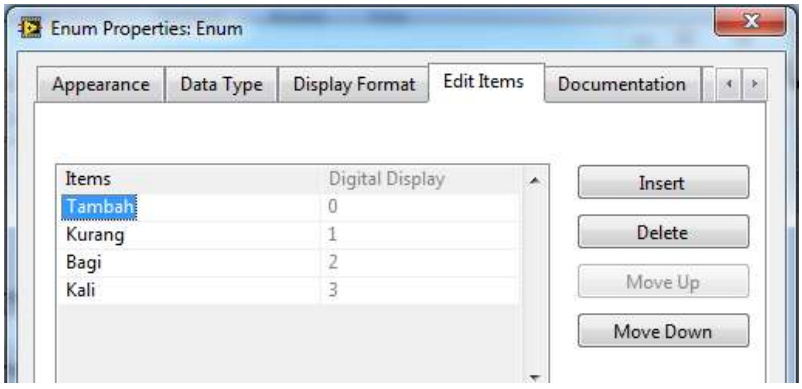
Gambar 6.12 Cara kerja Shift Register menggeser data

6.3.4 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.

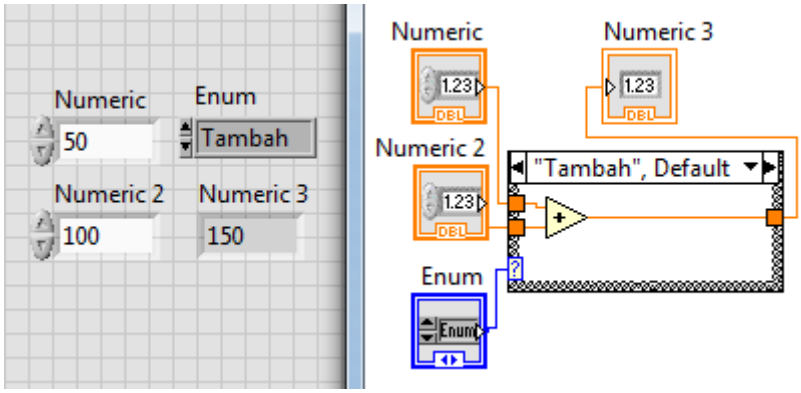
Blok mana yang akan dieksekusi, tergantung pada nilai input di terminal **selector (?)**. Di bagian atas setiap blok terdapat label yang menunjukkan nilai input yang mengaktifkannya. Tipe data nilai input ini bisa berupa Boolean (True atau False), Integer, String atau Enumerasi. Enumerasi adalah tipe data yang terdiri dari sekumpulan nilai.

Berikut ini diperlihatkan contoh struktur **Case** menggunakan data Enumerasi, untuk membuat kalkulator sederhana dengan 2 masukan. Objek Enumerasi ini dapat diambil di palet Controls di Front Panel, yaitu di kategori **Classic. Ring & Enum**. Pembaca juga dapat menemukan objek Enum tersebut dengan bantuan tombol **Search**. Setelah diperoleh, tempatkan objek tersebut di **Front Panel**, dan kemudian klik-kanan hingga muncul menu popup dan pilih **Edit Items..**. Setelah itu di kolom tabel pada jendela yang muncul, berturut-turut ketik kata “Tambah”, “Kurang”, “Bagi” dan “Kali” untuk nilai 0, 1, 2 dan 3.



Gambar 6.13 Data Enumerasi dengan nama Tambah, Kurang, Bagi, Kali

Kemudian pada jendela Block Diagram, hubungkan **icon Enum** tersebut dengan terminal **selector (?)** struktur **Case**, maka secara otomatis, label pada bagian atas blok struktur **Case** akan berubah dari True dan False, menjadi “Tambah”, dan “Kurang”. Klik kanan pada dinding blok struktur **Case** tersebut dan pilih **Add Case After** pada menu popup yang muncul, maka akan muncul sebuah blok yang berlabelkan “Bagi”. Ulangi lagi untuk memunculkan blok yang berlabelkan “Kali”.

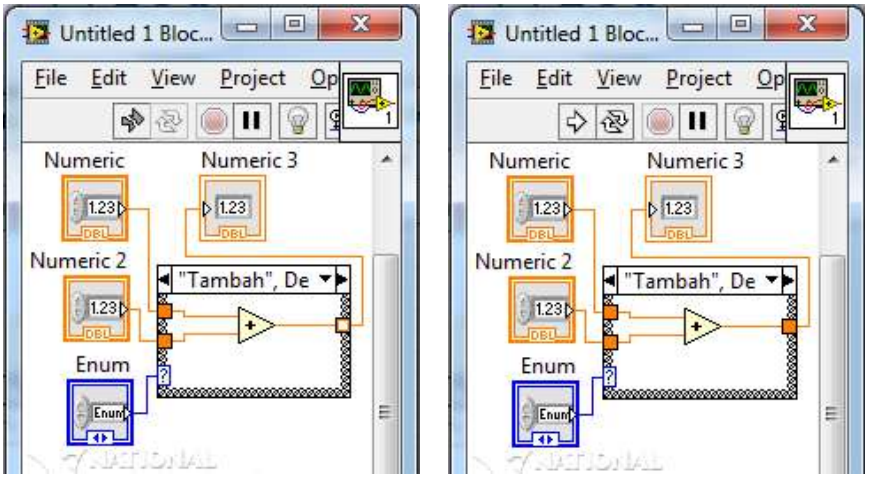


Gambar 6.14 Struktur Case dengan data Enum sebagai selectornya

Perhatikan bahwa pada blok berlabel “Tambah” diikuti dengan kata **Default**. Blok **Default** ini digunakan untuk menyediakan pilihan blok apabila tidak ada input yang sesuai. Jadi apabila input pada terminal **selector** struktur **Case** di luar keempat data Enum tersebut, maka struktuf **Case** akan mengaktifkan blok berlabel **Default** ini. Pembaca dapat memindahkan blok **Default** ini ke blok yang lain, dengan memilih **Make This The Default Case** pada blok yang akan dijadikan blok **Default**.

Data dapat dimasukkan ke satu atau beberapa blok di struktur Case. Data juga dapat dikeluarkan dari blok-blok di struktur Case, hanya saja, untuk data yang dikeluarkan, semua blok pada struktur **Case** harus terhubung dengan garis data tersebut. Apabila ada blok tidak terhubung, maka kotak terminal di dinding tempat garis data tersebut diteruskan akan berwarna

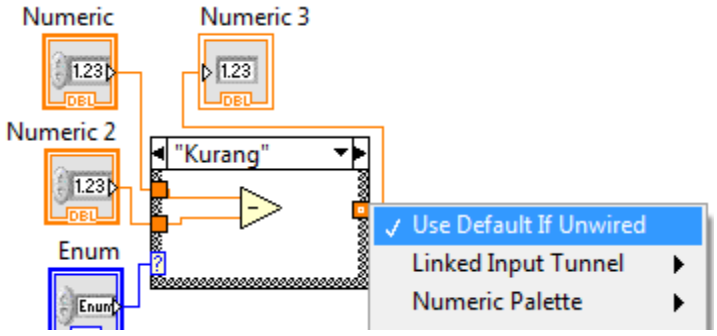
putih. Sedangkan apabila semua terhubung, maka kotak tersebut berwarna penuh, sesuai warna tipe datanya.



(a) (b)

Gambar 6.15 (a) Tidak semua blok terhubung, (b) Semua blok terhubung

Pembaca dapat meng-klik kanan kotak terminal dan memilih *Use Default If Unwired* pada menu popup yang muncul, untuk membuat blok-blok yang tidak terhubung secara otomatis memberikan nilai default (nilai 0).

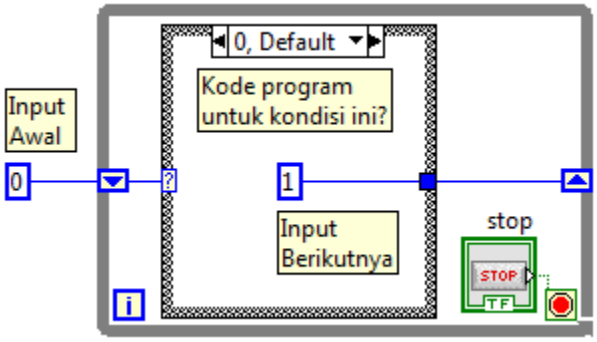


Gambar 6.16 Use Default If Unwired

6.4 State Machine

State Machine adalah sebuah cara pemrograman yang memungkinkan 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.

Di LabVIEW, cara pemrograman State Machine dapat diimplementasikan secara mudah, yaitu hanya dengan menggabungkan 3 buah struktur: While Loop, Case Structure dan Shift Register.



Gambar 6.17 Diagram dasar sebuah State Machine

Sebagai gambaran mengenai manfaat State Machine ini, perhatikan contoh aplikasi berikut ini:

Diinginkan sebuah simulasi kalkulator sederhana, yang memiliki tombol angka 0 sampai 9, dengan operasi Tambah (+), Kurang (-), Kali (x), dan Bagi (/). Cara kerja kalkulator ini adalah sebagai berikut:

1. Mula-mula pengguna memasukkan angka pertama, dengan menekan salah satu tombol angka (0-9). Kalkulator akan menampilkan angka pertama tersebut.

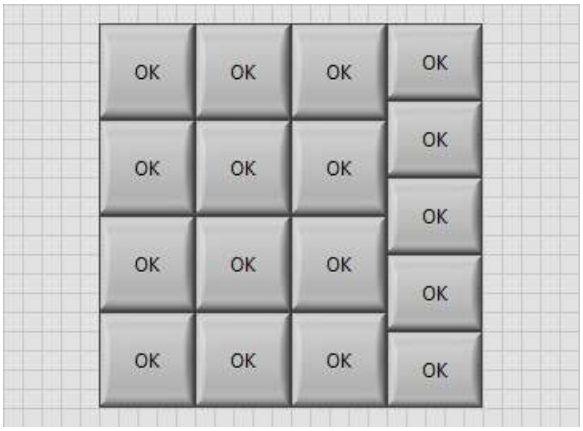
2. Kemudian pengguna menekan salah satu tombol operator perhitungan (+,-,x,/), Tampilan kalkulator tidak berubah, tetap menampilkan angka pertama.
3. Kemudian pengguna memasukkan angka kedua, dengan menekan salah satu tombol angka. Kalkulator menampilkan angka kedua ini.
4. Kemudian pengguna menekan tombol sama dengan (=), maka kalkulator akan menghitung dan menampilkan hasilnya.

Dari cara kerja di atas, dapat ditentukan komponen dari State Machine:

- **Keadaan**: tampilan kalkulator setiap tahapnya.
- **Input pemicu**: semua tombol

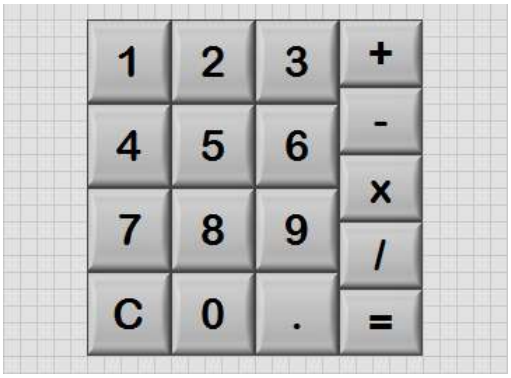
Berikut ini langkah-langkah pembuatan simulasi kalkulator sederhana menggunakan konsep State Machine:

1. Ambil tombol OK Button dari palet **Controls** sebanyak 17 buah, dan tempatkan pada jendela Front Panel dan susun seperti gambar berikut. Hilangkan label setiap tombol dengan meng-klik kanan tombol, pilih **Visible Items**, dan hilangkan centang pada **Label**.



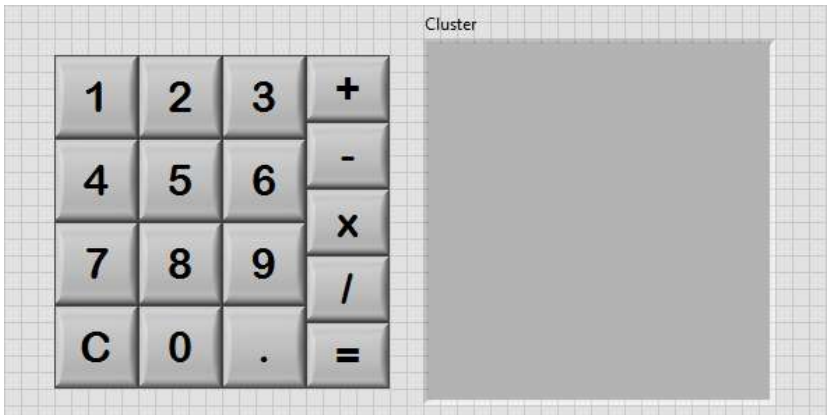
Gambar 6.18 Menempatkan 17 buah tombol OK Button

2. Ganti tulisan OK pada setiap tombol dengan angka dan tanda operator seperti gambar berikut ini:



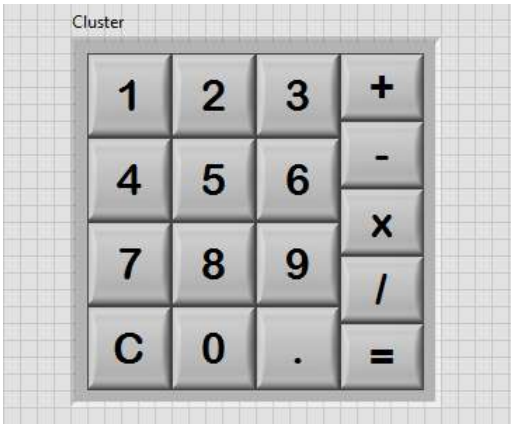
Gambar 6.19 Mengganti tulisan OK dengan angka dan tanda operator

3. Ambil objek Cluster dari palet Controls di kategori **Array, Matrix & Cluster**. Tempatkan pada Front Panel dan perbesar ukurannya.



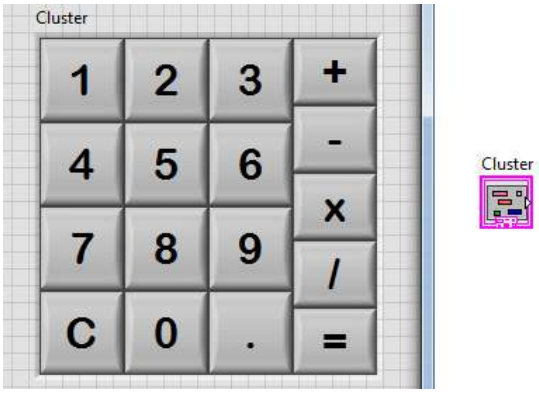
Gambar 6.20 Mengambil Cluster dan memperbesar ukurannya

4. Setelah ukuran Cluster cukup besar untuk memuat semua tombol, maka berikutnya pindahkan semua tombol ke dalam Cluster.



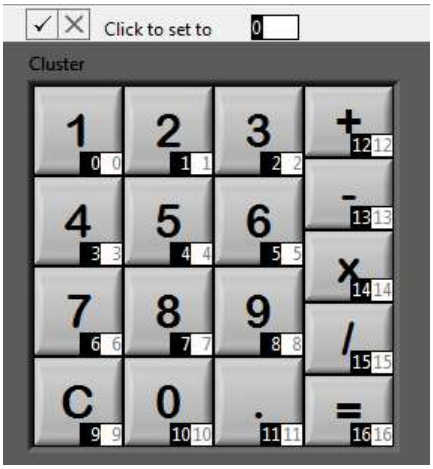
Gambar 6.21 Menempatkan semua tombol ke dalam Cluster

5. Klik kanan pada Cluster, dan pilih **AutoSizing** dan kemudian **Size to Fit** untuk menyesuaikan ukuran Cluster hingga tepat sama dengan blok tombol. Hilangkan pula Label Cluster. Perhatikan jendela Block Diagram, tampak bahwa semua icon tombol dapat digabungkan hingga menjadi sebuah icon saja, yaitu icon Cluster.



Gambar 6.22 Menempatkan semua tombol ke dalam Cluster

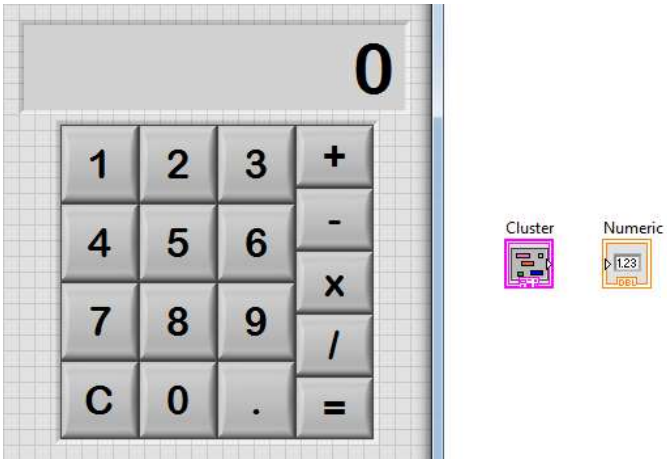
6. Klik kanan pada Cluster, dan pilih **Reorder Controls in Cluster**, maka akan muncul nomor indeks Cluster di tiap tombol secara berurutan.



Gambar 6.23 Muncul nomor indeks ketika Reorder Controls in Cluster

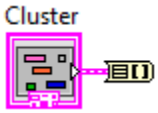
7. Atur urutan nomor indeks Cluster sehingga sesuai dengan gambar di atas, yaitu dimulai dari tombol 1 untuk nomor indeks 0, hingga tombol sama dengan (=) untuk nomor indeks 16.

8. Tambahkan objek Numeric Indicator, dan dengan tool **Text Settings** pada Toolbar, perbesar ukuran angkanya dan atur rata kanan.



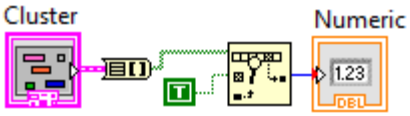
Gambar 6.24 Tambahkan objek Num Ind sebagai tampilan kalkulator

9. Diinginkan setiap kali salah satu tombol ditekan, maka objek **Num Ind** dapat menampilkan nomor indeks tombol tersebut. Untuk itu tambahkan pada icon Cluster fungsi **Cluster to Array** (dapat diambil di kategori Array), untuk mengubah Cluster menjadi Array.



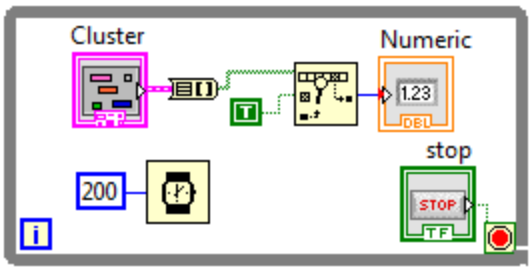
Gambar 6.25 Tambahkan Cluster to Array

10. Kemudian tambahkan fungsi **Search 1D Array** (dapat diambil di kategori Array) untuk menemukan manakah tombol dengan indeks yang bernilai **True,** dalam arti saat itu sedang ditekan, dan kemudian hubungkan output dari fungsi tersebut ke icon **Num Ind**.



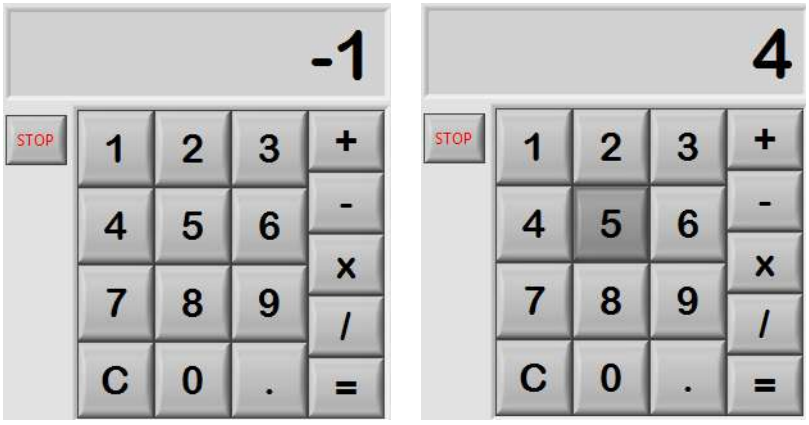
Gambar 6.26 Search 1D Array untuk mencari tombol yang ditekan

11. Tambahkan pula struktur **While Loop** untuk menjalankan terus-menerus program tersebut, dan sebuah fungsi **wait (ms)** di kategori Timing, untuk memberi tunda waktu sebesar 200 ms.



Gambar 6.27 Menambahkan While Loop dan Wait(ms) sebesar 200

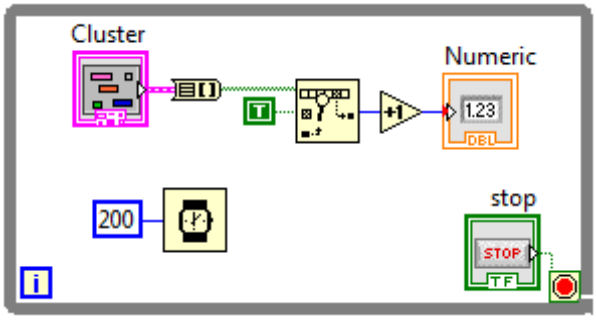
12. Jalankan program (tekan tombol Run), dan perhatikan bahwa ketika tombol tidak ditekan, maka ditampilkan angka -1, sedangkan ketika salah satu tombol ditekan, dan kemudian dilepas, muncul nomor indeksnya sesaat, kemudian kembali ke angka -1.



(a) (b)

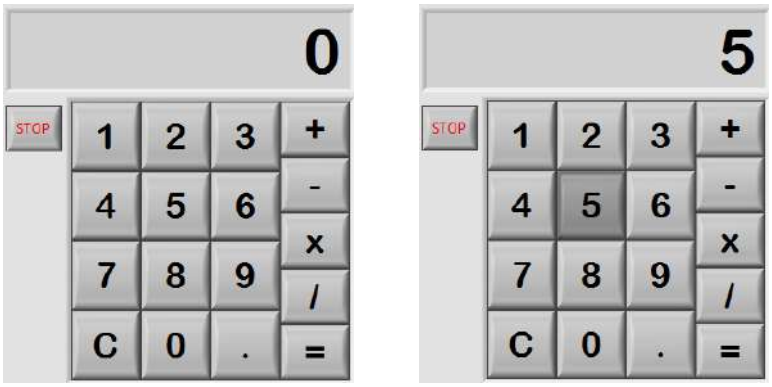
Gambar 6.28 (a) Tombol tidak ditekan, (b) Tombol angka 5 ditekan

13. Agar ketika tombol tidak ditekan, muncul angka 0, dan ketika tombol angka 5 ditekan, muncul angka 5, maka tambahkan fungsi **increment** (+1), yang dapat diambi di kategori **Numeric**.



Gambar 6.29 Menyisipkan fungsi increment untuk menambah 1 angka

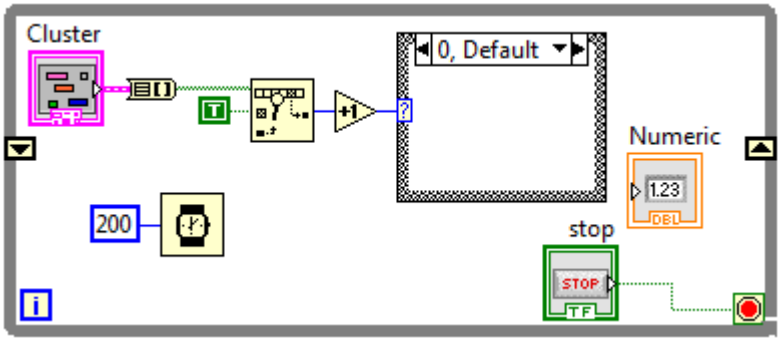
14. Jalankan program, maka ketika tombol tidak ditekan, muncul angka 0, dan ketika tombol 5 ditekan dan dilepas, muncul angka 5 sesaat.



(a) (b)

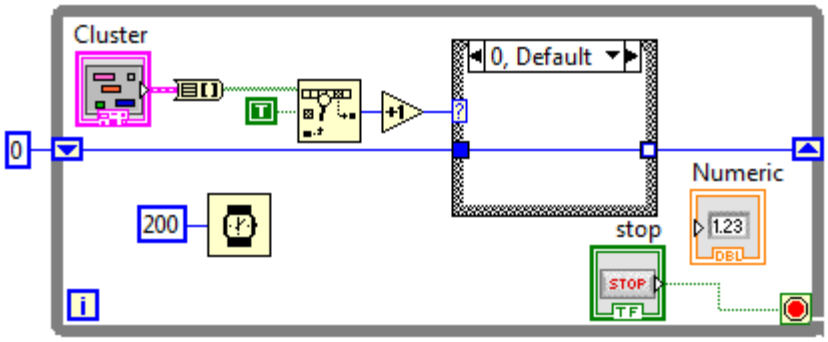
Gambar 6.30 (a) Tombol tidak ditekan, (b) Tombol angka 5 ditekan

15. Agar angka yang ditampilkan tidak hanya sesaat, tetapi dapat bertahan hingga tombol lain ditekan, maka tambahkan sebuah **Shift Register** dan sebuah struktur **Case**. Hubungkan output increment ke terminal **selector** struktur **Case,** maka secara otomatis Labelnya berubah dari **True/False** ke **0/1**, dengan nilai 0 sebagai **Default**.



Gambar 6.31 Menambahkan Shift Register dan struktur Case

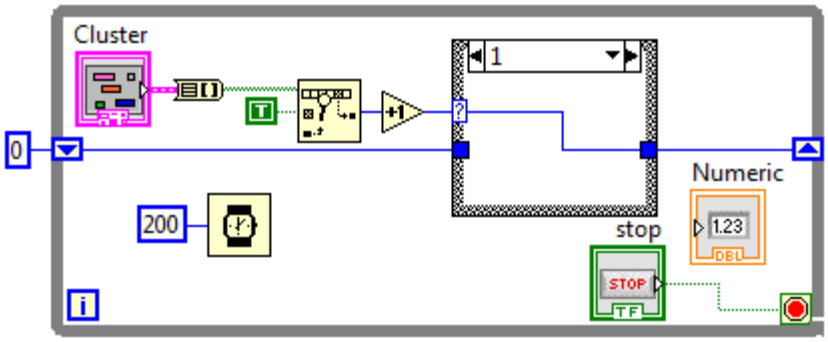
16. Beri nilai 0 sebagai nilai awalan **Shift Register**, dengan cara mengambil fungsi **Num Const** di kategori **Numeric** dan menghubungkannya ke terminal kiri **Shift Register**. Kemudian hubungkan output terminal kiri ini ke input terminal kanan **Shift Register** melalui struktur **Case** pada blok **Case 0** (Default). Hubungan ini berarti ketika tidak ada tombol yang ditekan (Default), maka data yang ada di loop sebelumnya akan digunakan untuk loop saat ini.



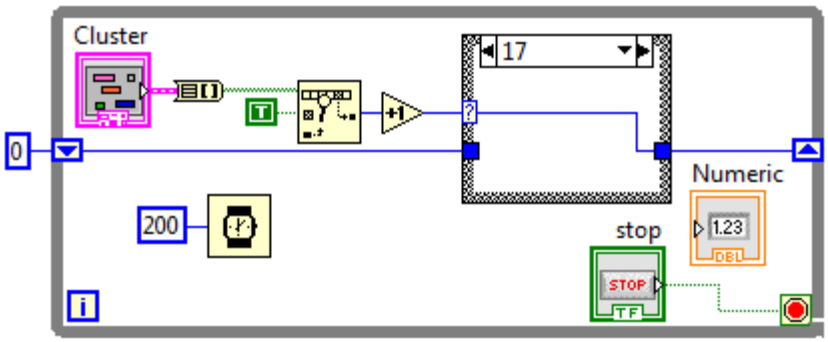
Gambar 6.32 Memberi nilai awal 0 dan mengisi nilai Shift Register saat Case Default dengan nilai data pada Loop sebelumnya

17. Perhatikan kotak terminal putih pada dinding kanan (output) struktur **Case**, yang menunjukkan bahwa ada blok yang outputnya tidak terhubung dengan kotak tersebut. Untuk itu, buka blok **Case 1**, dan hubungkan terminal kotak putih tersebut dengan terminal **selector**, yang berisi nilai tombol yang ditekan.

18. Karena ada 17 buah tombol, maka buat penambahan 16 blok lagi pada Struktur **Case**, dengan cara meng-klik kanan dinding struktur **Case** dan memilih **Add Case After** pada menu popup yang muncul sebanyak 16 kali. Kemudian hubungkan terminal **selector** untuk blok **Case** 2 hingga 17 tersebut ke terminal kotak putih, sehingga terminal kotak tersebut berwarna penuh sesuai tipe datanya.



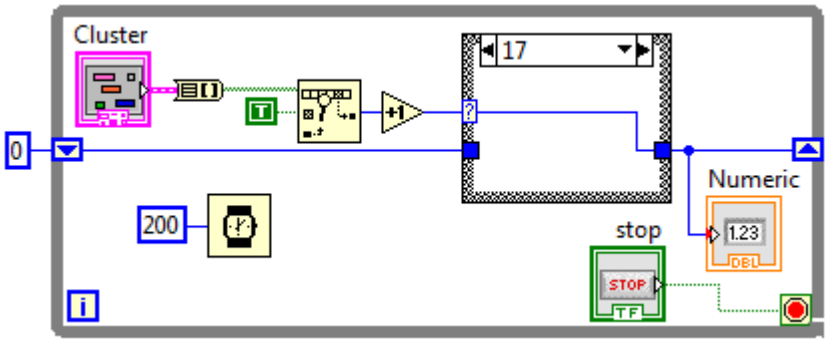
Gambar 6.33 Memberikan nilai terminal selector atau nilai tombol ke terminal kanan Shift Register pada saat blok Case 1



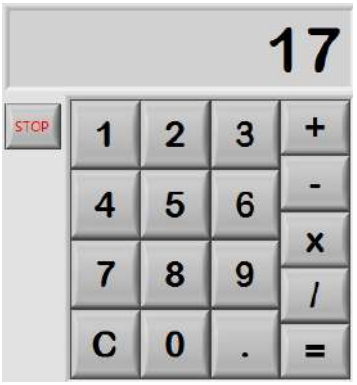
Gambar 6.34 Menambah blok Case dan mengulangi hal yang sama seperti pada blok Case 1 untuk blok Case 2 hingga blok Case 17

19. Hubungkan output struktur **Case** ke icon **Num Ind** untuk menampilkan hasilnya. Jalankan program dan perhatikan tampilan kalkulator setiap kali tombol ditekan dan dilepas. Seharusnya tampilan tersebut akan bertahan hingga tombol berikutnya ditekan. Sampai di sini, program telah sesuai dengan cara kerja pertama.

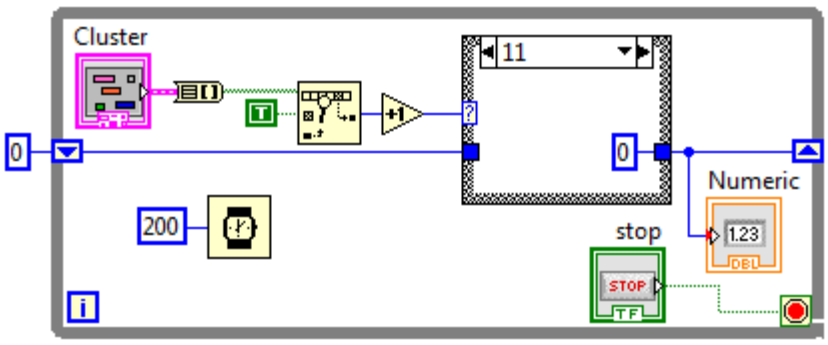
20. Sebelum melanjutkan ke cara kerja kalkulator kedua, perbaiki dulu tombol angka 0, di mana ketika tombol angka 0 ditekan, seharusnya ditampilkan angka 0, bukan angka 11. Hal ini dapat dilakukan dengan menghubungkan output blok Case 11 dengan nilai 0.



Gambar 6.35 Menghubungkan output struktur Case dengan Num Ind

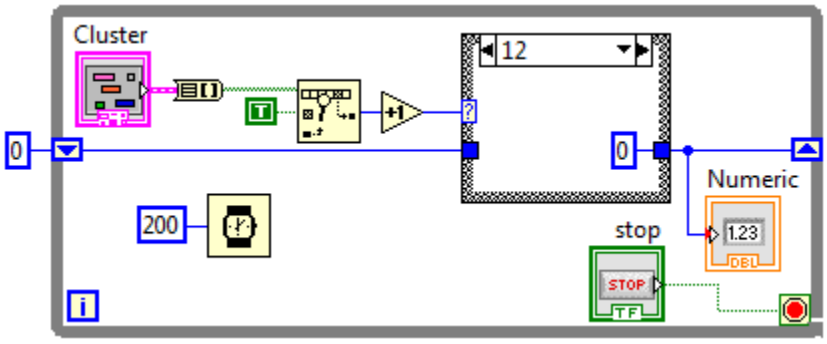


Gambar 6.36 Tampilan bertahan setelah tombol = ditekan dan dilepas



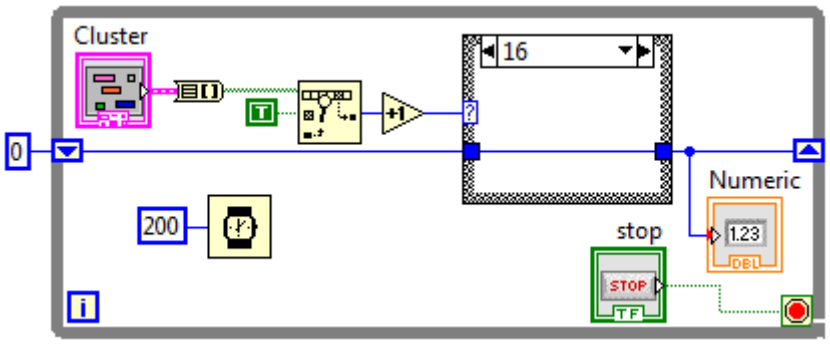
Gambar 6.37 Menghubungkan output blok Case 11 dengan nilai 0

21. Lakukan juga hal serupa dengan tombol C dan tombol . (titik), yaitu menghubungkan output blok Case 10 dan 12 dengan nilai 0.



Gambar 6.38 Menghubungkan output blok Case 12 dengan nilai 0

22. Berikutnya cara kerja kalkulator kedua, adalah membuat agar ketika tombol operator (+,-,x,/) ditekan, angka yang ditampilkan sebelumnya tidak berubah. Hal ini mudah saja dilakukan, yaitu dengan cara meniru isi dari blok Case 0 untuk blok Case 13, 14, 15, dan 16, yang merupakan blok Case untuk tombol operator (+,-,x,/).



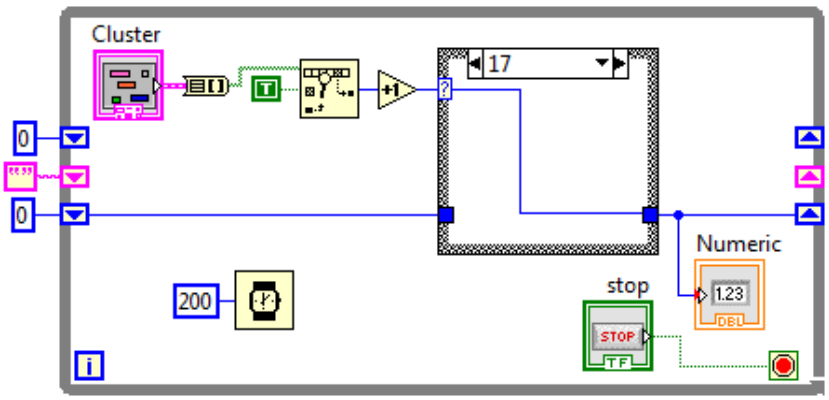
Gambar 6.39 Agar tampilan tidak berubah, blok Case 13, 14, 15 dan 16 meneruskan data Loop sebelumnya untuk data Loop berikutnya

23. Karena cara kerja kalkulator ketiga sama dengan cara kerja pertama, maka berikutnya adalah cara kerja keempat.

Cara kerja keempat ini akan menampilkan hasil perhitungan ketika tombol (=) ditekan. Berarti ketika tombol (=) ini ditekan, blok Case 17 akan bekerja melakukan perhitungan dengan 3 input yang sudah tersedia, yaitu angka pertama, kedua, dan operator perhitungan.

Pada langkah inilah konsep **State Machine** diperlukan. Jadi pada kasus ini, ada 3 keadaan yang telah dibuat, yaitu: angka pertama, angka kedua dan operator perhitungan yang dipilih. Sedangkan input pemicunya adalah tombol (=). Karena ketiga keadaan tersebut tidak dibuat pada saat tombol (=) ditekan, tetapi dibuat sebelumnya, maka Shift Register diperlukan. Karena Shift Register hanya bisa membawa satu jenis data, berarti ada 3 buah Shift Register yang diperlukan untuk membawa ketiga keadaan tersebut.

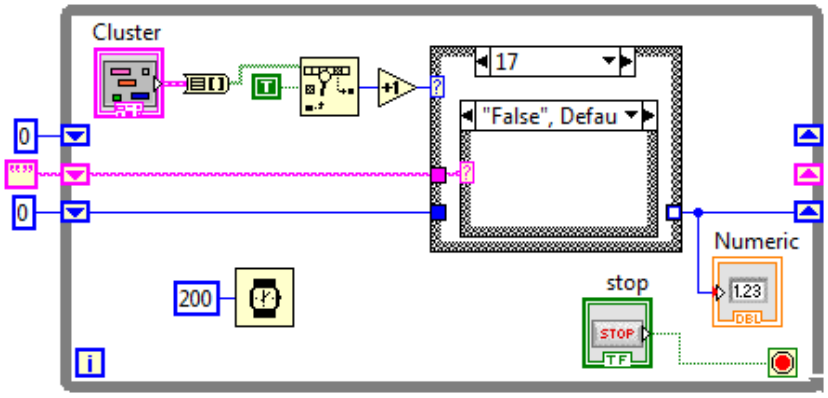
Untuk itu, tambahkan 2 buah Shift Register lagi. Untuk Shift Register pertama, Beri nilai awal dengan Num Const (angka 0), sedangkan untuk Shift Register kedua, beri nilai awal dengan Empty String.



Gambar 6.40 Menambahkan 2 buah Shift Register dengan nilai awal Numeric Constant (angka 0) dan Empty String Constant

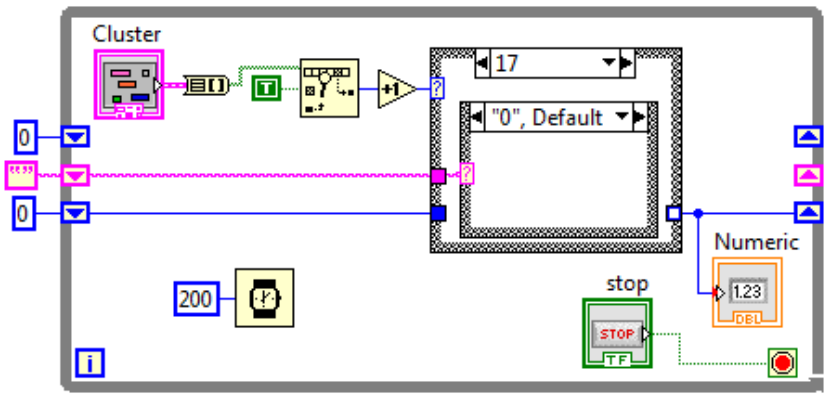
24. Pada blok Case 17, putus garis dari terminal selector ke output blok, dan kemudian tambahkan sebuah struktur Case di dalamnya.

Hubungkan terminal selector struktur Case yang baru ini dengan terminal kiri Shift Register yang memiliki nilai awal Empty String.



Gambar 6.41 Menambahkan struktur Case di dalam blok Case 17

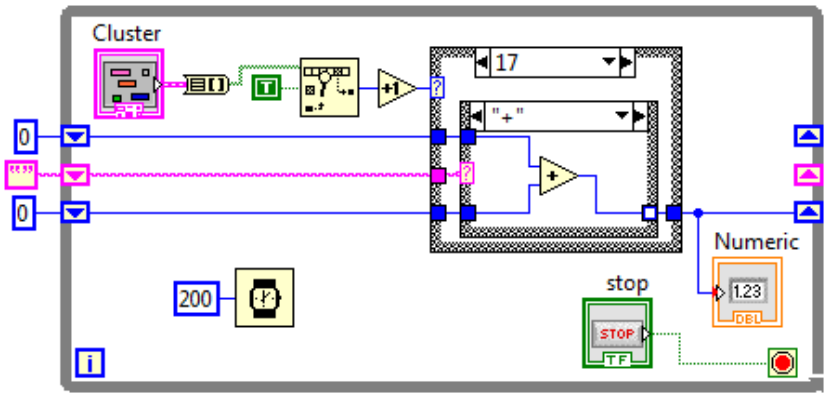
25. Struktur Case yang baru ini digunakan untuk melakukan operasi perhitungan. Untuk itu ganti nama label “False”, Default dengan “0”, Default, dan blok berikutnya berlabel “True” dengan “+”.



Gambar 6.42 Mengganti label “False” dengan “0”, “True” dengan “+”

26. Kemudian pada blok **“+”**, tambahkan fungsi Add, di mana input pertama fungsi Add ini dihubungkan dengan terminal kiri Shift

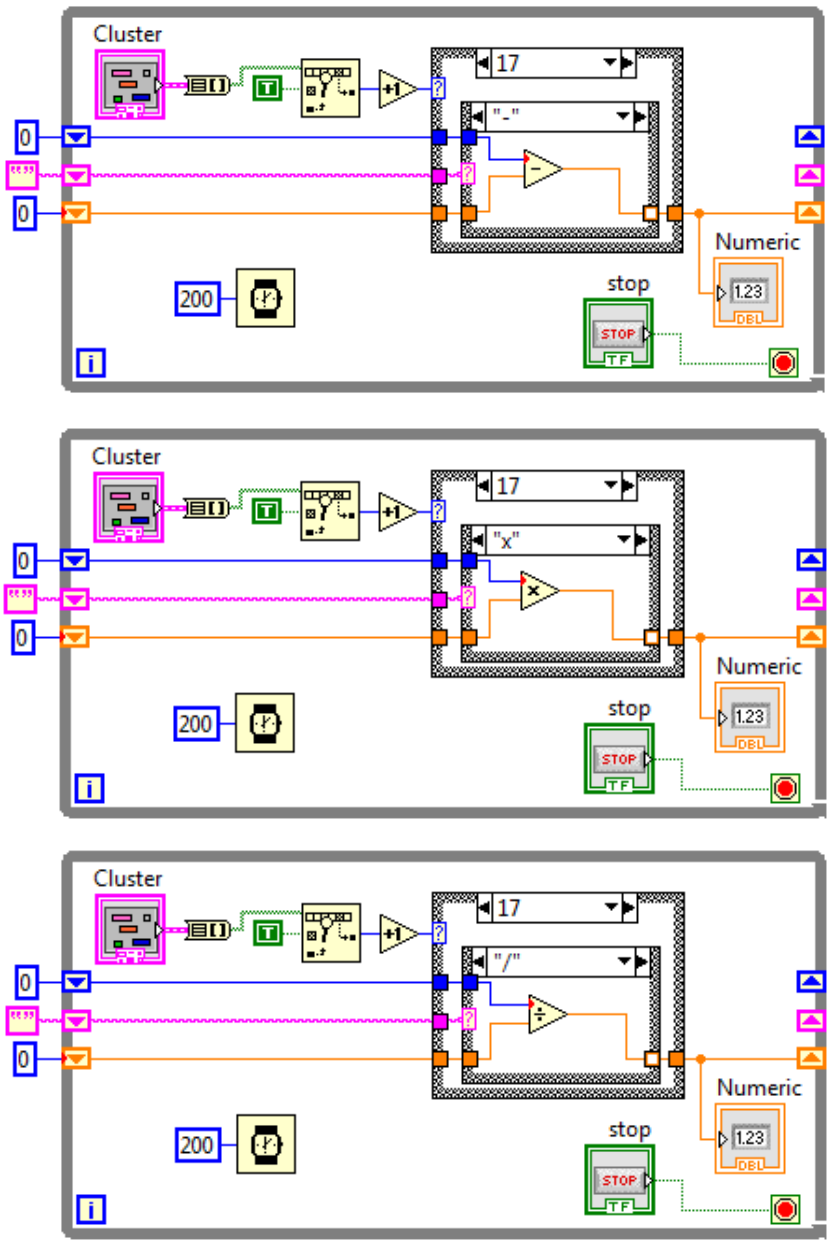
Register yang baru dengan nilai awal 0, dan input kedua fungsi Add ini dihubungkan dengan terminal kiri Shift Register yang lama. Output dari fungsi Add ini dihubungkan dengan output blok Case 17.



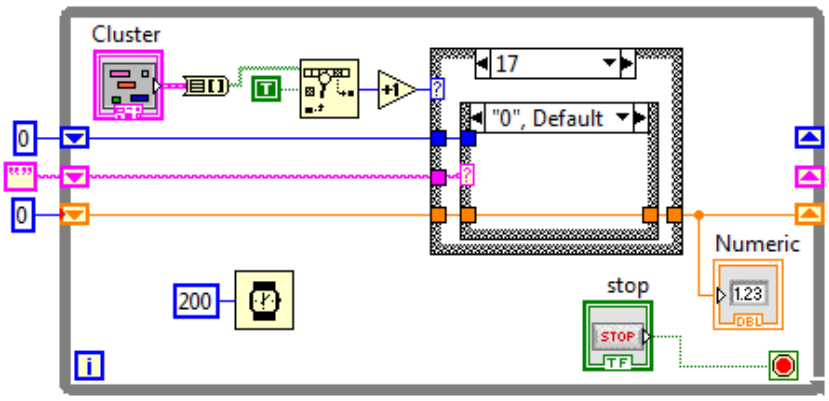
Gambar 6.43 Menambah fungsi Add di blok Case "+"

27. Tambahkan 3 buah blok Case lagi dengan meng-klik kanan dinding struktur Case yang baru dan memilih **Add Case After** pada menu popup yang muncul. Beri label berturut-turut **"-"**, **"x"** dan **"/"** dan tambahkan fungsi **Subtract** pada blok **Case "-"**, fungsi **Multiply** pada blok **Case "x"**, fungsi **Divide** pada blok **Case "/"**. Kemudian hubungkan input dan output untuk ketiga fungsi tersebut sama seperti pada input dan output fungsi Add.

28. Perhatikan bahwa kotak output struktur Case yang baru masih berwarna putih, karena ada blok yang belum menghubungkannya, yaitu blok **Case "0", Default**. Untuk itu pada blok **Case "0", Default**, hubungkan kotak terminal putih tersebut dengan input dari Shift Register yang lama. Ini berarti ketika tidak ada tombol operator yang dipilih (atau Default), maka data sebelumnya akan diteruskan.

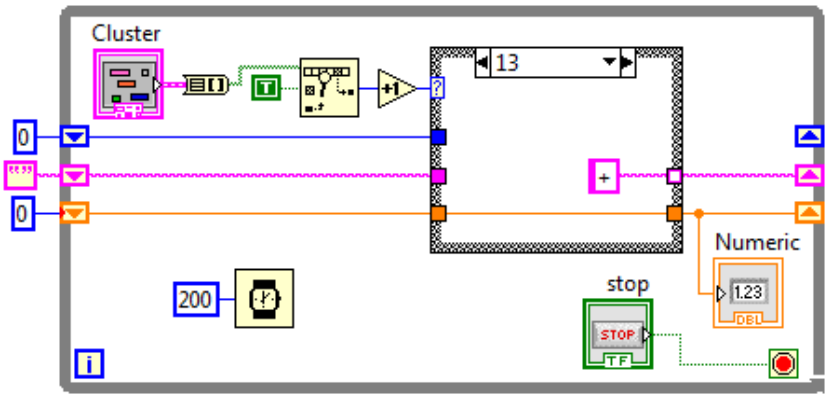


Gambar 6.44 Menambah fungsi Subtract di blok Case "-", fungsi Multiply di blok Case "x", dan fungsi Divide di blok Case "/"



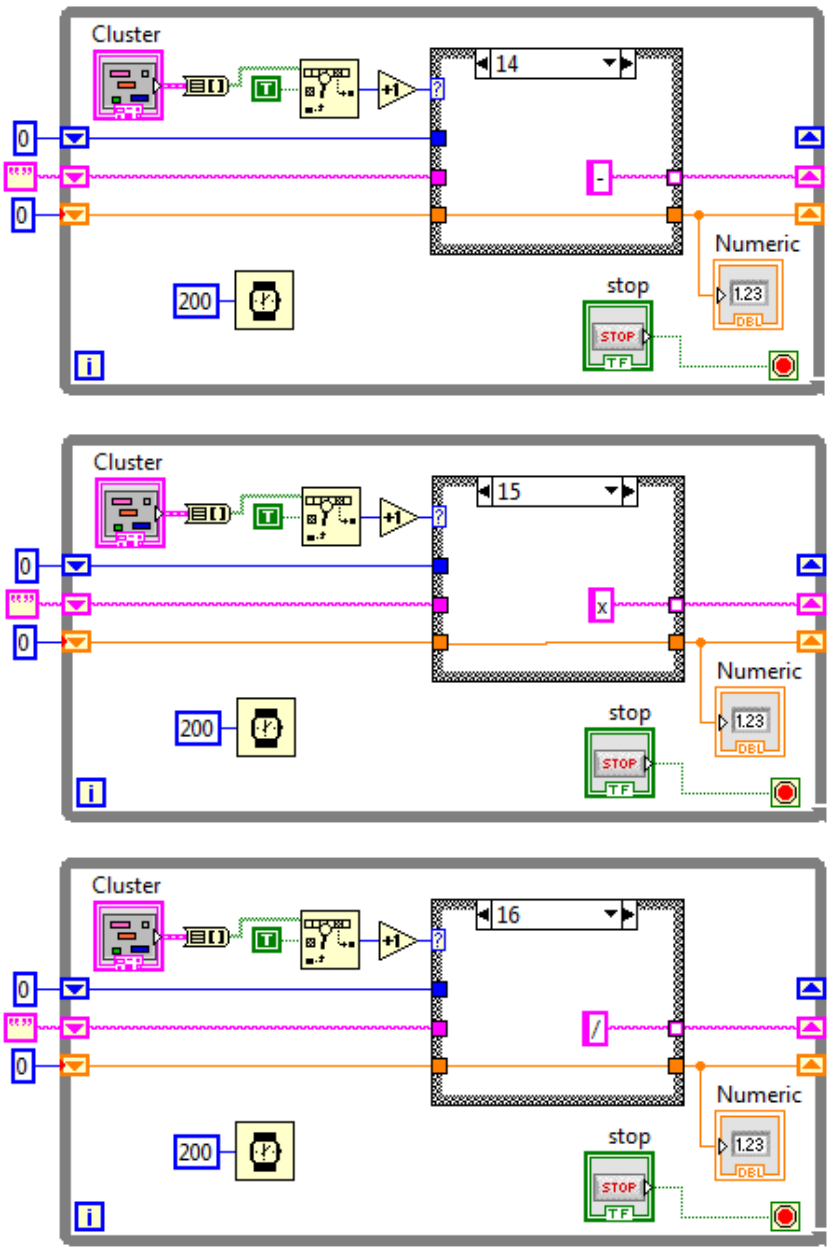
Gambar 6.45 Meneruskan data pada blok Case "0", Default

29. Karena Shift Register dengan nilai awal Empty String digunakan untuk membawa data jenis operator, maka tambahkan pada blok Case 13 yang merupakan blok untuk tombol +, sebuah string dengan karakter **+** yang dihubungkan dengan terminal kanan Shift Register.



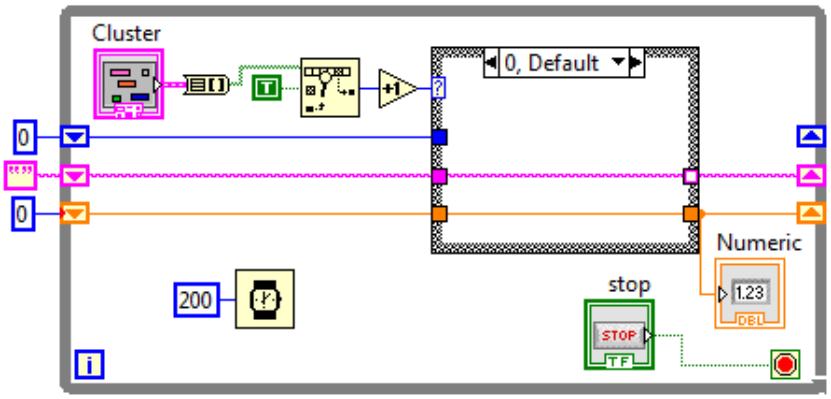
Gambar 6.46 Memberikan karakter + pada Shift Register ketika blok Case 13 bekerja atau apabila tombol + ditekan

30. Ulangi langkah di atas untuk blok Case 14, 15 dan 16 yang berturut-turut merupakan blok untuk tombol -, x dan /, yaitu dengan menghubungkan karakter -, x, dan / dengan Shift Register.

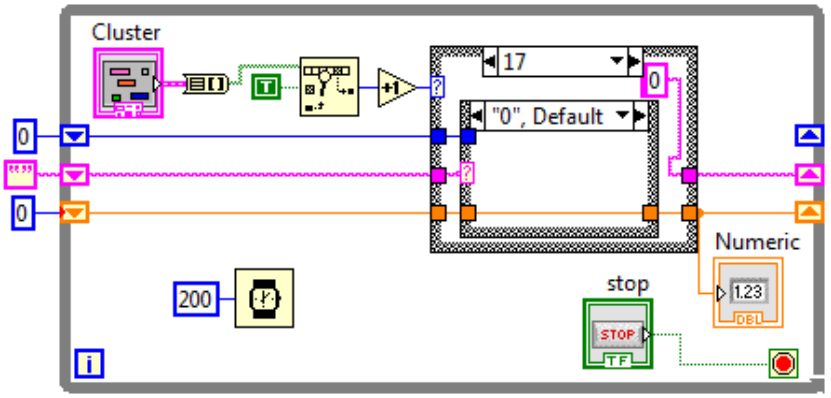


Gambar 6.47 Memberikan karakter -, x, atau / pada Shift Register apabila blok Case 14, 15 atau 16 bekerja, atau ketika tombol -,x atau / ditekan

31. Tampak bahwa kotak terminal operator masih berwarna putih, karena tidak semua blok terhubung dengan kotak terminal tersebut. Untuk itu hubungkan kotak terminal tersebut dengan input dari terminal kiri Shift Register yang membawa karakter operator, pada semua sisa blok Case, yaitu blok Case 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 dan 12. Khusus untuk blok Case 17, beri karakter 0.

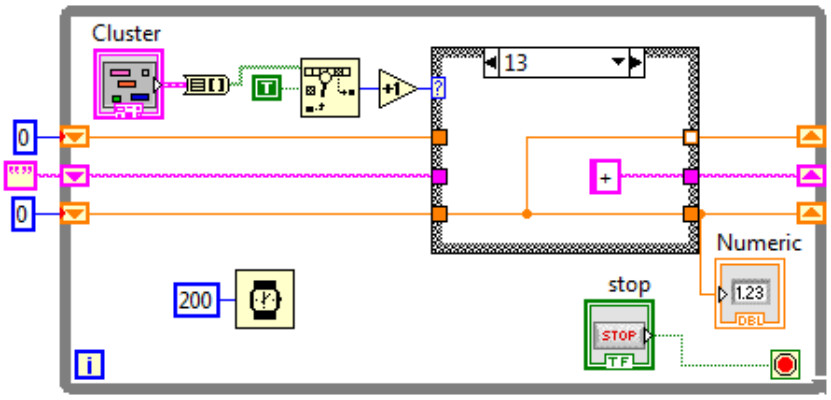


Gambar 6.48 Untuk blok Case 0 sampai 12, teruskan data dari terminal kiri ke terminal kanan Shift Register untuk membawa karakter operator



Gambar 6.49 Untuk blok Case 17, beri karakter 0 pada kotak terminal

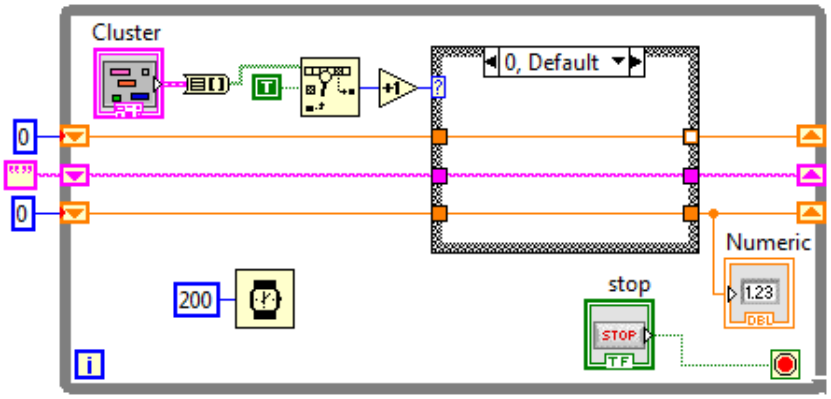
32. Kemudian ketika tombol operator ditekan, maka seharusnya data angka pertama tidak hanya diteruskan untuk ditampilkan, tetapi juga disimpan pada Shift Register yang baru, agar ketika tombol (=) ditekan, data angka pertama tersebut dapat diambil. Maka tambahkan garis data yang menghubungkan data dari Shift Register lama ke Shift Register baru untuk blok Case 13, 14, 15 dan 16.



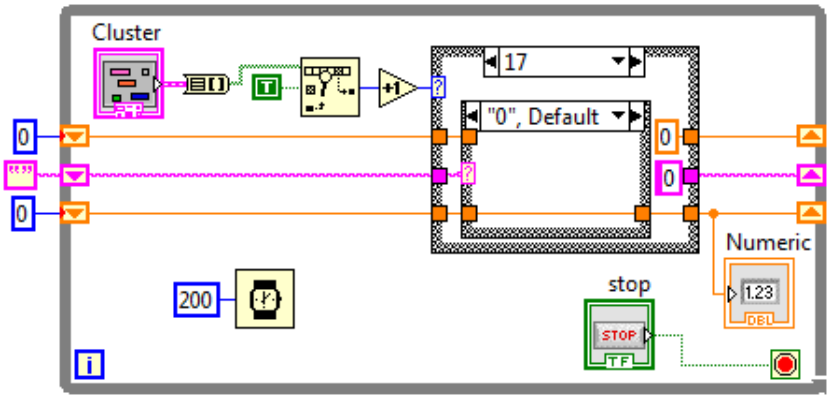
Gambar 6.50 Untuk Blok Case 13-16, hubungkan kedua Shift Register

33. Tampak kotak masih berwarna putih. Untuk itu, pada blok Case 0-12, hubungkan input dari terminal kiri Shift Register baru ke kotak terminal putih, untuk meneruskan data ketika tidak ada tombol operator yang ditekan. Khusus untuk blok Case 17, beri angka 0.

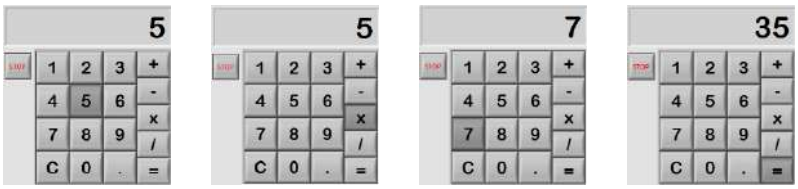
34. Jalankan program, maka simulasi kalkulator akan bekerja sesuai cara kerja yang diminta. Namun demikian, masih banyak keterbatasan dalam program ini, salah satunya adalah hanya bisa memasukkan 1 digit angka saja, yaitu angka antara 0-9. Begitu pula angka pecahan belum bisa, karena tombol titik belum berfungsi. Bab 8 akan memberikan solusi untuk permasalahan ini. Terlepas dari semua keterbatasan itu, pembaca telah mempelajari konsep **State Machine**, yang banyak dipakai dalam pemrograman LabVIEW.



Gambar 6.51 Untuk blok Case 0-12, teruskan data Shift Register



Gambar 6.52 Untuk Blok Case 17, beri angka 0 pada Shift Register



(a) tombol 5 (b) tombol x (c) tombol 7 (d) tombol =

Gambar 6.53 Program dijalankan untuk menghitung 5x7

BAB 7
ARRAY, CLUSTER, CHART DAN GRAPH

7.1 Array dan Cluster

Pembuatan simulasi kalkulator di bab sebelumnya, sudah menggunakan Array dan Cluster. Apakah sebenarnya Array dan Cluster itu? Apakah persamaan dan perbedaan di antara keduanya?

Array dan Cluster adalah sama-sama bentuk pengelompokan data, hanya bedanya, tipe data dalam satu Array harus sejenis, sedangkan tipe data dalam satu Cluster bisa bermacam-macam. Untuk lebih jelas mengenai penggunaan, cara pembuatan dan fungsi dari Array dan Cluster, berikut ini penjelasan masing-masing.

7.1.1 Array

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 jenis indeks. Sedangkan untuk Array 2D (sering disebut Matriks), memiliki 2 jenis indeks, yaitu indeks baris dan indeks kolom. Nomor indeks tersebut selalu dimulai dari 0.

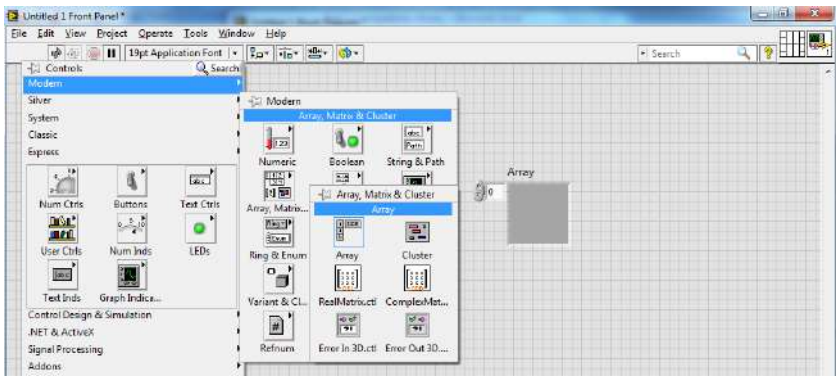
Array biasa digunakan untuk menyimpan data dalam jumlah yang besar, yang diperoleh dari suatu struktur perulangan. Array sangat berguna untuk menyederhanakan program dalam melakukan operasi atau komputasi yang berulang pada sekumpulan data yang sejenis.

7.1.2 Menciptakan Array

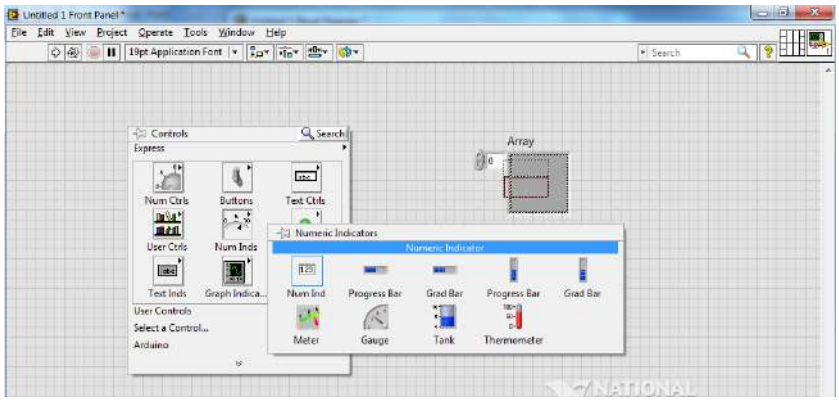
Ada 2 cara untuk menciptakan sebuah Array. Cara pertama adalah dengan memasukkan nilai elemen Array satu persatu dari objek Array di jendela Front Panel. Sedangkan cara kedua adalah secara otomatis membangkitkan Array dengan menggunakan struktur Loop di jendela Block Diagram.

Berikut langkah-langkah untuk membuat Array dengan cara pertama:

1. Klik kanan jendela Front Panel. Pada palet Controls yang muncul, pilih kategori **Modern**, kemudian **Array, Matrix & Cluster**, kemudian **Array**. Tempatkan objek Array tersebut di jendela Front Panel.
2. Buka kembali palet Controls, dan pilih objek **Num Ind** di kategori Numeric Indicator. Tarik objek tersebut dan tempatkan di dalam kotak Array, maka akan terbentuk sebuah Array 1D dengan elemen **Num Ind**. Tarik tepi kotak Array tersebut untuk melebarkan isinya sehingga muncul kotak-kotak elemen yang baru.

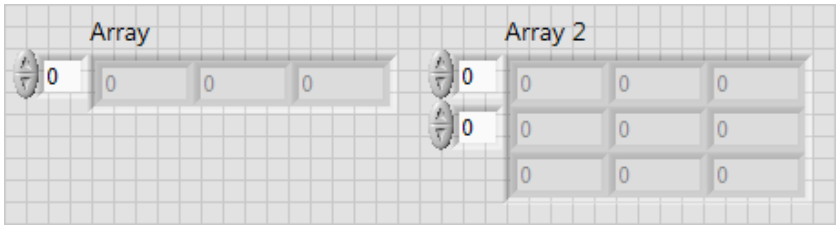


Gambar 7.1 Menciptakan Array dari jendela Front Panel



Gambar 7.2 Menempatkan objek Num Ind di kotak Array

3. Untuk membuat Array 2D, maka klik kanan kotak indeks di kiri Array dan pilih **Add Dimension**, maka kotak Indeks akan menjadi 2.

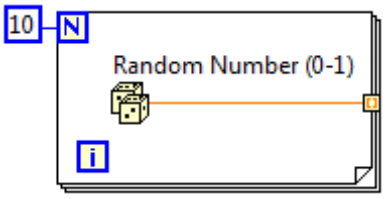


Gambar 7.3 Array 1D (1 kotak indeks) dan Array 2D (2 kotak indeks)

Catatan: selama kotak-kotak elemen di jendela Front Panel masih pudar tampilannya, berarti angka di kotak-kotak tersebut bisa diabaikan, karena bukan elemen dari Array. Namun bila kotak-kotak tersebut tidak lagi pudar, maka isi kotak tersebut merupakan elemen dari Array.

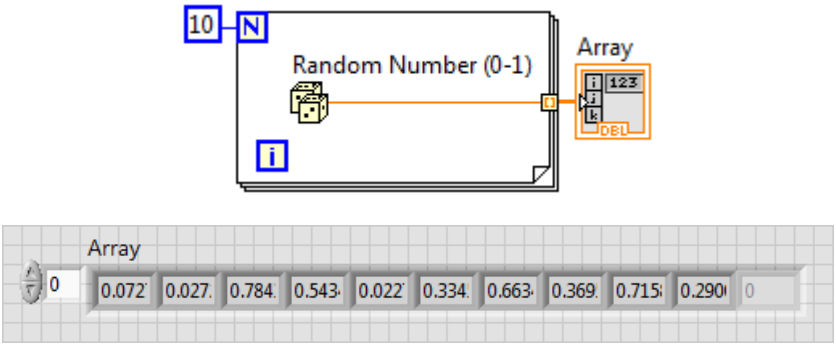
Berikut langkah-langkah untuk membuat Array dengan cara kedua:

1. Untuk membuat Array 1D dengan elemen sebanyak 10 buah, maka tempatkan For Loop di jendela Blok Diagram, dan beri angka 10 pada terminal input N. Kemudian tambahkan sebuah fungsi Random, tempatkan di dalam For Loop dan keluarkan outputnya.



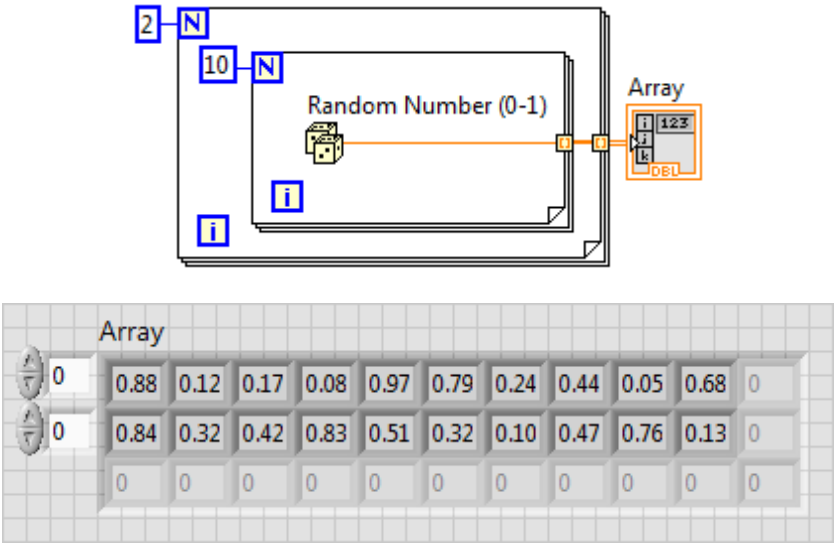
Gambar 7.4 Membuat Array 1D dengan elemen sebanyak 10 buah

2. Klik kanan pada kotak terminal bertanda kurung, dan pilih **Create**, kemudian pilih **Indicator** pada menu popup yang muncul. Jalankan program, maka sebuah Array 1D dihasilkan di jendela Front Panel.



Gambar 7.5 Hasil Array 1D dengan elemen sebanyak 10 buah

3. Untuk membuat Array 1D menjadi 2D, tambahkan sebuah For Loop lagi, melingkupi For Loop yang pertama. For Loop yang luar akan menciptakan elemen baris, sedangkan For Loop yang dalam akan menciptakan elemen kolom. Berikut contoh pembuatan Array 2D dengan 2 baris dan 10 kolom. Karena Indicator sebelumnya adalah Array 1D, maka hapus indicator tersebut dan buat yang baru dengan cara meng-klik kanan pada kotak output For Loop dan memilih **Create Indicator** pada menu popup yang muncul.



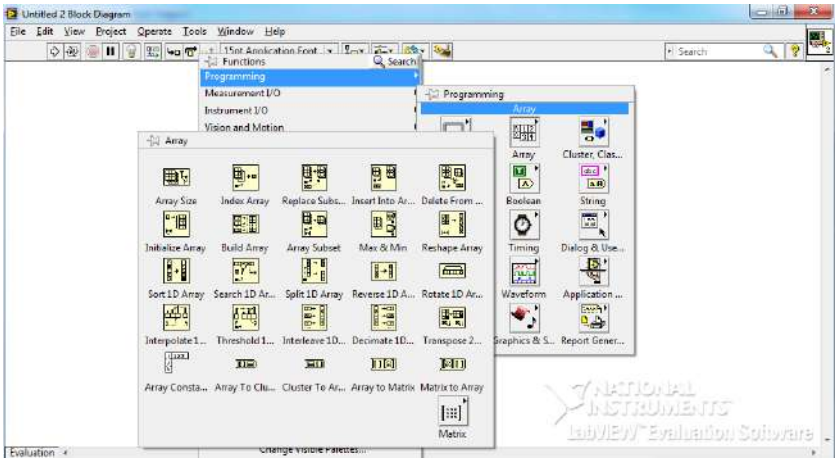
Gambar 7.6 Membuat Array 2D dengan 2 baris dan 10 kolom

While Loop dan For Loop dapat menghasilkan Array karena **Indexing** di-"enable". Jika **Indexing** di-"disable", maka While Loop dan For Loop hanya menghasilkan satu data saja, yaitu data dari Loop yang terakhir.

Sebuah Array dari Array tidak dapat diciptakan. Untuk menciptakan sebuah Array Multidimensi, dapat dilakukan dengan membuat Array dari Cluster, di mana setiap Cluster bisa berisi satu atau lebih Array.

7.1.3 Fungsi-Fungsi Array

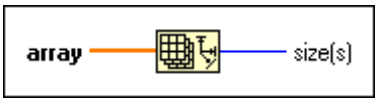
LabVIEW menyediakan banyak sekali fungsi untuk memanipulasi Array. Pembaca dapat membuka fungsi-fungsi Array tersebut dengan meng-klik kanan jendela Block Diagram, dan kemudian memilih kategori **Programming**, diikuti dengan **Array**.



Gambar 7.7 Fungsi-fungsi Array

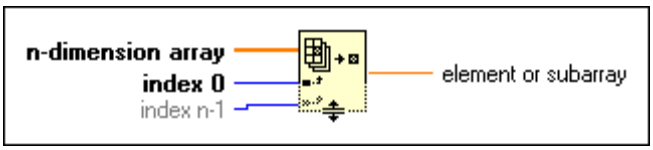
Berikut penjelasan singkat fungsi-fungsi tersebut:

1. **Array Size.** Fungsi ini menghasilkan jumlah elemen dalam setiap dimensi dari sebuah Array.



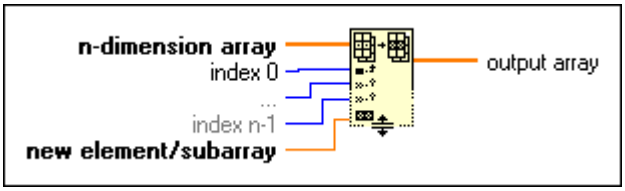
Gambar 7.8 Array Size

2. **Index Array.** Fungsi ini menghasilkan nilai elemen dari sebuah Array 1D atau subarray dari sebuah Array 2D sesuai indeksnya.



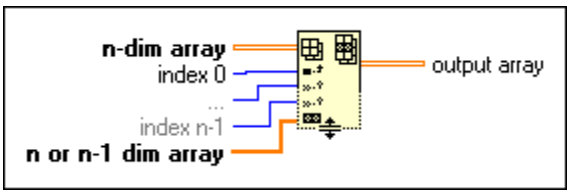
Gambar 7.9 Index Array

3. **Replace Array Subset.** Fungsi ini mengganti nilai yang ada di Array dengan nilai yang baru dengan lokasi sesuai indeks yang diberikan.



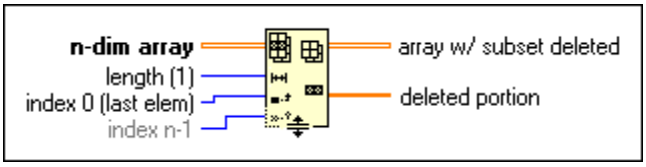
Gambar 7.10 Replace Array Subset

4. **Insert Into Array.** Fungsi ini menyisipkan nilai ke Array pada indeks yang diberikan.



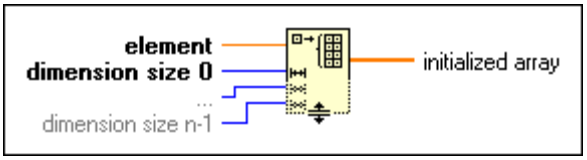
Gambar 7.11 Insert Into Array

5. **Delete From Array.** Fungsi ini menghilangkan elemen atau subarray sesuai indeks yang diberikan.



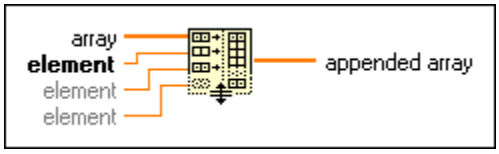
Gambar 7.12 Delete From Array

6. **Initialize Array.** Fungsi ini menciptakan sebuah Array dengan nilai elemen dan ukuran dapat ditentukan.



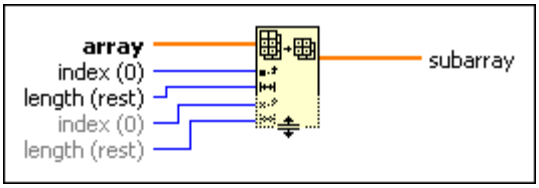
Gambar 7.13 Initialize Array

7. **Build Array.** Fungsi ini menggabungkan Array atau menambahkan elemen atau subarray ke sebuah Array (concatenate).



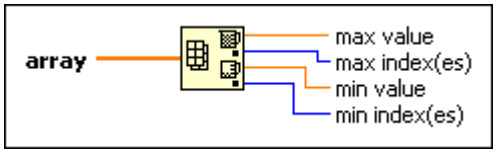
Gambar 7.14 Build Array

8. **Array Subset.** Fungsi ini menghasilkan irisan bagian dari sebuah array yang dimulai dari indeks dan jumlah elemen tertentu.



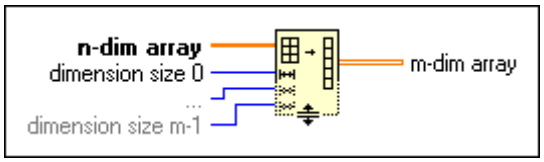
Gambar 7.15 Array Subset

9. **Array Max & Min**. Fungsi ini digunakan untuk mencari nilai yang terbesar dan terkecil dari suatu Array beserta nomor indeksnya.



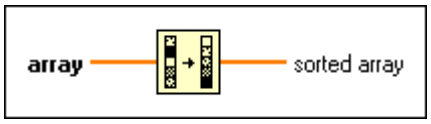
Gambar 7.16 Array Max & Min

10. **Reshape Array**. Fungsi ini untuk mengubah dimensi Array



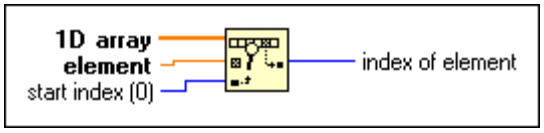
Gambar 7.17 Reshape Array

11. **Sort 1D Array**. Fungsi ini digunakan untuk mengurutkan elemen Array 1D dari nilai kecil ke nilai besar.



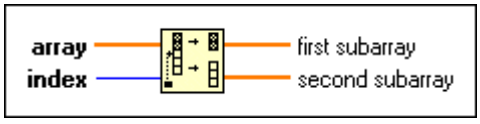
Gambar 7.18 Sort 1D Array

12. **Search 1D Array**. Fungsi ini digunakan untuk mencari nomor indeks dari suatu nilai elemen di Array 1D.



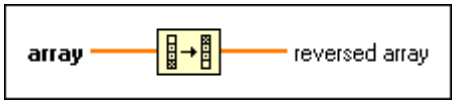
Gambar 7.19 Search 1D Array

13. **Split 1D Array**. Fungsi ini digunakan untuk membagi 2 sebuah Array 1D dengan batas tengah pemotongan ditentukan oleh nilai elemen yang dimasukkan.



Gambar 7.20 Split 1D Array

14. **Reverse 1D Array**. Fungsi ini digunakan untuk membalik urutan semua elemen Array 1D.

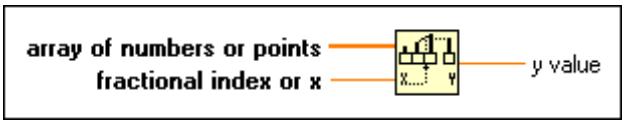


Gambar 7.21 Reverse 1D Array

15. **Rotate 1D Array**. Fungsi ini digunakan untuk menggeser ke kanan posisi elemen beberapa langkah sesuai nilai yang dimasukkan.

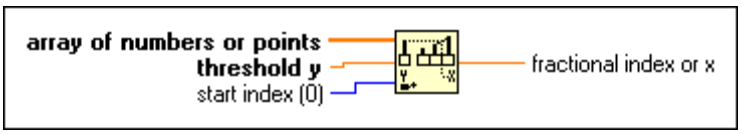
Gambar 7.22 Rotate 1D Array

16. **Interpolate 1D Array**. Fungsi ini digunakan untuk mendapatkan rata-rata dari elemen-elemen yang berada di kanan dan di kiri nomor indeks pecahan yang dimasukkan. Contoh, untuk nomor indeks pecahan sebesar 2,5, maka hasilnya adalah nilai rata-rata elemen di nomor indeks 2 dan elemen di nomor indeks 3.



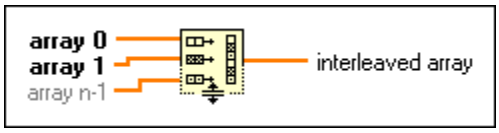
Gambar 7.23 Interpolate 1D Array

17. **Threshold 1D Array**. Fungsi ini menghasilkan nilai indeks pecahan, yang merupakan titik tengah antara nomor indeks dari 2 nilai elemen, di mana rata-rata kedua elemen tersebut adalah nilai ambang (*threshold*) yang dimasukkan ke dalam fungsi. Syarat untuk fungsi ini, nilai semua elemennya harus urut dari kecil ke besar.



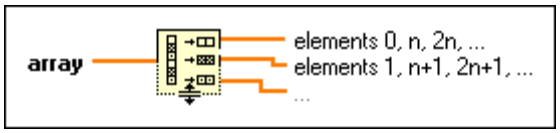
Gambar 7.24 Threshold 1D Array

18. **Interleave 1D Array**. Fungsi ini digunakan untuk menggabungkan beberapa Array menjadi satu Array, di mana elemen dari Array yang baru tersebut dimulai dari nomor indeks terkecil dari semua Array, kemudian naik hingga nomor indeks terbesar dari semua Array.



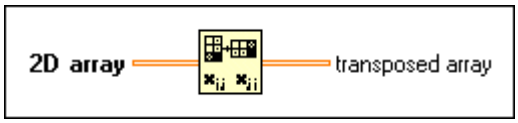
Gambar 7.25 Interleave 1D Array

19. **Decimate 1D Array**. Fungsi ini merupakan kebalikan dari Interleave 1D Array, di mana fungsi ini akan membagi Array menjadi beberapa Array yang lebih kecil.



Gambar 7.26 Decimate 1D Array

20. **Transpose 2D Array**. Fungsi ini digunakan untuk memindahkan elemen-elemen dengan menukar posisi kolom menjadi baris dan sebaliknya pada sebuah Array 2D.

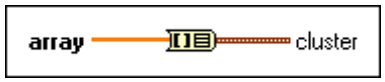


Gambar 7.27 Transpose 2D Array

21. **Array Constant**. Fungsi ini sama seperti objek Array di jendela Front Panel, yang mana digunakan untuk membuat Array dengan cara memasukkan nilai ke dalam kotaknya di jendela Block Diagram.

Gambar 7.28 Array Constant

22. **Array to Cluster**. Fungsi ini mengubah Array 1D menjadi Cluster.

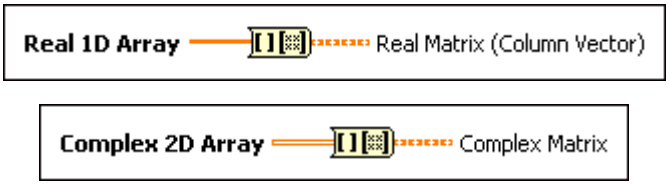


Gambar 7.29 Array to Cluster

23. **Cluster to Array**. Fungsi ini kebalikan dari fungsi Array to Cluster.

Gambar 7.30 Cluster to Array

24. **Array to Matrix**. Fungsi ini mengubah Array 1D menjadi Vektor atau Array 2D menjadi Matriks



Gambar 7.31 Real 1D Array to Vector atau Complex 2D Array to Matrix

7.1.4 Cluster

Seperti telah dituliskan di awal, Cluster adalah sama seperti Array, yaitu suatu kelompok data, hanya bedanya, komponen-komponen data Array harus bertipe data sama, sedangkan komponen-komponen data Cluster bisa berbeda tipe datanya.

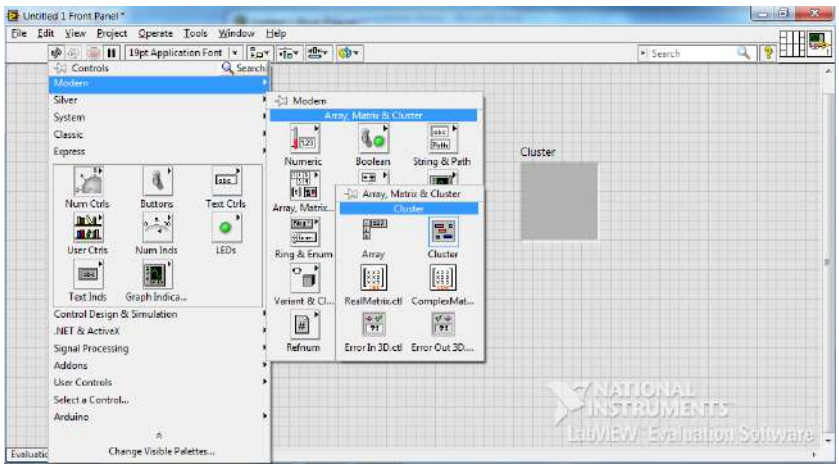
Cluster dan Array sama-sama memiliki urutan, apabila di Array, urutan tersebut dinyatakan dalam indeks, maka di Cluster urutan tersebut dinyatakan dalam order. Berbeda dengan Array yang nilai indeks elemennya berhubungan dengan posisi elemen tersebut, nilai order komponen Cluster tidak berhubungan dengan posisinya. Nilai order ini ditentukan oleh urutan penempatan. Objek yang pertama kali ditempatkan memiliki nilai order 0, selanjutnya 1, dan seterusnya. Nilai order semua objek di dalam Cluster ini dapat diubah dengan meng-klik kanan tepi Cluster dan memilih **Reorder Controls in Cluster**.

Kelebihan Cluster dibanding Array adalah pada kemudahan dan kesederhanaan dalam mengelompokkan datanya, dan ukuran data ini tetap, sehingga memastikan jumlah data tidak berubah. Sedangkan Array memiliki kelebihan pada fungsi-fungsinya yang lengkap untuk manipulasi kelompok data. Dengan menggunakan fungsi **Cluster to Array** dan **Array to Cluster** untuk berpindah dari Cluster ke Array dan sebaliknya (dengan catatan tipe data Cluster adalah sama), membuat kelebihan Array dan Cluster tersebut dapat dimanfaatkan.

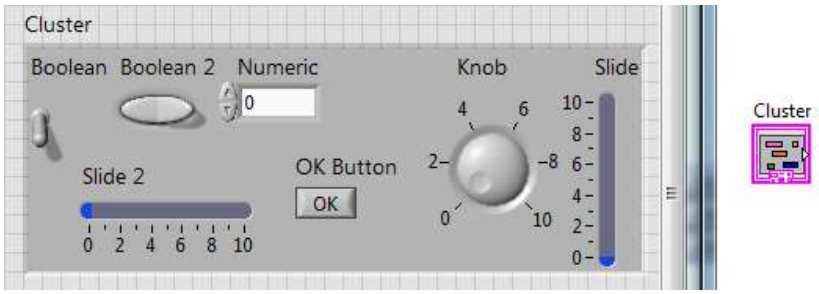
7.1.5 Menciptakan Cluster

Berikut langkah-langkah untuk membuat Cluster di jendela Front Panel:

1. Klik kanan jendela Front Panel. Pada palet Controls yang muncul, pilih kategori **Modern**, kemudian **Array, Matrix & Cluster**, kemudian **Cluster**. Tempatkan objek Cluster tersebut di jendela Front Panel.
2. Ambil objek yang akan menjadi data Cluster dari palet Controls, dan tempatkan di dalam kotak Cluster. Objek-objek tersebut bisa berupa objek-objek Control saja atau objek-objek Indicator saja, namun tidak bisa keduanya. Perbesar ukuran kotak Cluster bila data yang dimasukkan cukup banyak. Maka sebuah Cluster telah tercipta.



Gambar 7.32 Menciptakan Cluster dari jendela Front Panel



Gambar 7.33 Menempatkan berbagai objek Control di kotak Cluster dan icon representasinya di Block Diagram

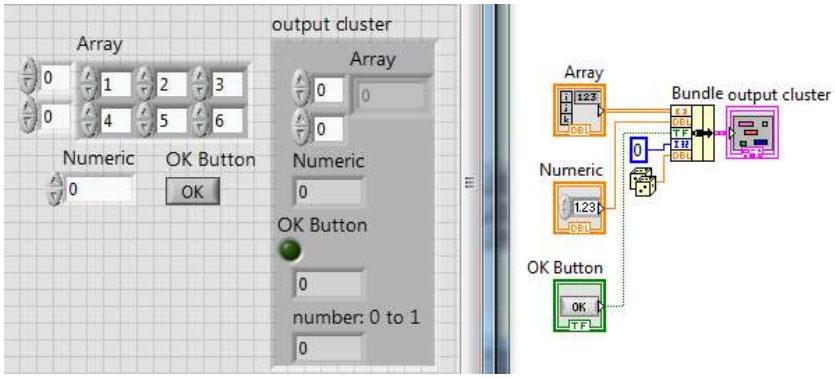
Perhatikan gambar di atas, sekalipun objek di Front Panel lebih dari satu, namun icon representasi objek di Block Diagram hanya satu buah, yaitu icon Cluster, yang mana membuat pemrograman menjadi sederhana.

Berikut langkah untuk membuat Cluster dari jendela Block Diagram:

1. Klik kanan pada jendela Block Diagram untuk memunculkan palet **Function**, kemudian pilih **Programming**, terus **Cluster, Class & Variant**, dilanjutkan dengan meng-klik **Bundle**.

2. Tempatkan icon fungsi Bundle tersebut di jendela Block Diagram. Tambahkan icon-icon terminal (Control atau Indicator), konstanta, fungsi, dll. sebagai input ke kaki input Bundle tersebut. Tarik icon Bundle ke atas atau ke bawah bila diinginkan penambahan input. Perhatikan bahwa tipe data dari setiap input ditampilkan pada icon.

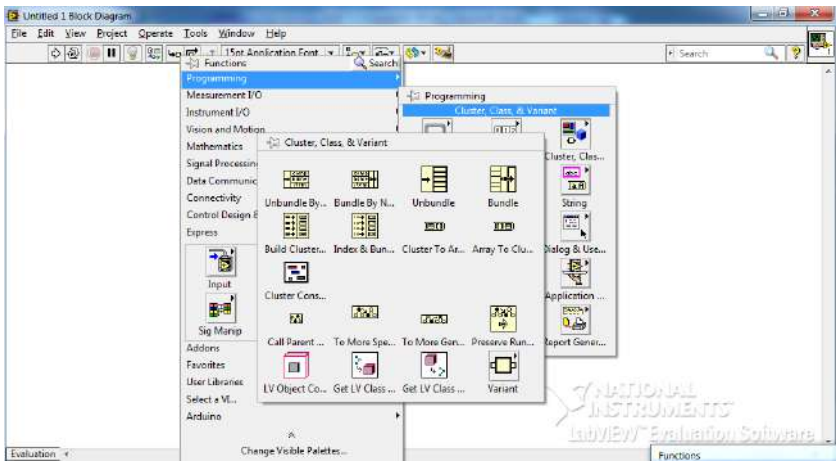
3. Klik kanan pada kaki output Bundle, dan pilih **Create Indicator** pada menu popup yang muncul. Maka sebuah Cluster telah tercipta.



Gambar 7.34 Menciptakan Cluster di Block Diagram dengan icon Bundle

7.1.6 Fungsi-Fungsi Cluster

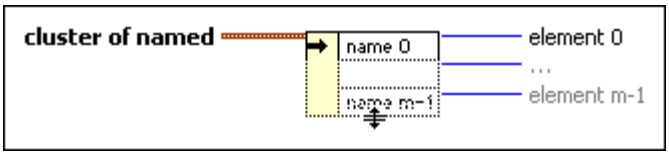
Icon Bundle yang digunakan untuk membuat Cluster di atas merupakan salah satu fungsi Cluster. Fungsi Cluster yang lain dapat dilihat dengan meng-klik kanan jendela Block Diagram, dan kemudian memilih kategori **Programming**, diikuti dengan **Cluster, Class & Variant**.



Gambar 7.35 Fungsi-fungsi Cluster

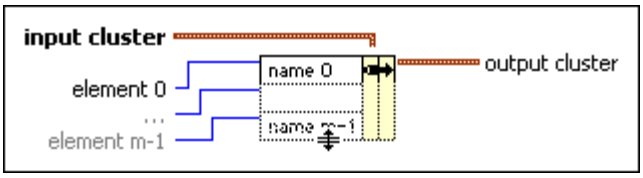
Berikut penjelasan singkat mengenai fungsi-fungsi tersebut:

1. **Unbundle by Name**. Fungsi ini digunakan untuk menguraikan Cluster menjadi komponen-komponen data individualnya dengan menampilkan nama label dari setiap komponen data tersebut.



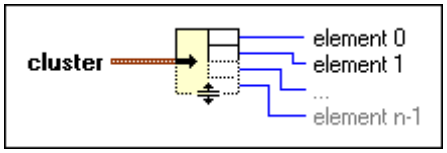
Gambar 7.36 Unbundle by Name

2. **Bundle by Name**. Fungsi ini digunakan untuk mengganti nilai komponen-komponen data pada Cluster sesuai dengan namanya.



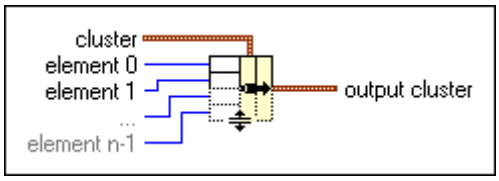
Gambar 7.37 Bundle by Name

3. **Unbundle**. Fungsi ini digunakan untuk menguraikan Cluster menjadi komponen-komponen data individualnya dengan menampilkan tipe datanya.



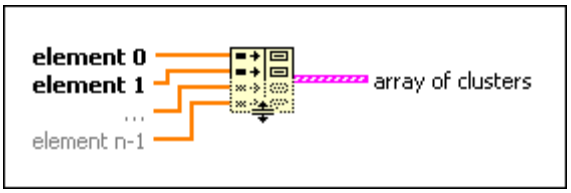
Gambar 7.38 Unbundle

4. **Bundle**. Fungsi ini digunakan untuk menyusun beberapa buah data yang dimasukkan menjadi sebuah Cluster.



Gambar 7.39 Bundle

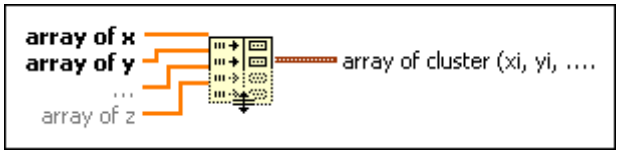
5. **Build Cluster Array**. Fungsi ini digunakan untuk mengubah elemen atau Array menjadi Cluster, kemudian menyusun Cluster-Cluster tersebut menjadi Array dari Cluster.



Gambar 7.40 Build Cluster Array

6. **Index & Bundle Cluster Array**. Fungsi ini digunakan untuk menggabungkan elemen dari beberapa Array, dari nomor indeks terkecil di semua Array, hingga nomor indeks terbesar di semua

Array, menjadi Cluster-Cluster sesuai nomor indeks, kemudian menyusun Cluster-Cluster tersebut menjadi Array dari Cluster.

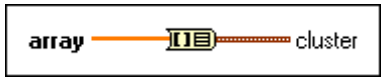


Gambar 7.41 Index & Bundle Cluster Array

7. **Cluster to Array**. Fungsi ini digunakan untuk mengubah Cluster menjadi Array 1D.

Gambar 7.42 Cluster to Array

8. **Array to Cluster**. Fungsi ini digunakan untuk mengubah Array 1D menjadi Cluster.



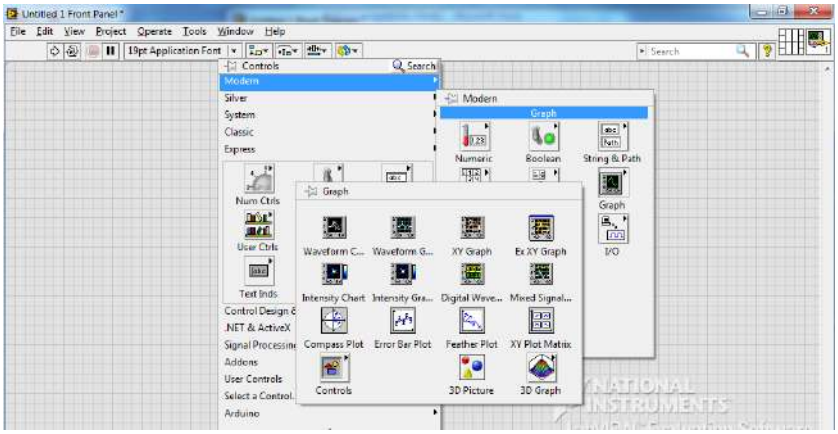
Gambar 7.43 Array to Cluster

9. **Cluster Constant**. Fungsi ini digunakan untuk menyediakan Cluster dengan komponen data berupa nilai konstan pada Block Diagram (sama seperti kotak Cluster di jendela Front Panel).

Gambar 7.44 Cluster Constant

7.2 Chart dan Graph

Sebagai bahasa pemrograman visual, LabVIEW cukup populer karena kemampuan pembuatan grafiknya yang mudah dan handal dengan bentuk grafik yang bervariasi. Pembaca dapat melihat bentuk grafik yang disediakan tersebut di palet Controls. Klik kanan jendela Front Panel hingga muncul palet **Controls**, pilih kategori **Modern**, kemudian pilih **Graph**, maka akan terlihat bermacam-macam bentuk grafik. Beberapa di antaranya adalah: Waveform Chart, Waveform Graph, XY Graph, Digital Waveform Graph, Mixed Signal Graph, 3D Graph, dll. Berikut keterangan serta penggunaan beberapa jenis grafik tersebut:

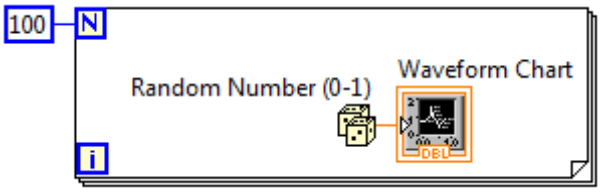


Gambar 7.45 Kategori Graph di palet Controls

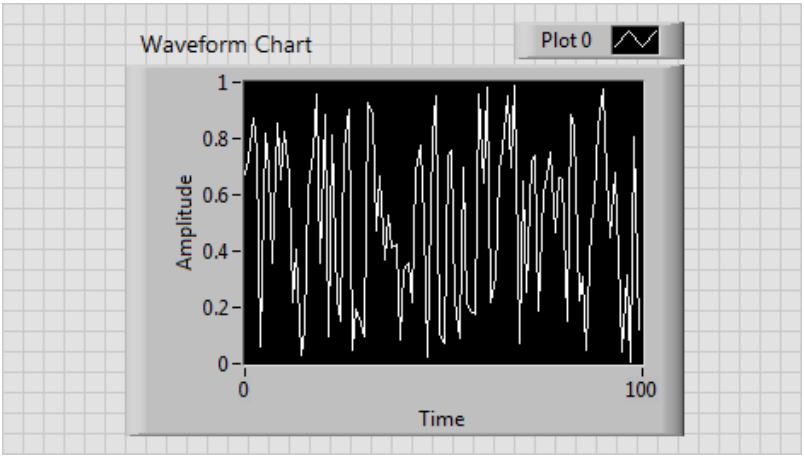
7.2.1 Waveform Chart dan Wafevorm Graph

Waveform Chart dan Waveform Graph sama-sama menampilkan data yang diterima pada laju yang konstan, namun keduanya berbeda dalam cara menampilkan dan mengupdate datanya. Waveform Graph mengumpulkan datanya dalam sebuah array dan kemudian menampilkannya dalam bentuk grafik, yang mana sama seperti pembuatan grafik di Microsoft Excel. Sebaliknya Waveform Chart membuat grafik dengan cara menambahkan titik data setiap waktu ke grafik. Agar lebih jelas, berikut ini contoh pembuatan grafik menggunakan

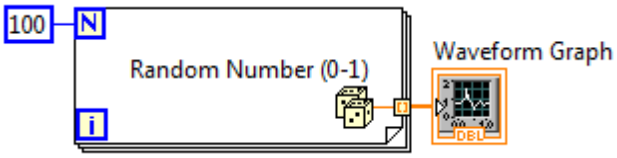
Waveform Chart dan Waveform Graph untuk data berupa 100 buah angka random:



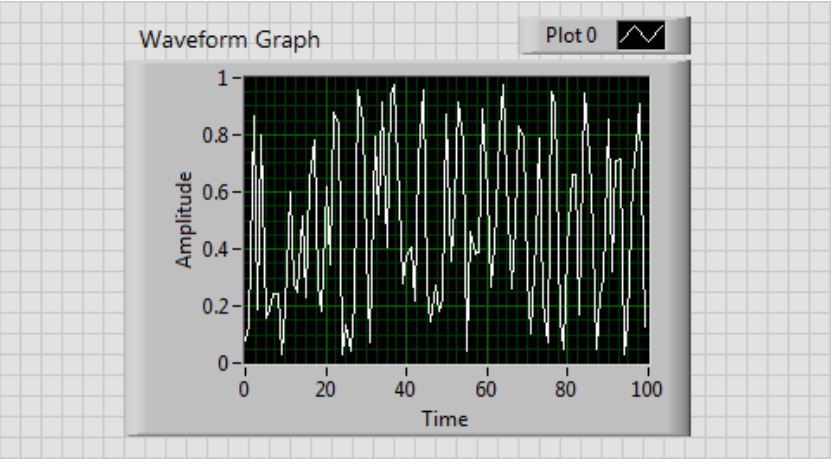
Gambar 7.46 Block Diagram Waveform Chart



Gambar 7.47 Waveform Chart diupdate setiap kali satu data diterima



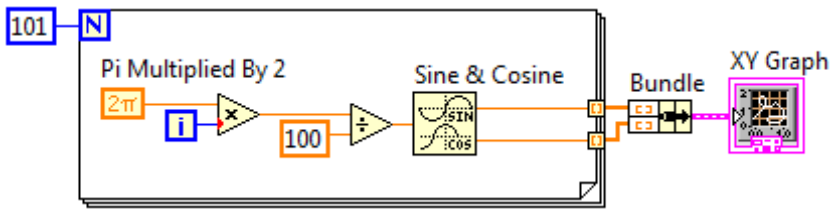
Gambar 7.48 Block Diagram Waveform Graph



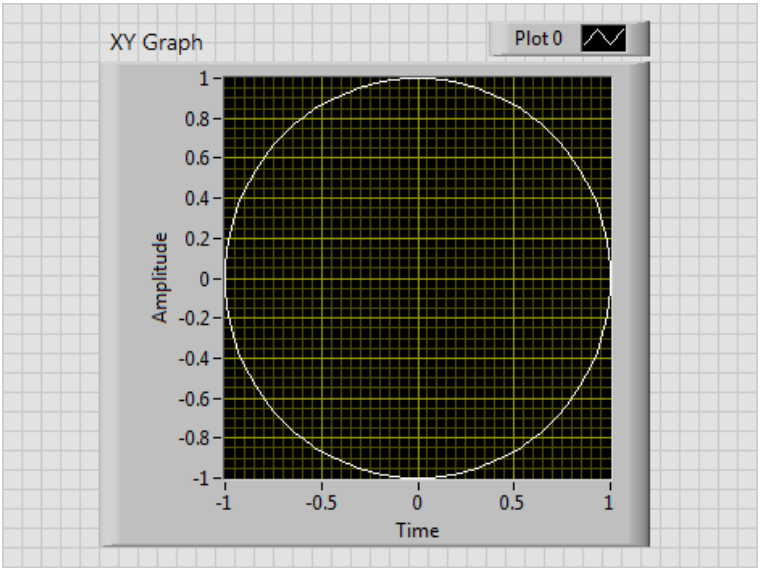
Gambar 7.49 Waveform Graph diupdate setelah semua data diterima

7.2.2 XY Graph

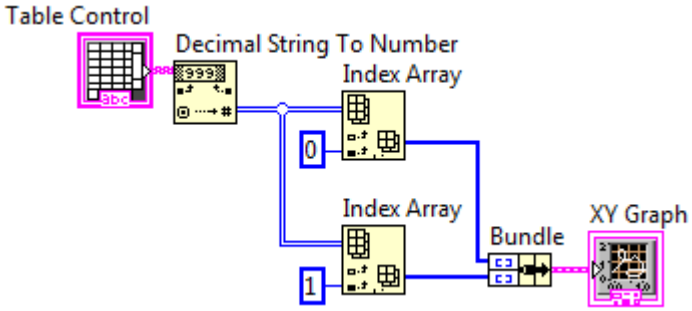
XY Graph adalah grafik Cartesian yang bisa digunakan untuk menampilkan berbagai fungsi matematika, seperti bentuk lingkaran, diagram Nyquist, Nichols, bidang S, bidang Z, dll. Berbeda dengan Waveform Chart dan Graph, XY Graph mampu menampilkan grafik yang tidak berbasis pada waktu. Di samping itu, perbedaan berikutnya adalah, XY Graph memasukkan input dengan memisahkan nilai-nilai pada koordinat sumbu X dengan sumbu Y, yang kemudian digabungkan dengan fungsi Bundle. Berikut ini berturut-turut contoh pembuatan bentuk lingkaran, segitiga dan kurve persamaan $y=25-x^2$ pada XY Graph.



Gambar 7.50 Kode di Block Diagram untuk menampilkan lingkaran

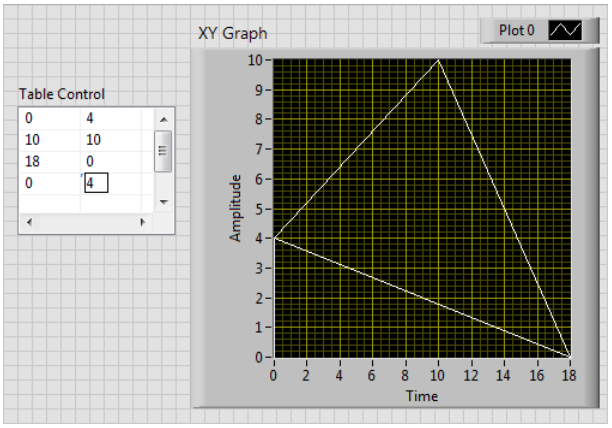


Gambar 7.51 XY Graph menampilkan lingkaran

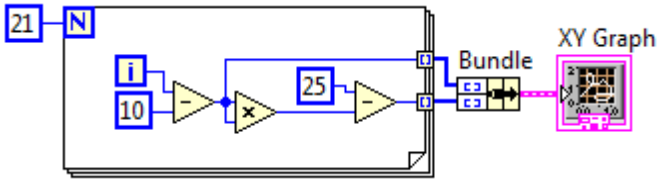


Gambar 7.52 Kode di Block Diagram untuk menampilkan segitiga

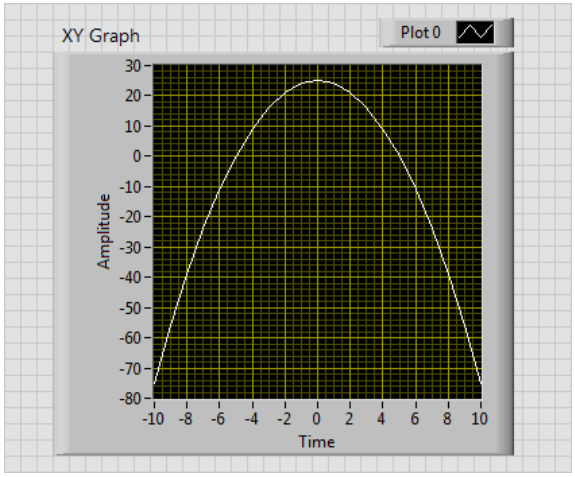
Perhatikan Block Diagram pada gambar 7.52 di atas, untuk membuat bentuk segitiga, digunakan bantuan tabel untuk memasukkan posisi atau titik koordinat ujung-ujung segitiga. Karena tipe data pada tabel adalah Array dari String, maka dibutuhkan konversi dari String ke Angka Desimal. Selanjutnya untuk memisahkan nilai koordinat sumbu X dengan sumbu Y, digunakan fungsi Index Array.



Gambar 7.53 XY Graph menampilkan segitiga



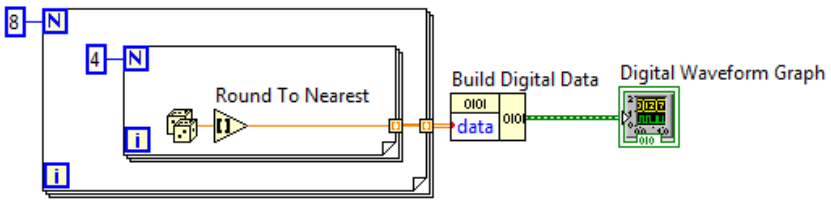
Gambar 7.54 Kode di Block Diagram untuk menampilkan $y=25-x^2$



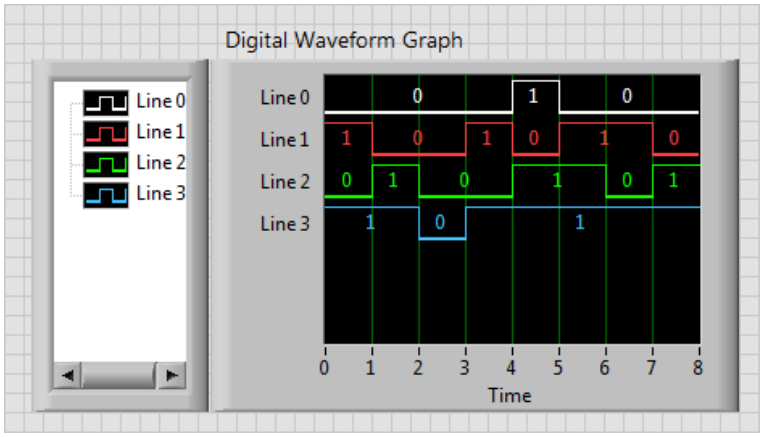
Gambar 7.55 XY Graph menampilkan kurve $y=25-x^2$

7.2.3 Digital Waveform Graph

Digital Waveform Graph menampilkan data sebagai pulsa atau kumpulan garis digital (*high* dan *low*). Berikut contoh pembuatan grafik data digital (dari angka random) menggunakan Digital Waveform Graph.



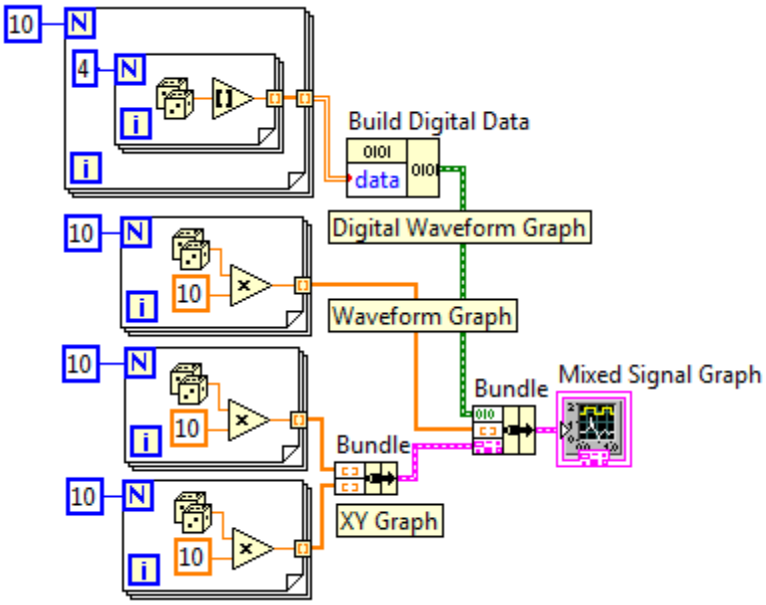
Gambar 7.56 Kode di Block Diagram untuk menampilkan data digital



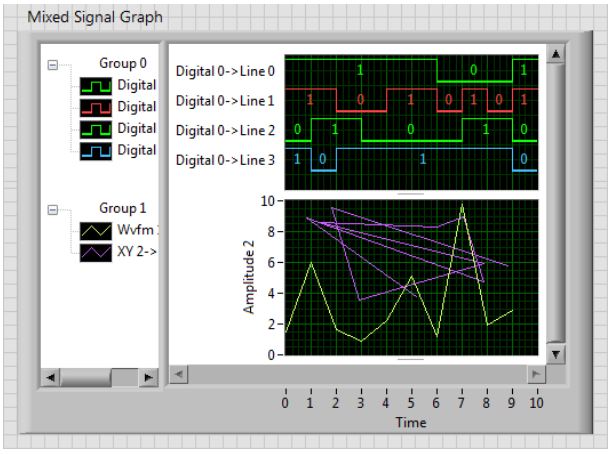
Gambar 7.57 Menampilkan data digital pada Digital Waveform Graph

7.2.4 Mixed Signal Graph

Sesuai dengan namanya, Mixed Signal Graph digunakan untuk menampilkan semua tipe grafik, baik Waveform Graph, XY Graph maupun Digital Waveform Graph dalam satu area secara bertingkat, dengan tujuan melihat dan membandingkan semua data pada masing-masing grafik tersebut pada basis waktu yang sama. Berikut ini contoh penggunaan Mixed Signal Graph untuk menampilkan 3 tipe grafik.



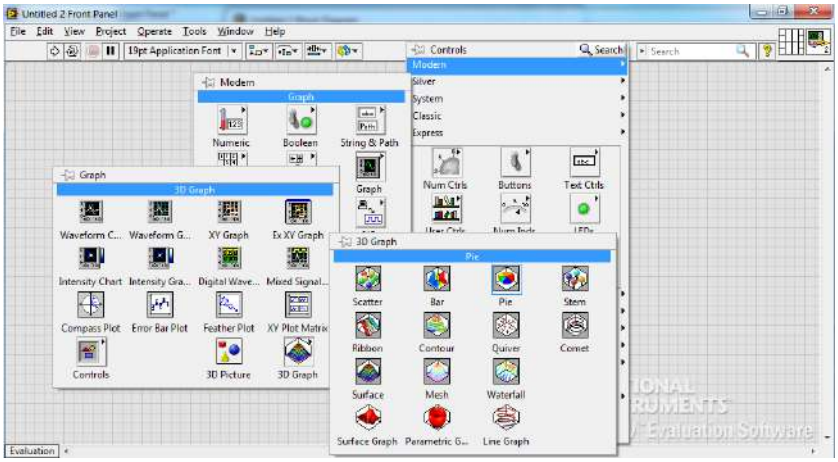
Gambar 7.58 Kode di Block Diagram untuk menampilkan 3 tipe grafik



Gambar 7.59 Mixed Signal Graph menampilkan 3 tipe grafik

7.2.5 3D Graph

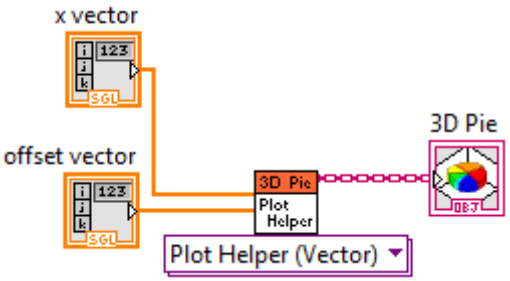
Dengan 3D Graph ini, pembaca dapat memvisualisasikan data dalam 3 dimensi. LabVIEW menyediakan berbagai jenis 3D Graph, yang dapat diambil dari palet Controls di Front Panel, yaitu di kategori Modern, Graph dan kemudian 3D Graph.



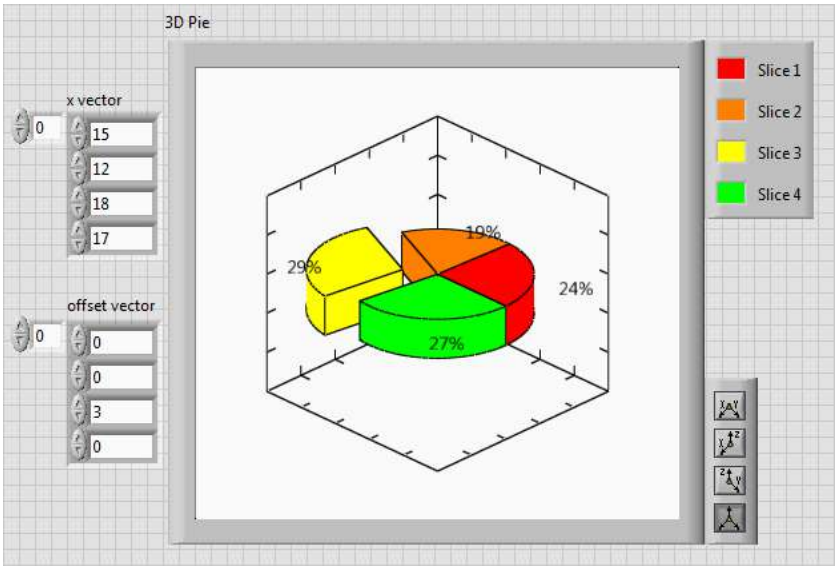
Gambar 7.60 Berbagai jenis 3D Graph yang tersedia

Sebagai contoh, berikut ini diperlihatkan penggunaan 3D Graph jenis Pie Chart untuk menampilkan prosentase dari hasil perolehan suara setiap calon pada pemilihan ketua kelas, dengan data seperti ditunjukkan dalam tabel berikut ini:

No.	Nama	Suara
1.	Amir	15
2.	Arum	12
3.	Anjas	18
4.	Asri	17



Gambar 7.61 Kode di Block Diagram untuk grafik Pie Chart



Gambar 7.62 Front Panel untuk grafik Pie Chart

Keterangan untuk kode di Block Diagram, **x vector** adalah data yang akan ditampilkan, sedangkan **offset vector** adalah berapa jauh jarak potongan data tersebut terhadap titik pusatnya, seperti terlihat pada gambar 7.62.

BAB 8
STRING, FILE EXE DAN FILE I/O

8.1 String

Kembali ke aplikasi pembuatan simulasi kalkulator pada bab 6, di mana kalkulator memiliki banyak keterbatasan, di antaranya adalah belum bisa memasukkan angka lebih dari 1 digit, dan juga angka pecahan, karena tombol titik belum berfungsi. Untuk itu, pada bab ini, dengan bantuan String, permasalahan tersebut dapat diatasi. Apa itu String?

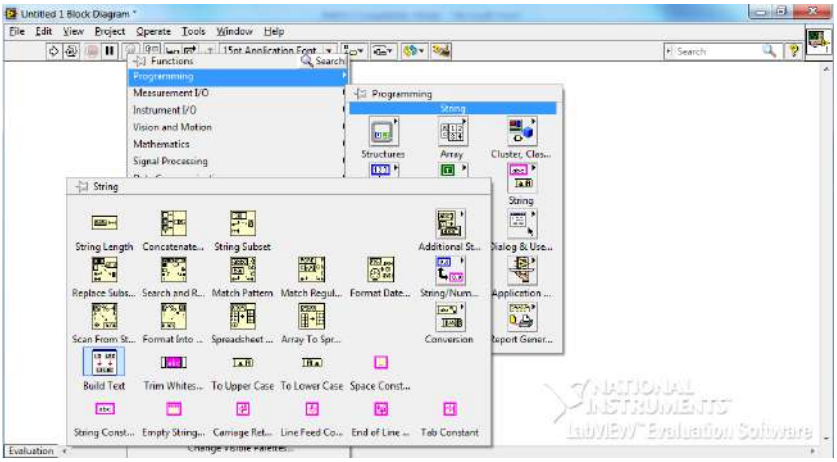
String adalah susunan karakter-karakter ASCII, baik yang dapat ditampilkan (*displayable*) maupun yang tidak dapat ditampilkan (*non-displayable*).

Beberapa manfaat String adalah sebagai berikut:

1. Memberikan informasi berbentuk teks yang jelas.
2. Merupakan kode standar untuk komunikasi data secara serial.
3. Sebagai format data untuk pengolahan file, seperti pembacaan dan penyimpanan file.

8.1.1 Fungsi-fungsi String

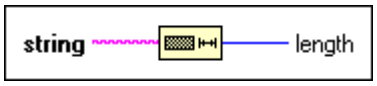
LabVIEW memiliki banyak fungsi untuk memanipulasi string. Untuk memunculkannya, klik kanan jendela Block Diagram untuk membuka palet Functions, pilih **Programming**, kemudian pilih **String**.



Gambar 8.1 Fungsi-fungsi String

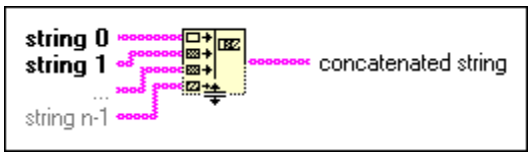
Berikut penjelasan singkat beberapa fungsi tersebut:

1. **String Length**. Fungsi ini digunakan untuk menghitung panjang karakter dalam sebuah String.

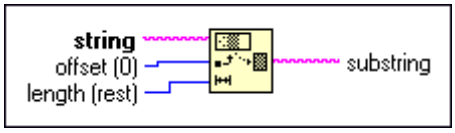


Gambar 8.2 String Length

2. **Concatenate String**. Fungsi ini digunakan untuk menggabungkan semua karakter atau string di inputnya menjadi sebuah string.
3. **String Subset**. Fungsi ini digunakan untuk menghasilkan irisan string yang dimulai dari nilai offset hingga sepanjang nilai length.

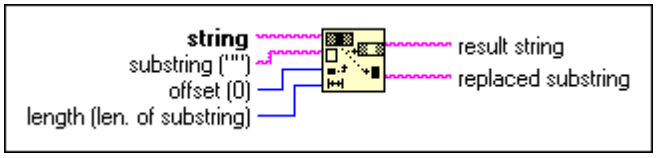


Gambar 8.3 Concatenate String



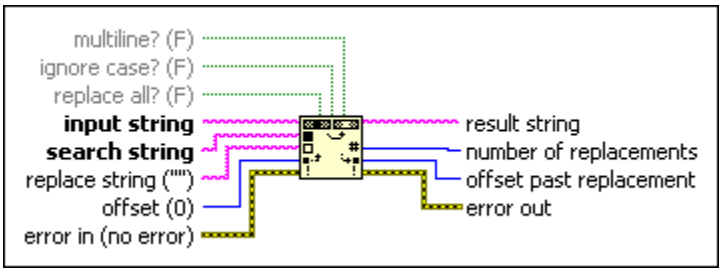
Gambar 8.4 String Subset

4. **Replace Substring**. Fungsi ini digunakan untuk menggantikan beberapa karakter di dalam string dengan karakter yang baru yang dimulai dari nilai offset hingga sepanjang nilai length.



Gambar 8.5 Replace Substring

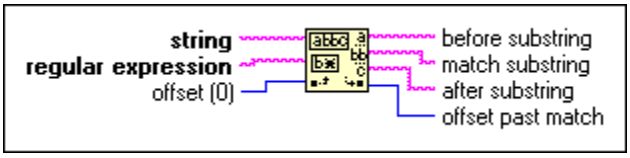
4. **Search and Replace String**. Fungsi ini digunakan untuk mencari karakter atau string dan menggantikannya dengan string yang baru.



Gambar 8.6 Search and Replace String

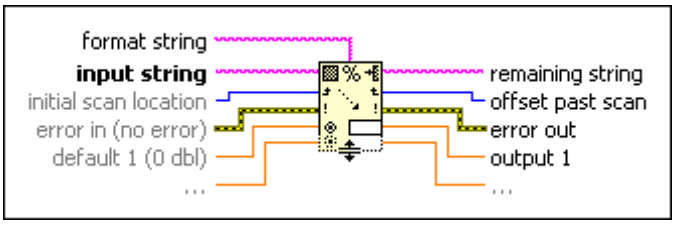
5. **Match Pattern**. Fungsi ini mencari karakter atau substring pada sebuah string, dimulai dari nilai offset, dan jika ditemukan, maka

string tersebut dibagi menjadi 3, yaitu sebelum substring, pada substring, sesudah substring dan jumlah karakter dari offset ke substring tersebut dapat diketahui.



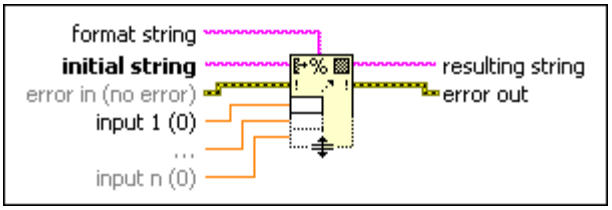
Gambar 8.7 Match Pattern

6. **Scan From String**. Fungsi ini men-scan input String dan mengubah ke data angka/Numerik sesuai dengan format yang dimasukkan.



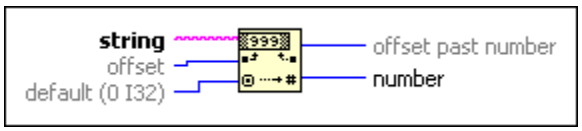
Gambar 8.8 Scan From String

7. **Format Into String**. Fungsi ini mengubah input data angka/Numerik menjadi String sesuai dengan format yang dimasukkan.



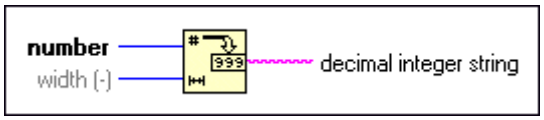
Gambar 8.9 Format Into String

8. **Decimal String to Number**. Fungsi ini digunakan untuk mengubah karakter angka di dalam String menjadi angka Desimal Integer.



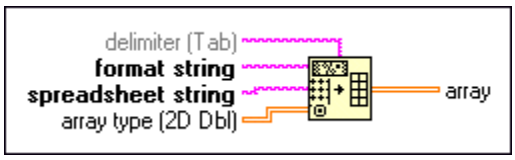
Gambar 8.10 Decimal String to Number

9. **Number to Decimal String**. Fungsi ini digunakan untuk mengubah angka menjadi String dari angka Desimal Integer.



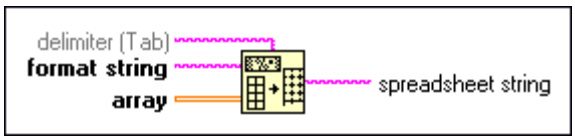
Gambar 8.11 Number to Decimal String

10. **Spreadsheet String to Array**. Fungsi ini mengubah Tabel String menjadi Array.



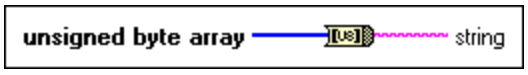
Gambar 8.12 Spreadsheet String to Array

11. **Array to Spreadsheet String**. Fungsi ini mengubah Array menjadi Tabel String.



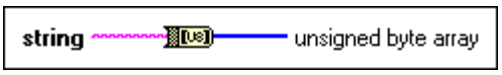
Gambar 8.13 Array to Spreadsheet String

12. **Byte Array to String**. Fungsi ini mengubah Array Byte yang merepresentasikan karakter ASCII menjadi sebuah String.



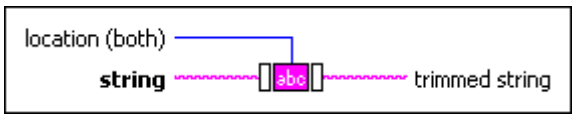
Gambar 8.14 Byte Array to String

13. **String to Byte Array**. Fungsi ini mengubah String menjadi Array. String dalam hal ini berupa kode angka, contoh kode ASCII a = 97.



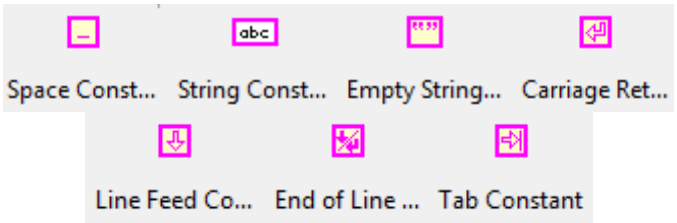
Gambar 8.15 String to Byte Array

14. **Trim Whitespace**. Fungsi ini menghilangkan semua whitespace ASCII (seperti Tab, Space, Enter, LF) yang berada di depan string, di belakang atau di kedua tempat tersebut.



Gambar 8.16 Trim Whitespace

15. **Konstanta-konstanta String**. Fungsi ini digunakan untuk memberikan kode ASCII atau karakter tertentu ke Block Diagram.



Gambar 8.17 Konstanta-konstanta String

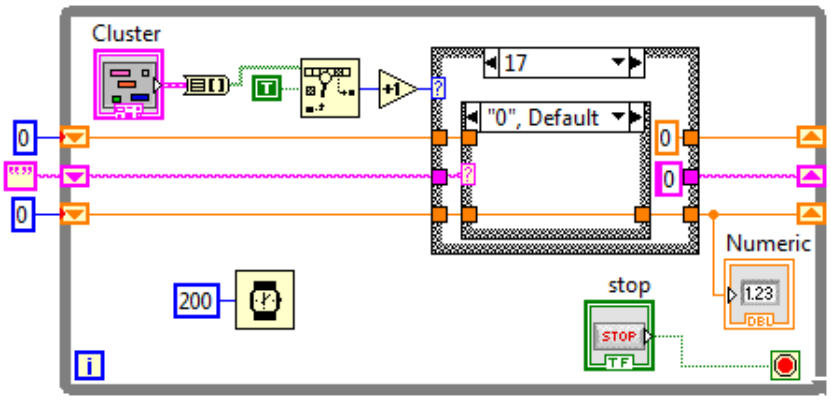
8.1.2 Aplikasi String

Agar lebih jelas mengenai pemakaian dan fungsi String, maka berikut ini disajikan contoh aplikasi String untuk menyempurnakan pembuatan simulasi kalkulator. Seperti telah disebutkan di awal, kalkulator yang dibuat masih memiliki beberapa keterbatasan, di antaranya adalah belum bisa memasukkan angka yang lebih dari 1 digit dan juga angka pecahan.

Kedua masalah ini akan dapat diatasi dengan mudah menggunakan salah satu fungsi String, yaitu **Concatenate String**.

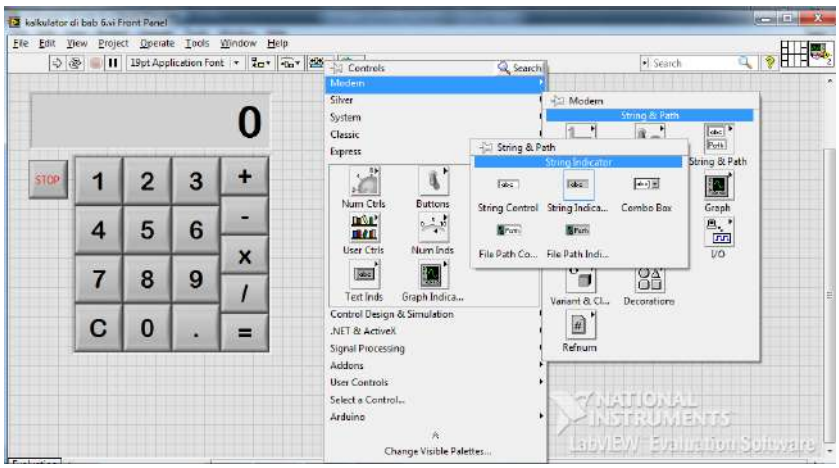
Dengan **Concatenate String**, karakter-karakter dapat digabungkan dengan mudah, termasuk tanda titik. Namun syaratnya tentu saja, tipe datanya harus String. Untuk itu perlu pengubahan tipe data dari Numerik menjadi String menggunakan fungsi **Format Into String**. Kemudian dilakukan penggabungan dengan **Concatenate String**. Hasil penggabungan diubah lagi dari String menjadi Numerik dengan menggunakan fungsi **Scan From String**. Berikut langkah-langkahnya:

1. Buka program kalkulator yang telah dibuat di Bab 6, dengan Block Diagram seperti ditunjukkan pada gambar berikut:

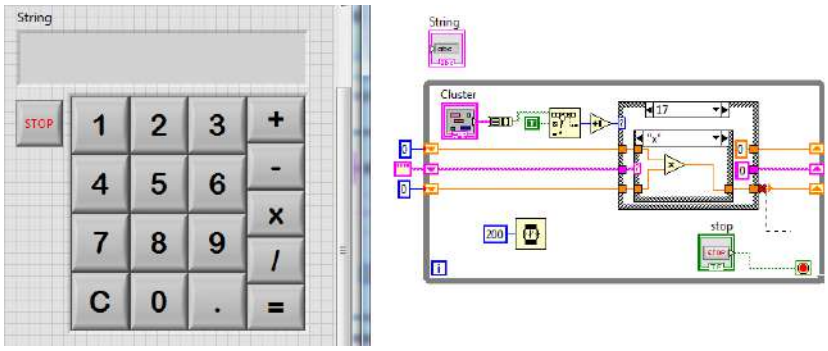


Gambar 8.18 Kode Block Diagram pada gambar 6.52

2. Karena tampilan kalkulator harus menampilkan hasil penggabungan karakter angka dan titik, maka tampilan tersebut harus diubah dari tipe data Numeric (**Num Ind**) menjadi tipe data String (**String Indicator**). Untuk itu, hapus objek **Num Ind** dan ambil objek **String Indicator** dari palet Controls dan tempatkan di jendela Front Panel.

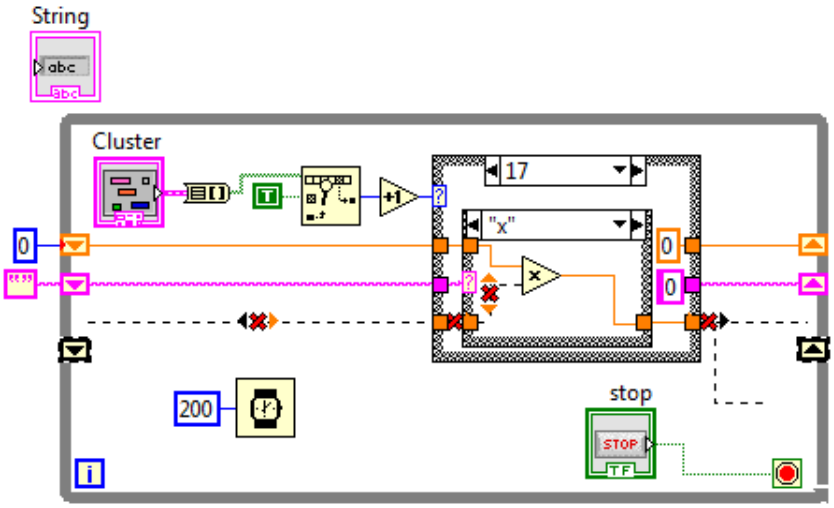


Gambar 8.19 Mengambil objek String Indicator dari palet Controls



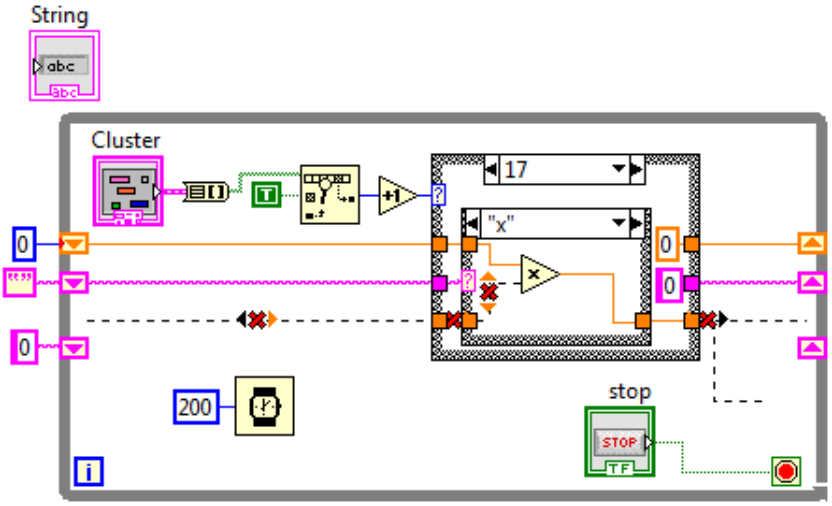
Gambar 8.20 Hapus Num Ind, dan ganti dengan String Indicator

3. Hapus pula **Shift Register** ketiga yang bertipe data **Numeric Integer**, kemudian tambahkan **Shift Register** baru.



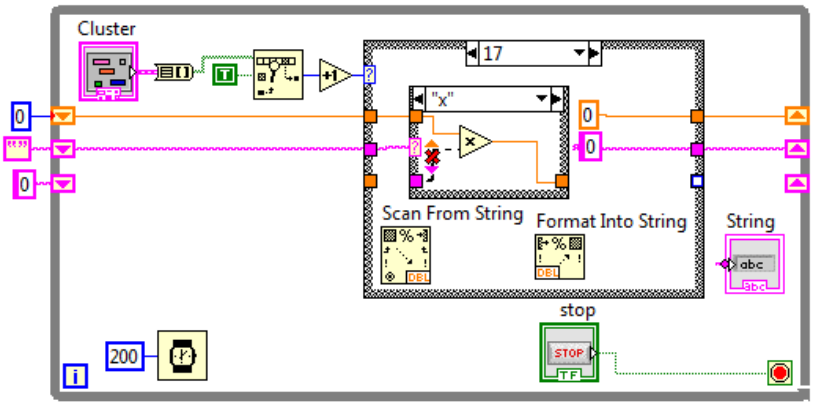
Gambar 8.21 Hapus Shift Register ketiga, tambahkan Shift Register baru

4. Beri nilai awal **Shift Register** ketiga yang baru ini dengan **String Constant**, dan isi kotaknya dengan nilai **0**.



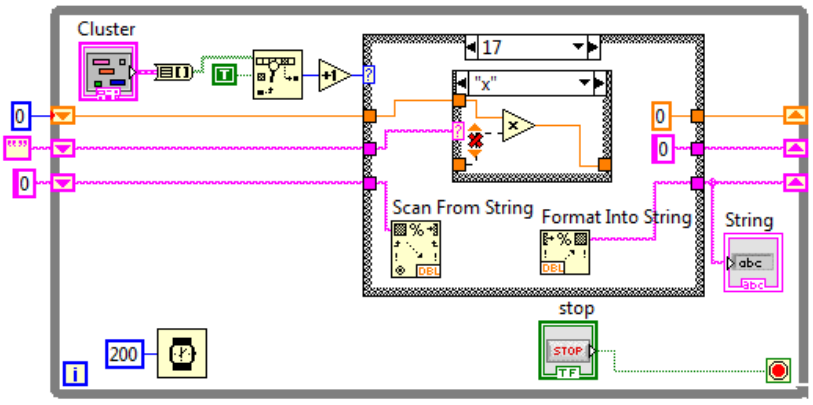
Gambar 8.22 Beri nilai String Constant = 0 pada Shift Register ketiga

5. Masukkan **String Indicator** di dalam While Loop. Kemudian tambahkan fungsi **Scan From String** dan **Format Into String**, dan tempatkan di blok **Case 17** seperti terlihat pada gambar berikut ini:



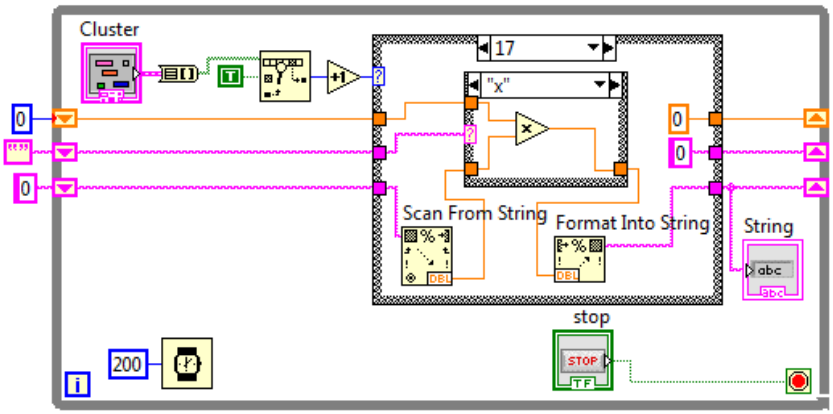
Gambar 8.23 Memasukkan Scan From String dan Format Into String di blok Case 17 dan String Indicator di dalam While Loop

6. Hubungkan terminal kiri **Shift Register** ketiga dengan fungsi **Scan From String**, dan terminal kanan **Shift Register** ketiga dengan **Format Into String** dan **String Indicator.**



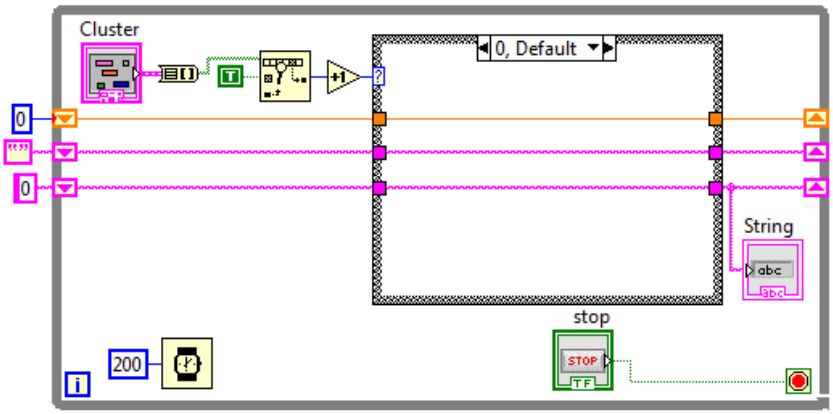
Gambar 8.24 Menghubungkan Scan From String, Format Into String dan String Indicator dengan Shift Register ketiga

7. Hubungkan output **Scan From String** ke input blok Case perhitungan **(+,-,x,/)** dan output blok Case tersebut ke input **Format Into String**.



Gambar 8.25 Menghubungkan blok Case operasi perhitungan (+,-,x,/) dengan output Scan From String dan input Format Into String

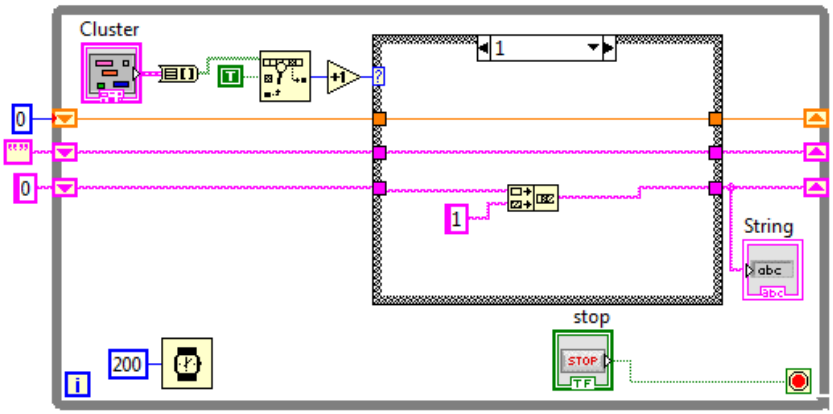
8. Untuk blok **Case 0, Default**, teruskan garis dari terminal kiri **Shift Register** ketiga ke terminal kanan Shift Register tersebut.



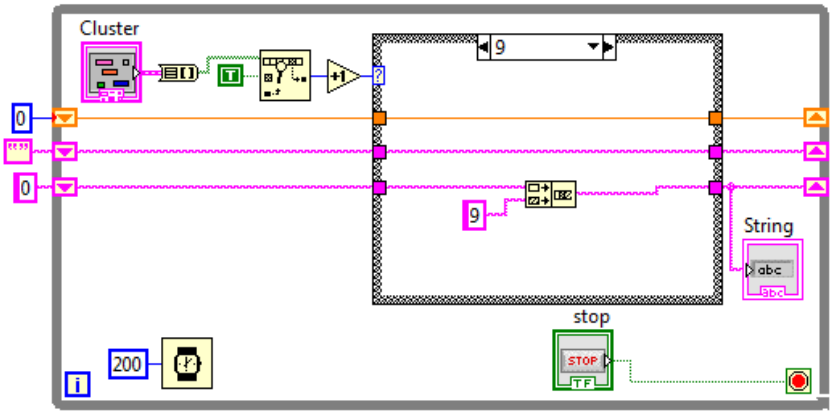
Gambar 8.26 Meneruskan data dari terminal kiri ke terminal kanan Shift Register ketiga melalui blok Case 0 Default

9. Kemudian untuk blok **Case 1 – 9** (tombol angka 1 – 9), tambahkan fungsi **Concatenate String**, dengan kaki input 1 fungsi tersebut

terhubung dengan terminal kiri Shift Register ketiga dan kaki input 2 terhubung dengan angka yang bersesuaian dengan blok Case-nya.

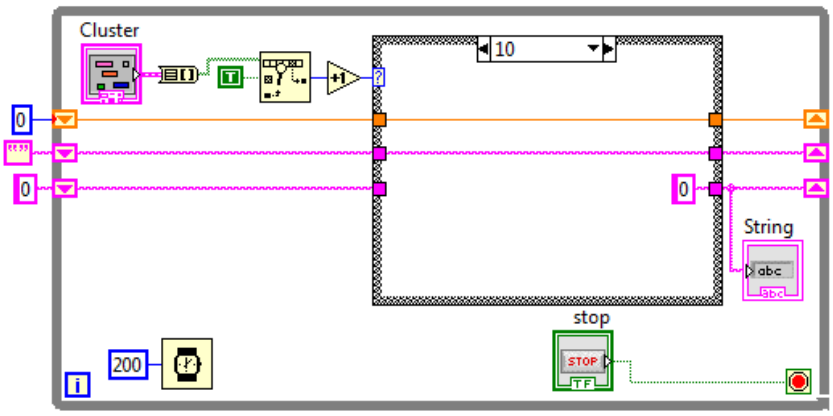


Gambar 8.27 Menambahkan Concatenate String di blok Case 1-9, dengan kaki input 1 dengan terminal kiri Shift Register dan kaki input 2 dengan angka yang bersesuaian dengan blok Case-nya



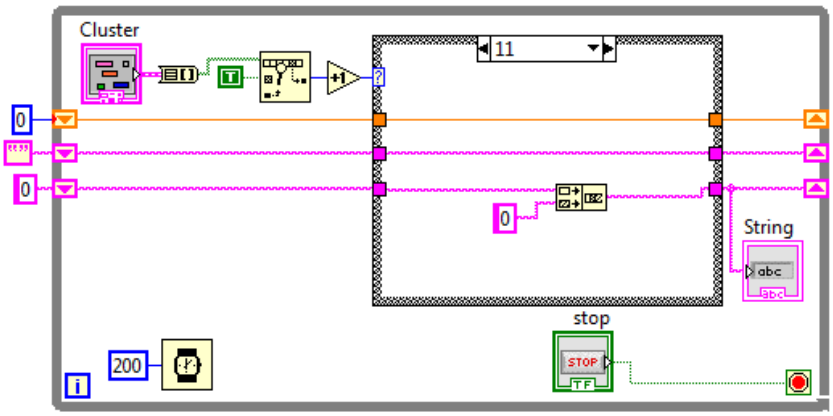
Gambar 8.28 Menambahkan Concatenate String di blok Case 1-9, dengan kaki input 1 dengan terminal kiri Shift Register dan kaki input 2 dengan angka yang bersesuaian dengan blok Case-nya

10. Sedangkan untuk blok **Case 10** (tombol C), beri nilai 0 pada terminal keluaran, karena tampilan harus 0 setiap kali **tombol C** ini ditekan

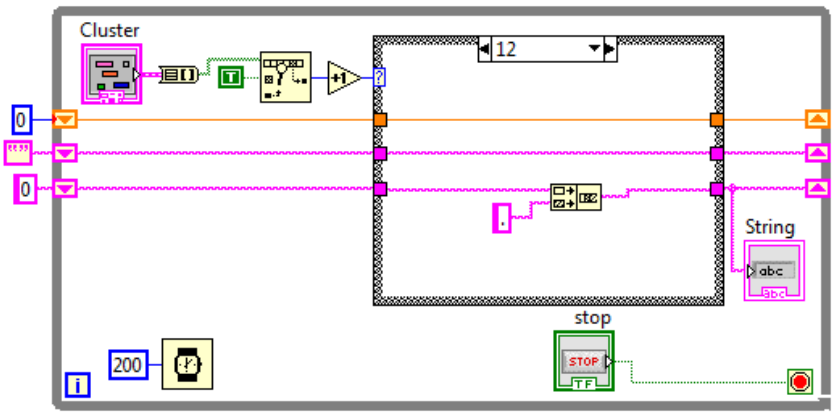


Gambar 8.29 Memberi nilai 0 ketika tombol C (blok Case 10) ditekan

11. Untuk blok **Case 11** (tombol 0) dan **Case 12** (tombol titik .), karena keduanya harus digabungkan sama seperti angka, maka tambahkan pula fungsi **Concatenate String** untuk kedua blok Case tersebut.

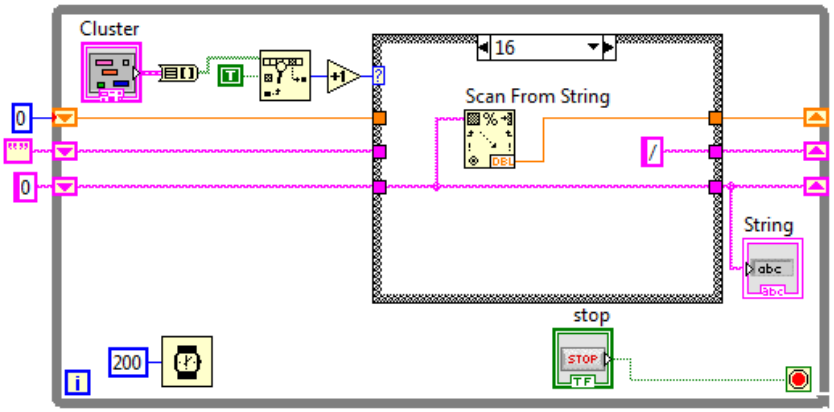


Gambar 8.30 Menambah Concatenate String di tombol 0 (blok Case 11)



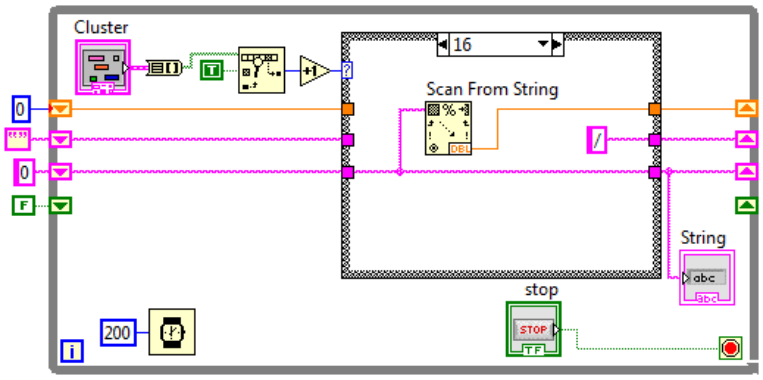
Gambar 8.31 Menambah Concatenate String di tombol . (blok Case 12)

12. Kemudian untuk blok **Case 13 – 16** (blok tombol operasi perhitungan +,-,x,/), setiap kali penekanan salah satu tombol tersebut, maka data tidak hanya ditampilkan, tetapi juga harus disimpan ke Shift Register pertama, agar bisa diambil kembali ketika **tombol =** ditekan. Karena tipe data Shift Register pertama adalah **Numeric**, maka data dari Shift Register ketiga harus diubah dari **String** menjadi **Numeric**.



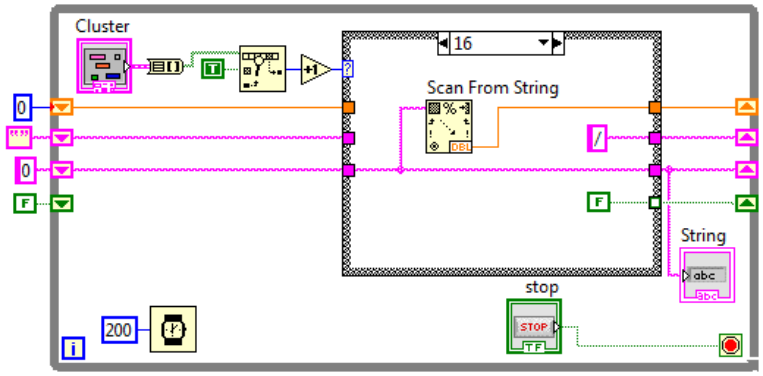
Gambar 8.32 Untuk blok Case 13-16, data tidak hanya diteruskan ke Shift Register ketiga, tetapi juga disimpan ke Shift Register pertama, dengan Scan From String yang mengubah data String ke Numeric

13. Karena fungsi **Concatenate String** harus dimatikan ketika **tombol operasi perhitungan (+,-,x,/)**, **tombol C** dan **tombol =** ditekan, dan kemudian dihidupkan ketika **tombol angka (0-9)** dan **tombol titik (.)** ditekan, maka tambahkan **Shift Register** keempat yang akan mengatur fungsi tersebut dan beri nilai awal dengan nilai **False (F)**.



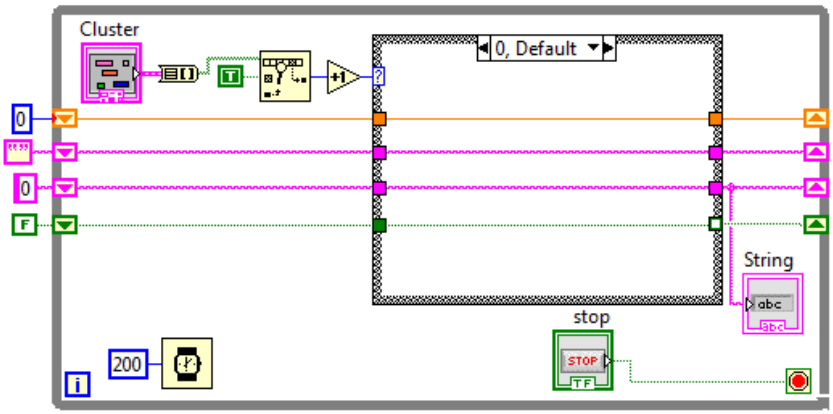
Gambar 8.33 Menambah Shift Register keempat dengan nilai awal False

14. Kemudian untuk blok **Case 10** (tombol C), blok **Case 13-16** (tombol +,-,x,/) dan blok **Case 17** (tombol =), beri nilai **False** dari dalam bloknya masing-masing ke terminal kanan Shift Register keempat.



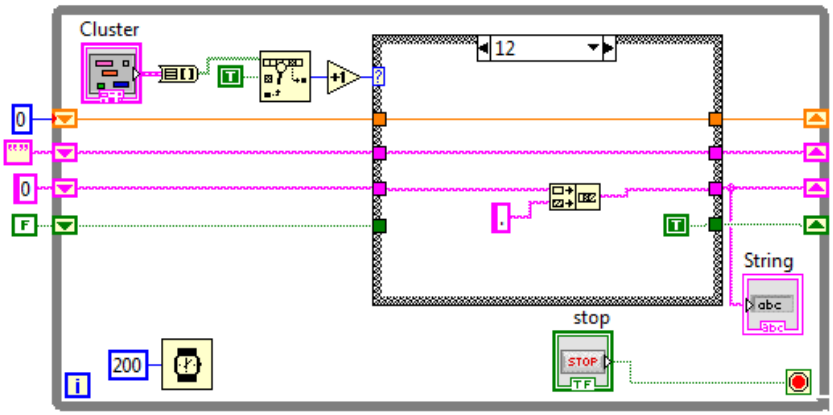
Gambar 8.34 Untuk blok Case 10, 13-17, beri nilai False ke Shift Register

15. Untuk blok **Case 0, Default**, teruskan data dari terminal kiri ke terminal kanan Shift Register keempat ini.



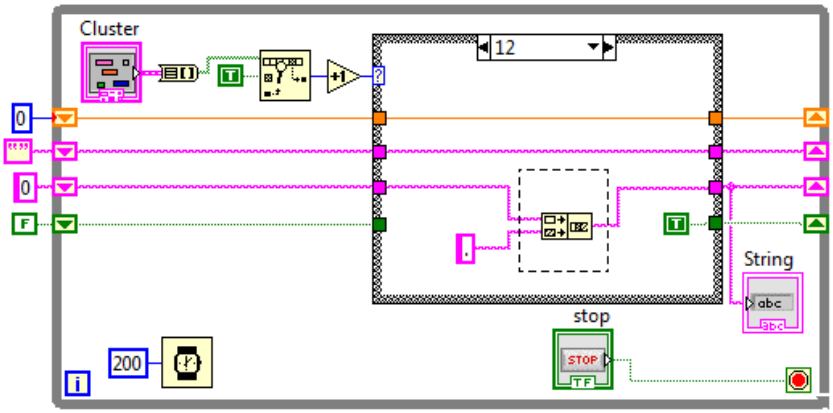
Gambar 8.35 Untuk blok Case 0, Default, teruskan data dari kiri ke kanan

16. Untuk blok **Case 1-9** (tombol angka 1-9), blok **Case 11** (tombol 0), dan blok **Case 12** (tombol .), beri nilai dari dalam bloknya masing-masing nilai **True**, dan hubungkan ke terminal kanan Shift Register.



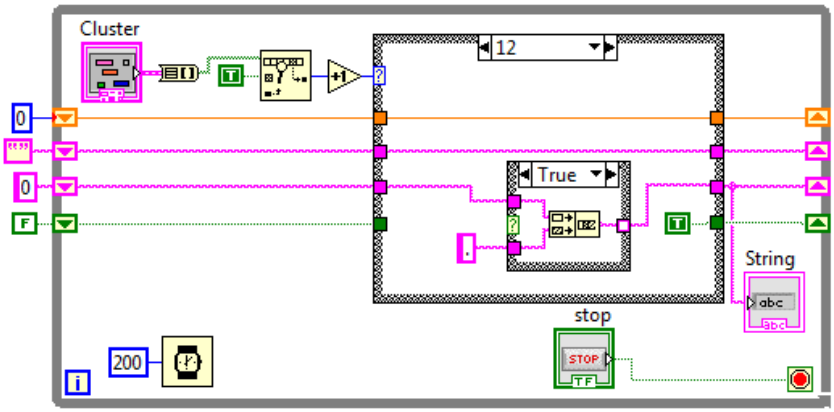
Gambar 8.36 Untuk blokCase 1-9, 11, 12, beri nilai True dari dalam bloknya masing-masing ke terminal kanan Shift Register keempat

17. Tambahkan **Case Structure** pada fungsi **Concatenate String**.

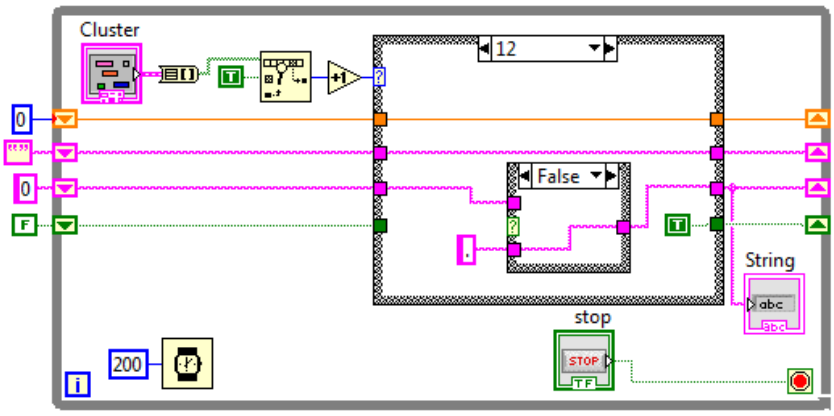


Gambar 8.37 Lingkupi fungsi Concatenate String dengan Case Structure

18. Untuk label blok **Case True**, seharusnya fungsi **Concatenate String** tampak, sedangkan untuk label blok **Case False**, seharusnya tidak ada fungsi **Concatenate String**, sebagai gantinya, input karakter diteruskan ke output blok Casenya. Dengan cara ini, ketika **terminal selector (?)** diberi nilai **True**, maka fungsi Concatenate String hidup, sebaliknya ketika diberi nilai False, maka fungsi tersebut dimatikan.

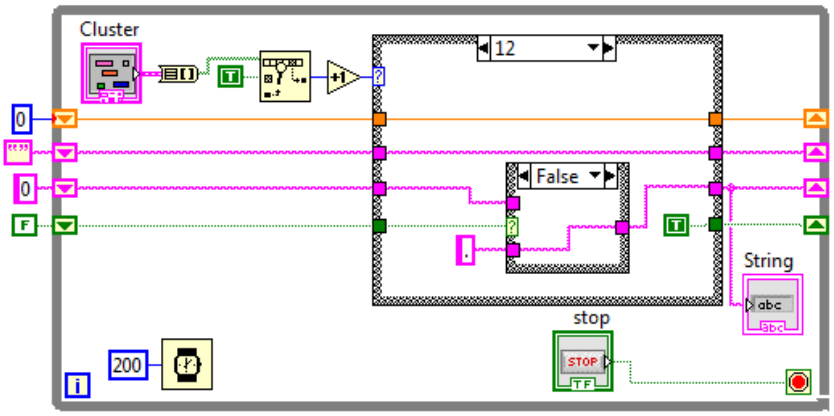


Gambar 8.38 Pada label True, fungsi Concatenate String tampak



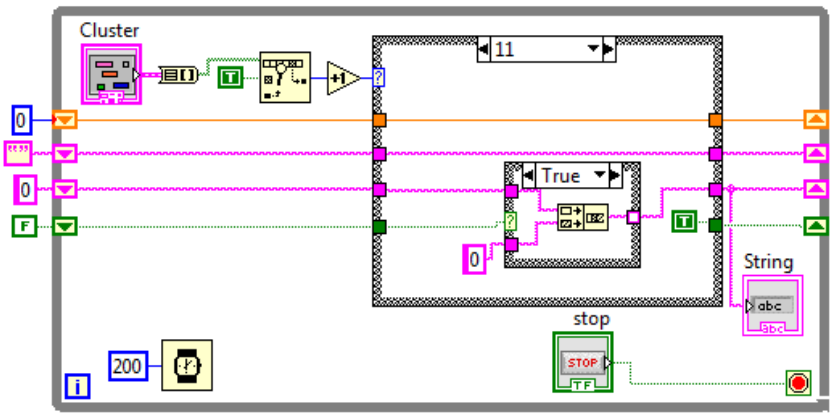
Gambar 8.39 Pada label False, tidak ada Concatenate String, sebagai gantinya, hubungkan input karakter angka/titik ke output blok Case

19. Maka, agar fungsi Concatenate String bisa dihidupkan dan dimatikan, hubungkan **terminal selector (?)** dengan input dari terminal kiri **Shift Register** keempat.



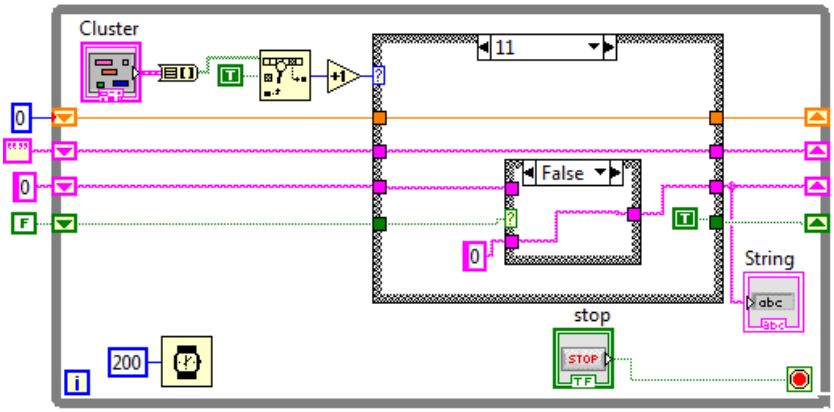
Gambar 8.40 Hubungkan terminal selector dengan input Shift Register

20. Karena fungsi **Concatenate String** tidak hanya di blok **Case 12**, tetapi juga di blok **Case 1-9** (tombol 1-9) dan blok **Case 10** (tombol 0), maka ulangi langkah-langkah no. 17 – 19 untuk setiap blok tersebut.



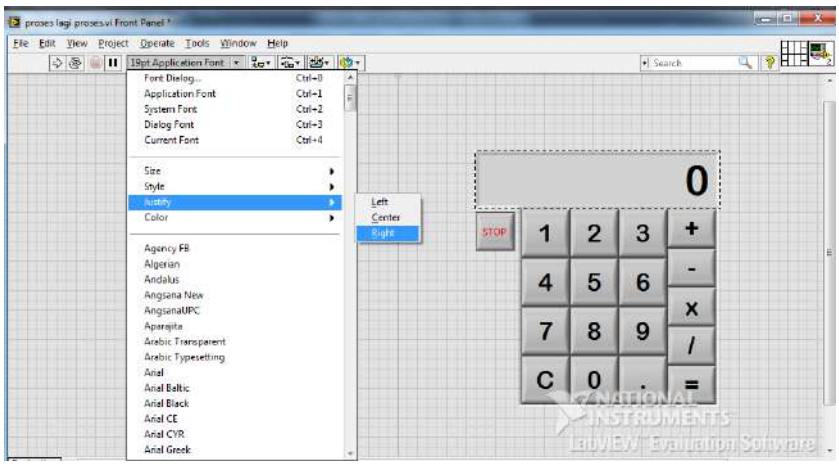
Gambar 8.41 Mengulangi langkah 17-19 untuk blok Case 1-9 dan 11

21. Jangan lupa untuk menghubungkan input karakter di kiri blok dengan output di kanan blok di setiap label **False**.



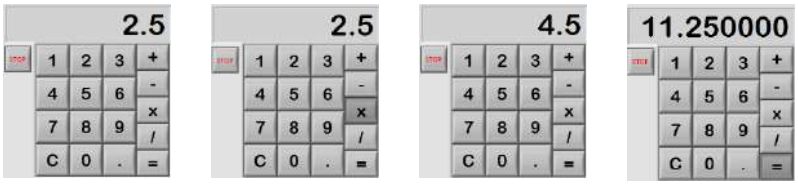
Gambar 8.42 Untuk label False, hubungkan input karakter di kiri blok dengan output di kanan blok Case

22. Terakhir, pada tampilan kalkulator di jendela Front Panel, perbesar ukuran teks di dalam **String Indicator** dengan menekan **tombol Ctrl** dan **tombol =** bersama-sama, dan buat teks tersebut rata kanan dengan *Justify Right,* serta hilangkan **Label String** di **Visible Items.**



Gambar 8.43 Memperbesar ukuran teks, membuat rata kanan dengan Justify Right, dan menghilangkan Label String pada tampilan kalkulator

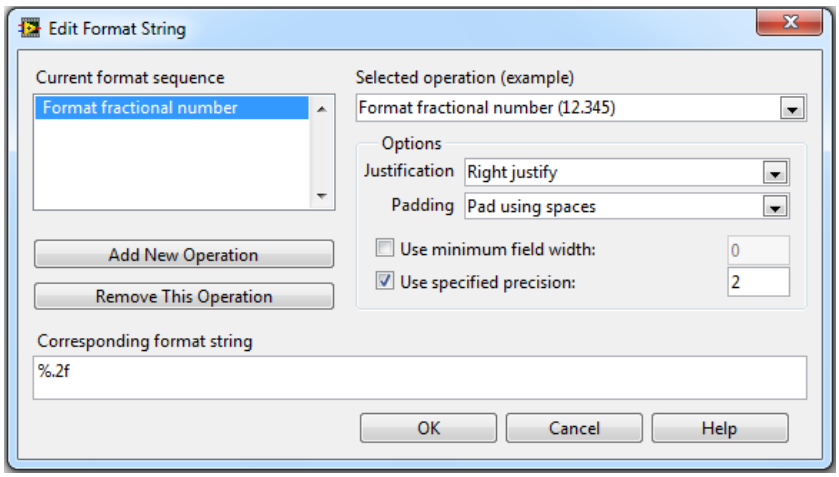
23. Jalankan program dengan menekan tombol **Run**. Buatlah perkalian antara angka 2,5 dengan 4,5, seharusnya hasilnya adalah 11,25.



(a) Angka 2,5 (b) tombol x (c) Angka 4,5 (d) Hasil 11,25

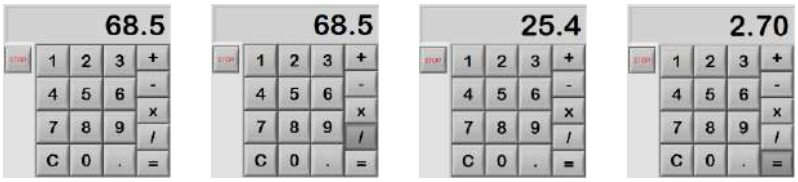
Gambar 8.44 Perkalian 2,5 dengan 4,5 menghasilkan 11,25

24. Tampak hasil perkalian pada tampilan memiliki 6 digit angka di belakang koma. Untuk menguranginya hingga hanya 2 digit angka di belakang koma, maka buka jendela Block Diagram. Klik 2 kali pada fungsi **Format Into String** di blok **Case 17**. Pada kotak dialog yang terbuka, beri nilai **2** pada opsi **Use specified precision (a**pabila tidak dberi nilai, maka secara default nilainya adalah 6 digit).



Gambar 8.45 Beri angka 2 pada opsi Use specified precision

25. Jalankan program sekali lagi, dan lakukan pembagian antara angka 68,5 dengan 25,4. Perhatikan bahwa sekarang hasil perhitungannya memiliki nilai 2 digit angka di belakang koma, seperti berikut ini.



(a) Angka 68,5 (b) tombol / (c) Angka 25,4 (d) Hasil 2,70

Gambar 8.46 Hasil perhitungan memiliki 2 digit angka di belakang koma

26. Hanya sampai di sini contoh aplikasi String untuk penyempurnaan kalkulator, yang membuat kalkulator dapat melakukan perhitungan dengan 2 buah angka yang lebih dari 1 digit dan angka-angka pecahan desimal (dengan koma). Masih banyak manfaat dan aplikasi String lainnya. Pada bab 9 akan diperlihatkan penggunaan format data String dalam komunikasi serial, yang mana menjadi pokok bahasan utama dalam buku ini.

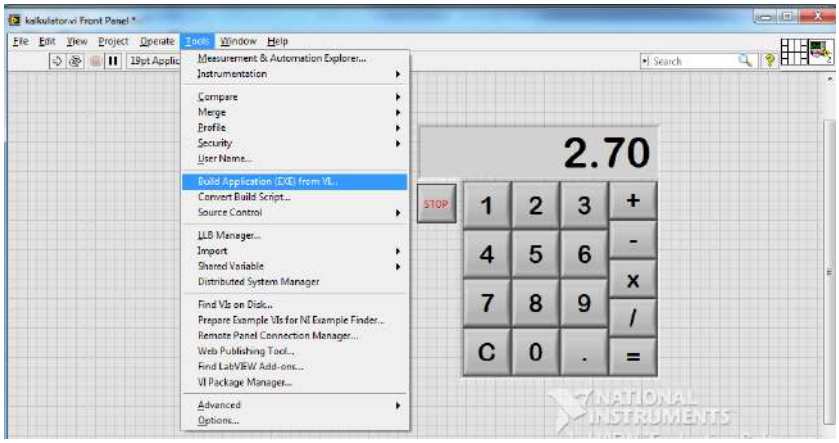
8.2 Membuat File EXE

Yang dimaksud **file EXE** di sini adalah sebuah file program yang langsung dapat dijalankan pada sebuah komputer, tanpa perlu melakukan instalasi LabVIEW pada komputer tersebut, atau sering disebut sebagai sebuah ***Stand Alone Application***.

Dengan membuat file EXE ini, jelas banyak keuntungan yang diperoleh, seperti distribusi program menjadi mudah dengan ukuran file yang lebih kecil, karena tidak perlu menginstal LabVIEW untuk menjalankannya. Juga pengguna program tidak akan dapat mengubah atau melihat kode program yang dibuat oleh pemrogram (Block Diagram tidak dapat dimunculkan, hanya Front Panel saja yang ditampilkan).

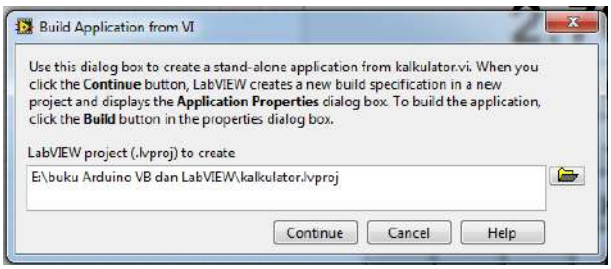
Berikut ini langkah-langkah pembuatan file EXE tersebut:

1. Pada jendela Front Panel atau Block Diagram, buka menu **Tools**, dan klik pada pilihan **Build Application (EXE) from VI…**



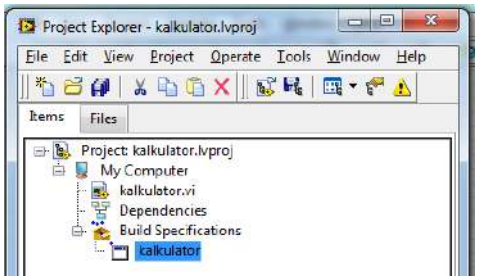
Gambar 8.47 Pilih Build Application (EXE) from VI ...

2. Maka muncul kotak dialog **Build Application from VI** yang akan menempatkan hasil aplikasi di lokasi sesuai kolom. Klik **Continue**.



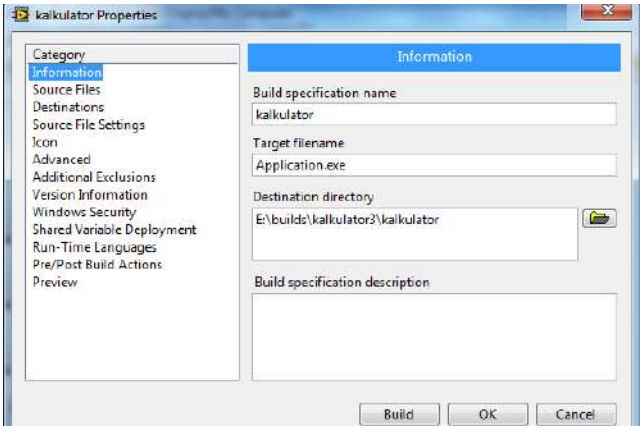
Gambar 8.48 Kotak Dialog Build Application from VI

3. Maka sebuah project tercipta dengan nama sesuai file program.



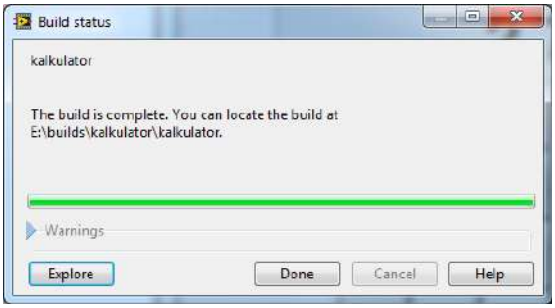
Gambar 8.49 Hasil project yang tercipta

4. Kemudian muncul kotak dialog **properties** file EXE. Klik **Build**.



Gambar 8.50 Kotak dialog properties

5. Maka proses pembuatan berjalan dengan status proses terlihat pada kotak **Build Status**. Setelah status ***complete***, klik **Done**, maka file EXE telah berhasil dibuat.



Gambar 8.51 Kotak Build status

6. Buka file EXE pada lokasi seperti yang ditunjukkan pada keterangan di **Build Status**. Perhatikan bahwa tampilan program file EXE tersebut sama seperti tampilan jendela Front Panel di gambar 8.47.

Gambar 8.52 Menjalankan file EXE

7. Untuk mengubah tampilan file EXE tersebut, pembaca dapat melakukannya dengan mengubah tampilan jendela Front Panel.

Gambar 8.53 Tampilan jendela Front Panel diatur seminimal mungkin

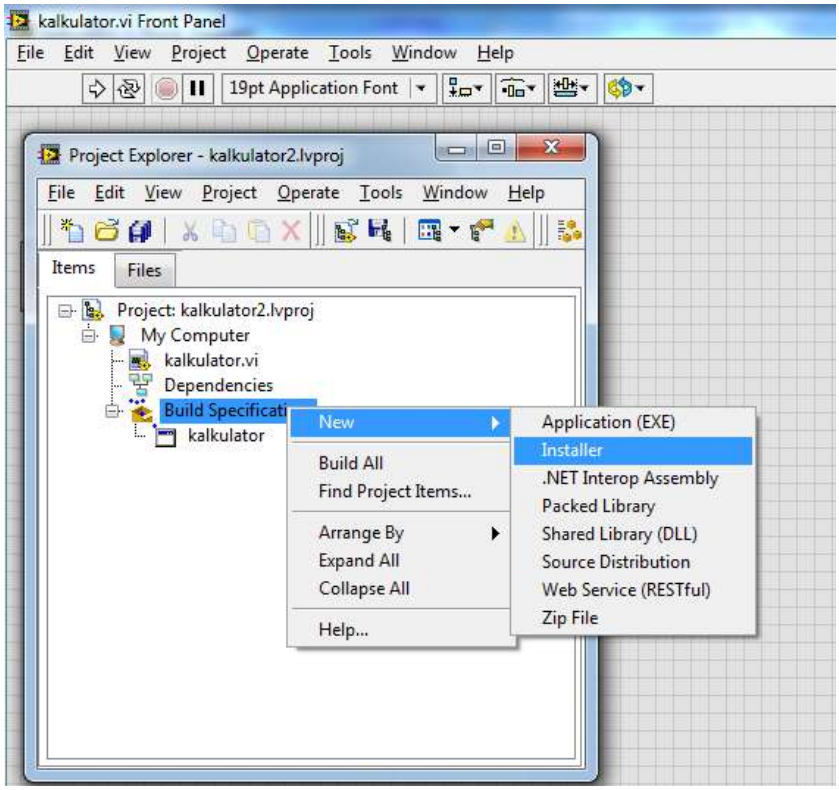
8. Setelah tampilan jendela Front Panel diubah, lakukan langkah 2-5 di atas, maka tampilan file EXE akan berubah mengikuti jendela Front Panel, seperti ditunjukkan pada gambar berikut:

Gambar 8.54 Tampilan file EXE selalu mengikuti tampilan Front Panel

9. File EXE yang dihasilkan tersebut hanya dapat dijalankan pada komputer yang sudah terinstal LabVIEW. Apabila pembaca menginginkan file EXE yang bisa dijalankan di komputer yang belum

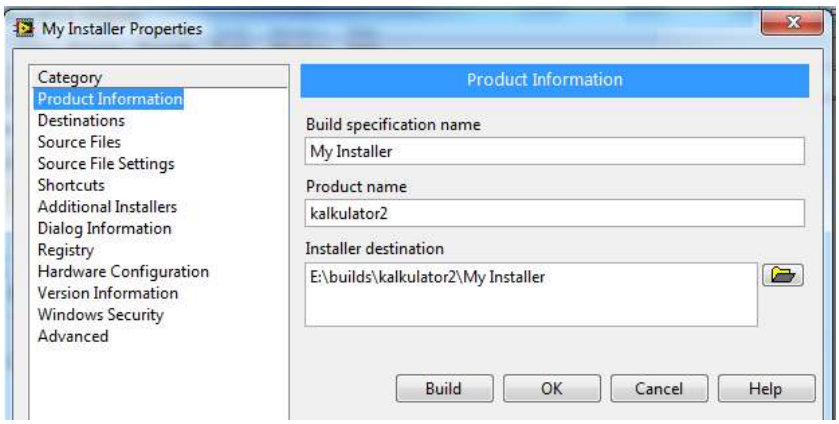
terinstal LabVIEW, maka buka project yang telah dibuat sebelumnya, dengan membuka menu **File**, kemudian **Open Project**.

10. Pada kotak **Project Explorer** yang terbuka, klik kanan pada **Build Specification**, klik **New**, kemudian klik **Installer**.

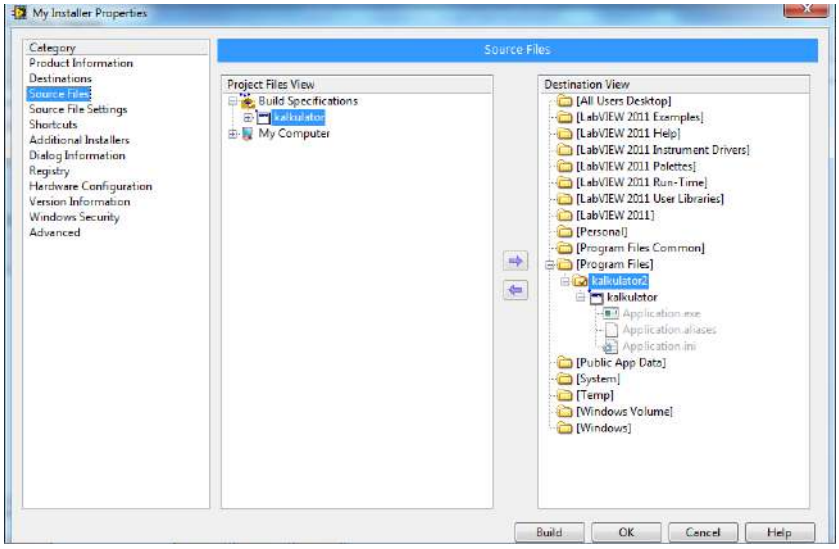


Gambar 8.55 Klik kanan Build Specification, pilih New, pilih Installer

11. Maka muncul kotak dialog **properties Installer**. Sebelum menekan tombol **Build**, tambahkan dulu **Source Files**. Buka kategori **Source Files**, pilih **kalkulator** di kolom **Project Files View**, kemudian tekan tombol tanda panah ke kanan, maka file kalkulator akan muncul di kolom **Destination View** sebagai **Source File**.



Gambar 8.56 Kotak dialog properties Installer

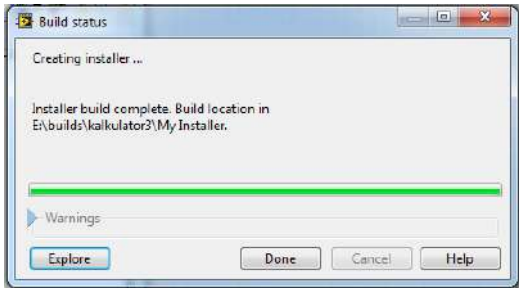


Gambar 8.57 Menambahkan file kalkulator sebagai Source File

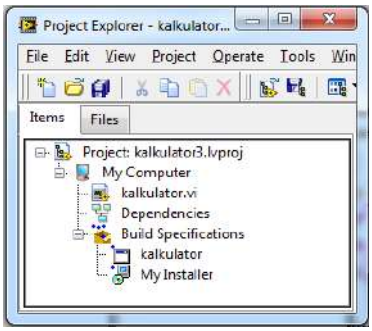
12. Tekan tombol **Build**. Maka muncul kotak **Build status**. Setelah selesai, klik **Done**, maka **Installer** telah berhasil dibuat.

13. Untuk menjalankan file EXE pada komputer yang belum terinstal LabVIEW, maka buka file Installer tersebut, klik pada **Setup**, maka program **Run Time Engine LabVIEW** akan terinstal di komputer

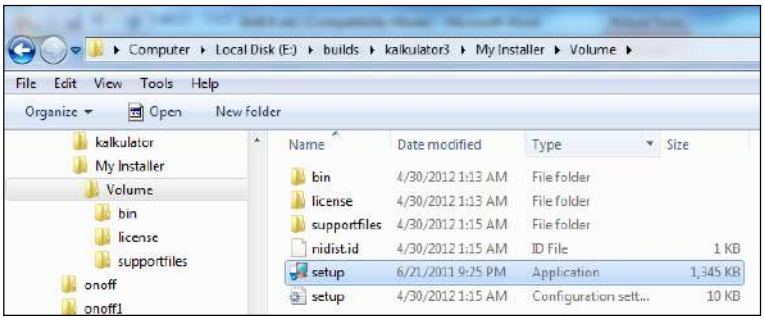
tersebut. Sekali terinstal, maka untuk file-file EXE berikutnya, tidak lagi diperlukan file Instaler, cukup file EXE saja yang dijalankan.



Gambar 8.58 Setelah Build status complete, klik Done



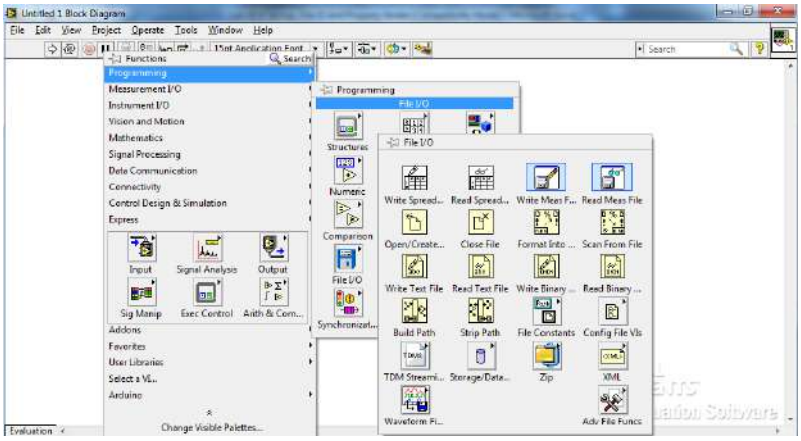
Gambar 8.59 Tampak file My Installer telah berhasil dibuat



Gambar 8.60 Klik Setup untuk menginstal Run Time Engine LabVIEW

8.3 File I/O

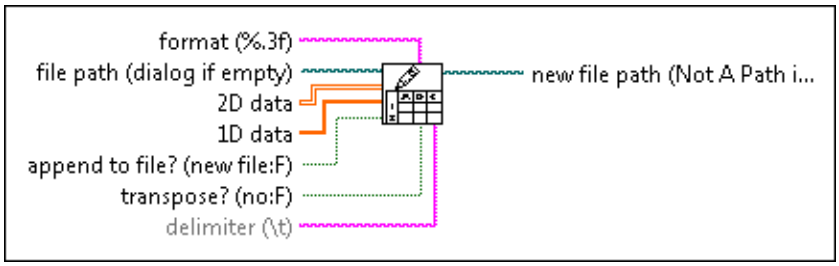
File I/O digunakan untuk operasi pembacaan/penulisan data pada sebuah file. Pembaca dapat melihat berbagai macam fungsi File I/O dengan membukanya di palet **Functions**, **Programming** dan **File I/O**.



Gambar 8.61 Fungsi-fungsi file I/O

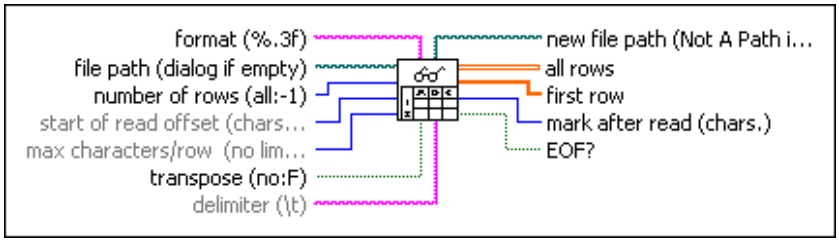
Berikut keterangan singkat dari beberapa fungsi file I/O tersebut:

8. **Write To Spreadsheet File**. Fungsi ini digunakan untuk mengubah sebuah Array 1D atau 2D dengan tipe data string, integer atau double, ke sebuah String teks dan menuliskan teks tersebut ke file yang baru atau menyisipkannya ke file yang sudah ada.



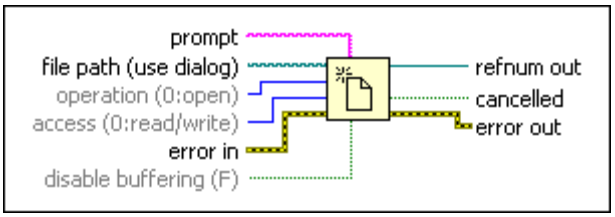
Gambar 8.62 Write To Spreadsheet File

9. **Read From Spreadsheet File**. Fungsi ini digunakan untuk membaca file yang berisi sejumlah baris teks angka dan mengubahnya ke Array 2D dengan tipe data string, integer atau double.



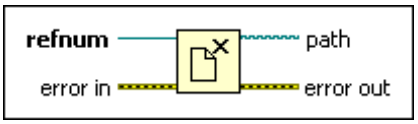
Gambar 8.63 Read From Spreadsheet File

10. **Open/Create/Replace File**. Fungsi ini digunakan untuk membuka sebuah file yang sudah ada, menciptakan file yang baru atau menumpuk (me-replace) file yang sudah ada.



Gambar 8.64 Open/Create/Replace File

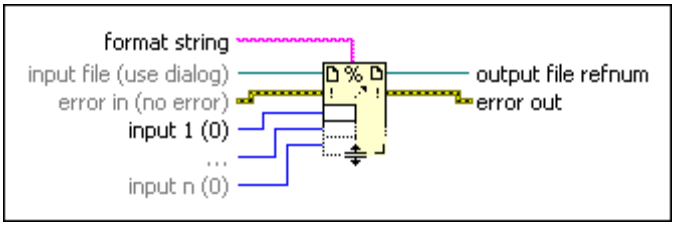
11. **Close File**. Fungsi ini digunakan untuk menutup sebuah file yang terbuka yang ditentukan oleh **refnum** dan menghasilkan jalur (*path*) ke file tersebut.



Gambar 8.65 Close File

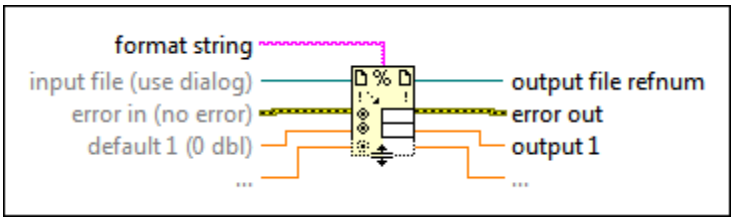
12. **Format Into File**. Fungsi ini digunakan untuk mengubah tipe data string, numeric, path atau Boolean menjadi teks dan menuliskannya

ke sebuah file. Secara default, penulisan selalu di awal file. Apabila diinginkan di akhir file, maka gunakan fungsi **Set File Position**.



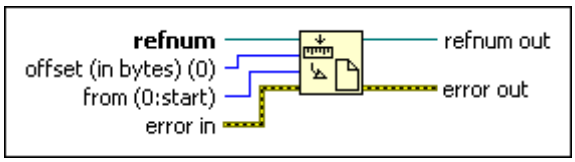
Gambar 8.66 Format Into File

13. **Scan From File**. Fungsi ini digunakan untuk mengubah teks menjadi tipe data tertentu. Pengubahan teks ini dilakukan untuk semua isi teks. Apabila diinginkan pengubahan sebagian, maka gunakan fungsi **Read From Text File** dipadukan dengan fungsi **Scan From String**.



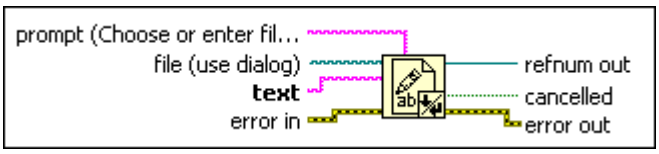
Gambar 8.67 Scan From File

14. **Set File Position**. Fungsi ini untuk mengatur posisi penulisan.



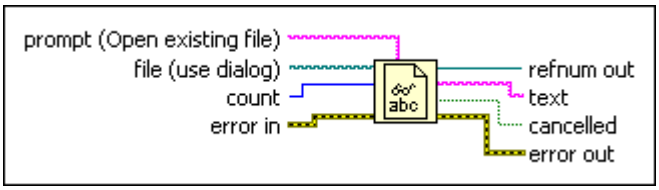
Gambar 8.68 Set File Position

15. **Write To Text File**. Fungsi ini digunakan untuk menulis sebuah string atau sebuah Array dari string sebagai baris data ke sebuah file.



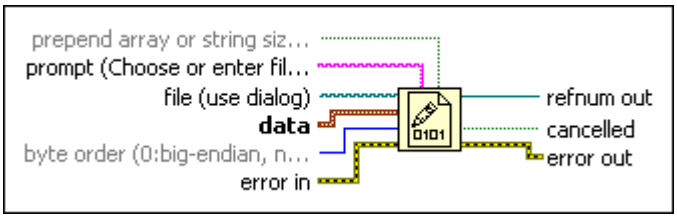
Gambar 8.69 Write To Text File

16. **Read From Text File**. Fungsi ini digunakan untuk membaca sejumlah karakter atau baris data dari sebuah file.



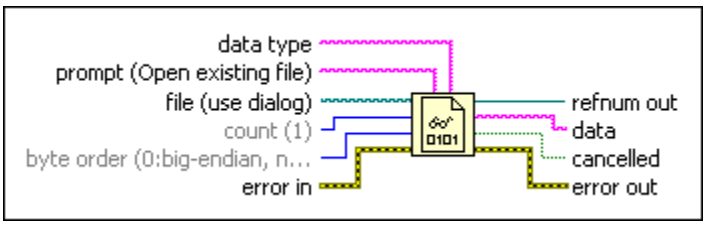
Gambar 8.70 Read From Text File

17. **Write To Binary File**. Fungsi ini digunakan untuk menulis data biner ke sebuah file yang baru, menambahkan data (*append*) ke file yang sudah ada, atau menumpuk (*replace*) data pada sebuah file.



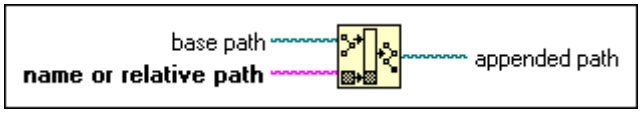
Gambar 8.71 Write To Binary File

18. **Read From Binary File**. Fungsi ini digunakan untuk membaca data biner dari sebuah file.



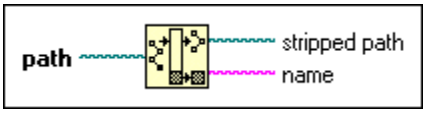
Gambar 8.72 Read From Binary File

19. **Build Path**. Fungsi ini digunakan untuk menciptakan sebuah jalur yang baru dengan menambahkan nama ke jalur yang sudah ada.



Gambar 8.73 Build Path

20. **Strip Path**. Fungsi ini digunakan untuk menghasilkan nama dari komponen terakhir sebuah jalur dan juga menghasilkan jalur induknya (*striped path*).



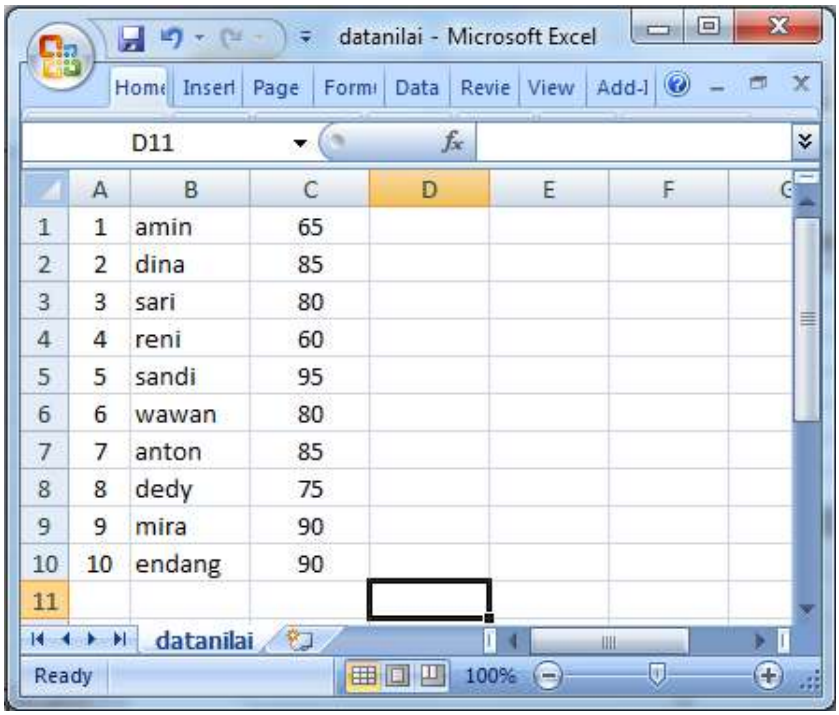
Gambar 8.74 Strip Path

Agar lebih jelas mengenai pemakaian dan fungsi-fungsi File I/O di atas, maka berikut ini disajikan contoh aplikasi File I/O untuk pembacaan data dari sebuah file dan penyimpanan data ke dalam sebuah file.

8.3.1 Pembacaan data dari sebuah file

Contoh berikut ini akan menyajikan bagaimana membuat sebuah grafik di LabVIEW dengan data yang diambil atau dibaca dari sebuah file txt. File txt tersebut dapat dibuat dengan **Notepad** atau **Excel**. Untuk mudahnya, pembaca dapat menggunakan **Excel**, karena di **Excel** sudah terdapat

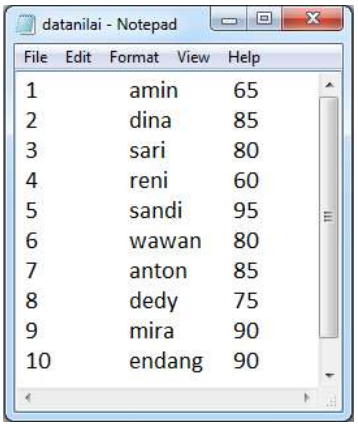
kolom-kolom, yang membuat pemisahan yang jelas antara data satu dengan data lainnya. Sedangkan apabila pembaca menggunakan **Notepad**, maka pembaca harus menekan tombol **Tab** untuk memisahkan data yang sebaris, dan tombol **Enter** untuk data berikutnya. Sebagai contoh data, berikut ini hasil nilai ujian matematika dari 10 orang siswa, yang datanya dimasukkan di **Excel**:



Gambar 8.75 Data hasil ujian yang dimasukkan di Excel

Kemudian simpan data nilai di atas dengan memilih **Save As** dari menu **File**. Pilih lokasi penyimpanan file yang mudah ditemukan pada kotak dialog **Save As** yang muncul, dan juga yang terpenting adalah, pada kolom **Save as type**, di bawah kolom **File name**, pilih tipe file **Text (Tab delimited)**, maka file txt akan dihasilkan. Sekalipun file tersebut berekstensi txt, untuk membukanya kembali di Excel tidak menjadi

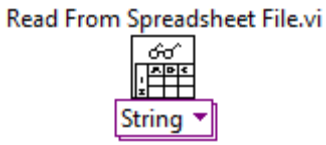
masalah, hanya dengan memilih **Tab** sebagai **delimiter**-nya saat ditanyakan, maka Excel akan dapat menempatkan data tersebut dalam kolom yang sama seperti Gambar 8.75 di atas. File txt tersebut juga dapat dibuka dengan **Notepad**, yang terlihat seperti gambar berikut ini:



Gambar 8.76 File txt dari Excel yang dibuka dengan Notepad

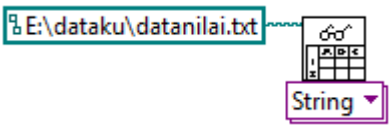
Setelah file txt berhasil dibuat, maka berikut ini langkah-langkah pembacaan data dari file tersebut dan pembuatan grafiknya di LabVIEW:

1. Ambil fungsi **Read From Spreadsheet File** dari palet Functions dan tempatkan pada Block Diagram. Kemudian berhubung tipe data file txt yang dibuat di atas tidak hanya terdiri dari angka saja, tetapi juga teks nama, maka ganti tipe data fungsi dari **Double** menjadi **String**.



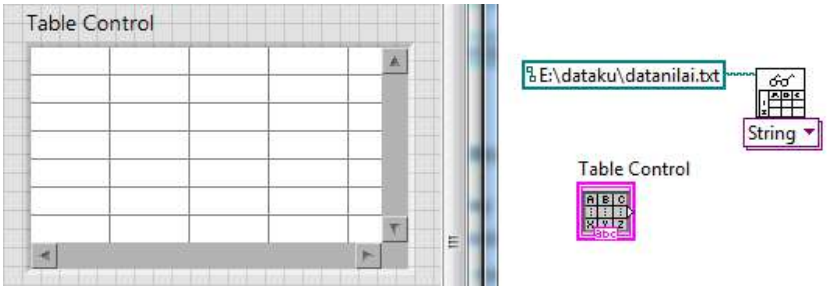
Gambar 8.77 Ganti tipe data dari Double menjadi String

2. Klik kanan pada kaki **file path** (kaki paling atas di sisi kiri), dan pilih **Create Constant** dari menu popup yang muncul. Kemudian ketikkan lokasi file txt pada kotak Constant tersebut.



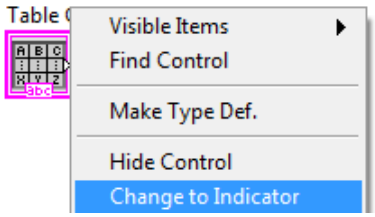
Gambar 8.78 Mengisi kaki file path dengan lokasi file txt

3. Pada jendela Front Panel, tempatkan **Table Control** yang diambil dari palet Controls (gunakan bantuan tombol Search).



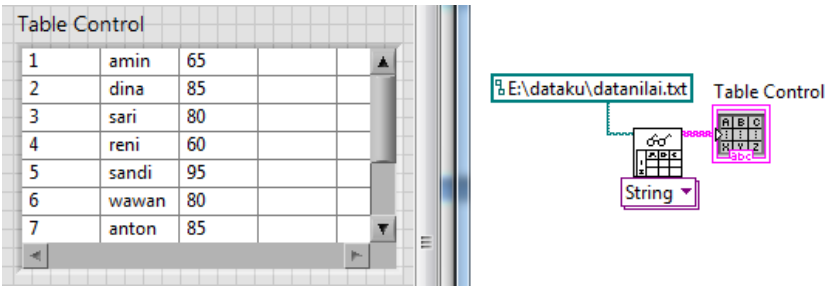
Gambar 8.79 Menempatkan Table Control di jendela Front Panel

4. Untuk bisa menampilkan data dari fungsi **Read From Spreadsheet file** pada **Table Control**, maka ubah tipe objek **Table Control** dari tipe Control menjadi Indicator, dengan cara meng-klik kanan icon objek tersebut, dan memilih **Change to Indicator**.



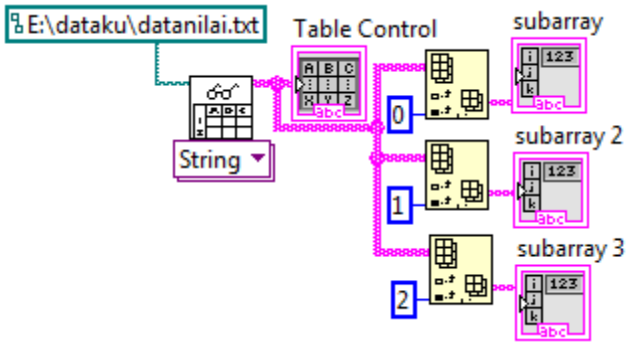
Gambar 8.80 Mengubah tipe Control menjadi Indicator

5. Hubungkan kaki **all rows** (kaki kanan atas) fungsi **Read From Spreadsheet file** ke kaki input **Table Control** (sekarang **Table Indicator**), kemudian jalankan program dengan menekan tombol Run, maka seharusnya data dari file txt muncul di **Table Control**.



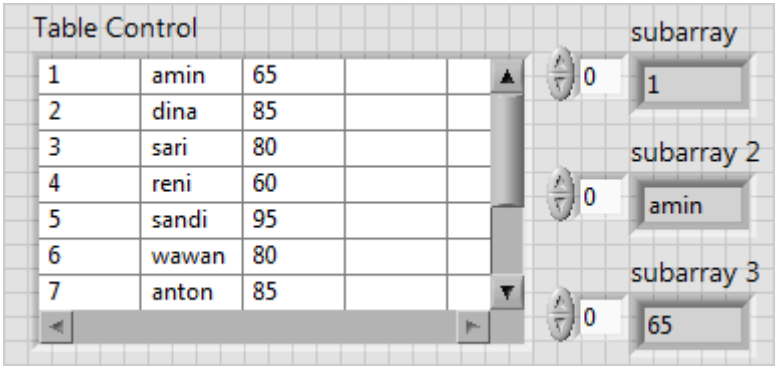
Gambar 8.81 Data file txt muncul di Table Control

6. Untuk membuat grafik dari data di dalam **Table Control**, maka perlu memisahkan data tersebut menggunakan fungsi **Index Array**, dengan kaki **disabled index (col)** diberi nilai 0 untuk mengambil data kolom pertama, nilai 1 untuk data kolom kedua dan seterusnya. Tambahkan **Indicator** dengan cara meng-klik kanan pada kaki output **Index Array**, dan kemudian pilih **Create Indicator**.



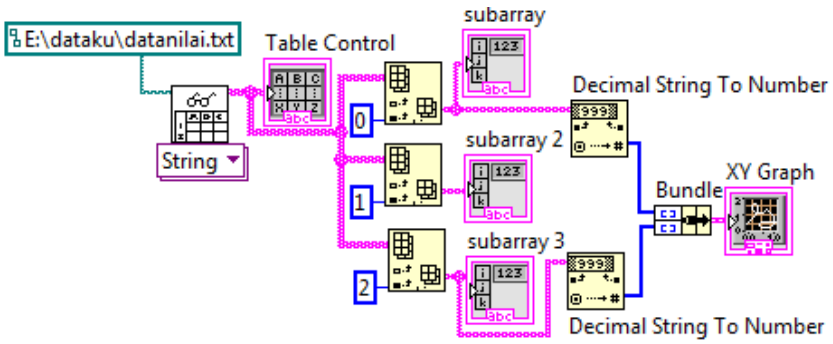
Gambar 8.82 Memisahkan data per kolom dari Table dengan Index Array

7. Jalankan program, maka di Front Panel akan terlihat seperti berikut:



Gambar 8.83 Tampak hasil pemisahan data per kolom pada subarray

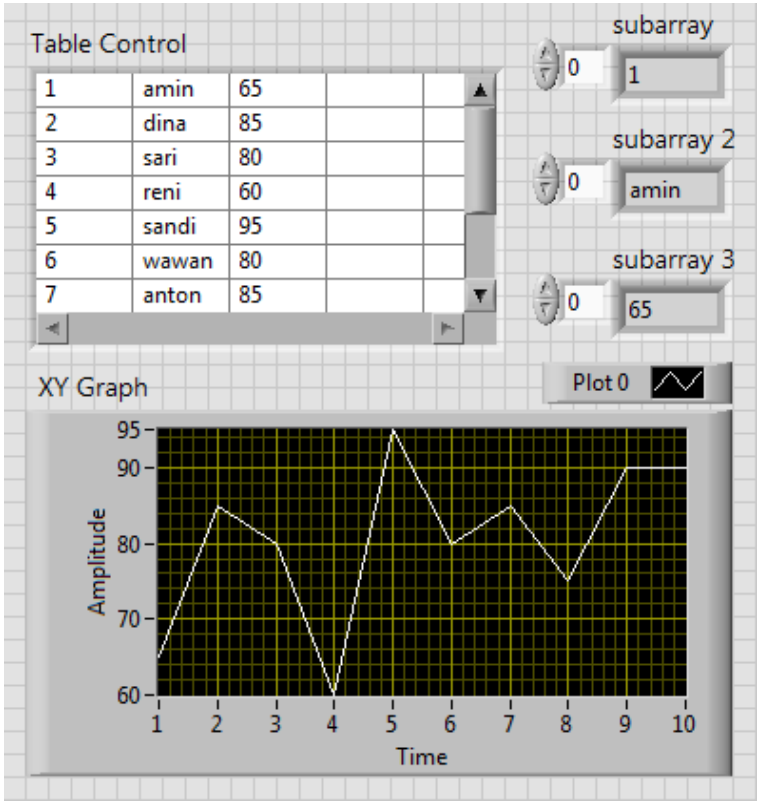
8. Kemudian tambahkan objek **XY Graph** di **Front Panel**, dan fungsi **Decimal String to Number**, untuk mengubah String menjadi Number pada data kolom pertama dan data kolom ketiga, yang akan dibuat menjadi data sumbu X dan sumbu Y untuk **XY Graph**.



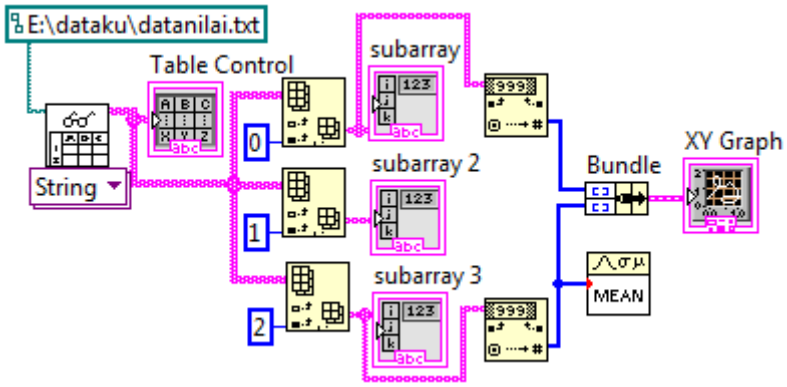
Gambar 8.84 XY Graph mengambil data dari kolom pertama dan kolom ketiga yang sebelumnya telah diubah dari String menjadi Number

9. Jalankan program dengan menekan tombol **Run**, maka akan muncul grafik nilai pada XY Graph seperti ditunjukkan pada Gambar 8.85.

10. Apabila diinginkan untuk menampilkan nilai rata-rata pada grafik tersebut, maka tambahkan fungsi **Mean** untuk menghasilkan nilai rata-rata data kolom ketiga seperti ditunjukkan pada Gambar 8.86.

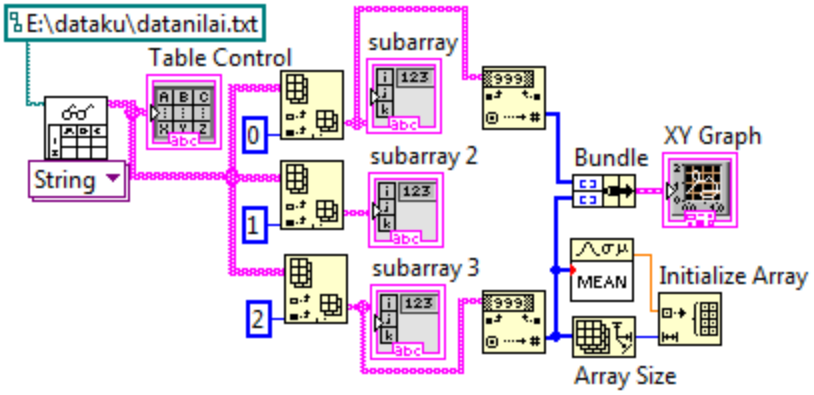


Gambar 8.85 Hasil grafik



Gambar 8.86 Menambahkan fungsi Mean untuk mendapatkan rata-rata

11. Kemudian agar data tersebut bisa ditampilkan dalam bentuk garis pada **XY Graph**, maka data tunggal tersebut harus menjadi Array yang seukuran dengan Array grafik sebelumnya. Untuk itu tambahkan fungsi **Array Size** untuk mendapatkan ukuran Array sebelumnya, dan fungsi **Initialize Array** untuk menciptakan Array nilai rata-rata seperti ditunjukkan pada gambar berikut.

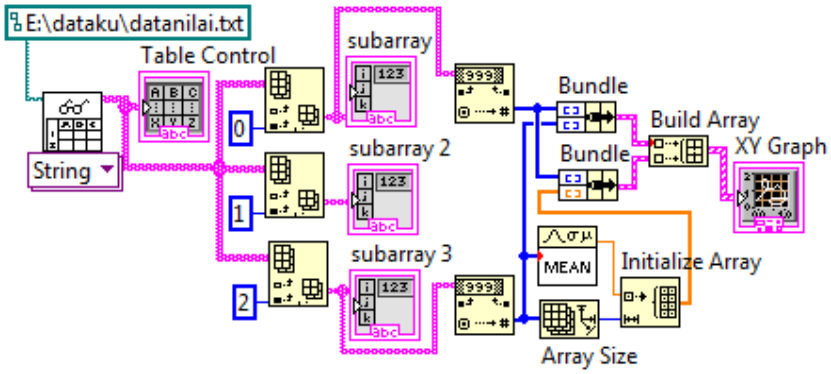


Gambar 8.87 Menambahkan fungsi Array Size dan Initialize Array

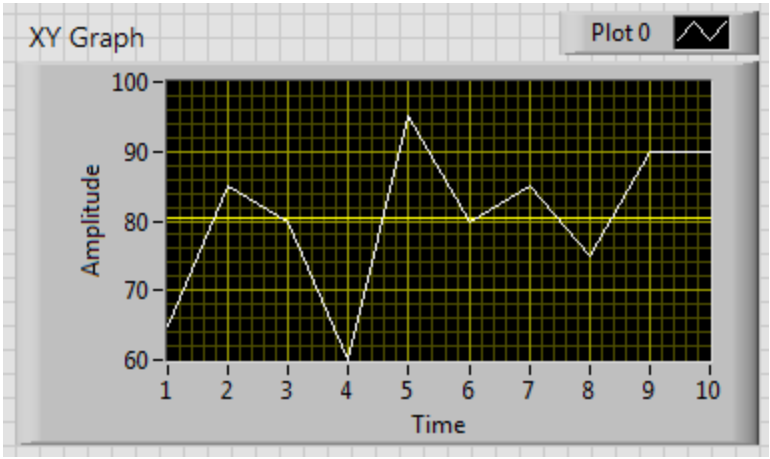
12. Terakhir, gabungkan data output **Initialize Array** sebagai data sumbu Y, dan data kolom pertama sebagai sumbu X, dengan menggunakan fungsi **Bundle**, kemudian satukan dengan grafik sebelumnya menggunakan fungsi **Build Array** seperti Gambar 8.88.

13. Jalankan program dengan menekan tombol Run, maka akan terlihat 2 grafik pada XY Graph, yaitu grafik nilai dan grafik nilai rata-rata seperti ditunjukkan pada Gambar 8.89.

14. Untuk pengubahan data, pembaca dapat melakukannya di software Excel atau di Notepad dengan membuka file txt yang disebut dalam program, dan kemudian menyimpannya kembali dengan tipe txt. Sampai di sini aplikasi File I/O untuk pembacaan file telah selesai.



Gambar 8.88 Menggabungkan kedua grafik dalam satu XY Graph dengan fungsi Build Array



Gambar 8.89 Menampilkan kedua grafik di Front Panel, tampak grafik nilai rata-rata berbentuk garis mendatar dengan nilai 80,5 pada sumbu Y

8.3.2 Penyimpanan data ke sebuah file

Contoh aplikasi File I/O berikutnya adalah penyimpanan data ke sebuah file. Dalam aplikasi ini, pembaca diminta untuk membuat sebuah **datalogger** yang melakukan pengambilan data suhu di suatu ruangan setiap detik, yang kemudian data tersebut disimpan pada sebuah file.

File tersebut berbentuk kolom-kolom data (*spreadsheet*) yang dapat dibuka dengan Excel, dengan isi data meliputi beberapa hal berikut ini,

1. Kolom pertama berisi catatan tanggal
2. Kolom kedua berisi catatan jam, menit dan detik.
3. Kolom ketiga berisi nilai suhu perdetik.
4. Kolom keempat berisi nilai rata-rata suhu setiap 3 kali pengambilan data.
5. Kolom kelima berisi catatan apakah nilai suhu tersebut melebihi batas nilai tertentu, dimana batas nilai ini dapat diatur dari Front Panel. Apabila melebihi batas, maka diberi catatan "LEBIH", dan apabila tidak melebihi batas, maka diberi catatan "NORMAL".

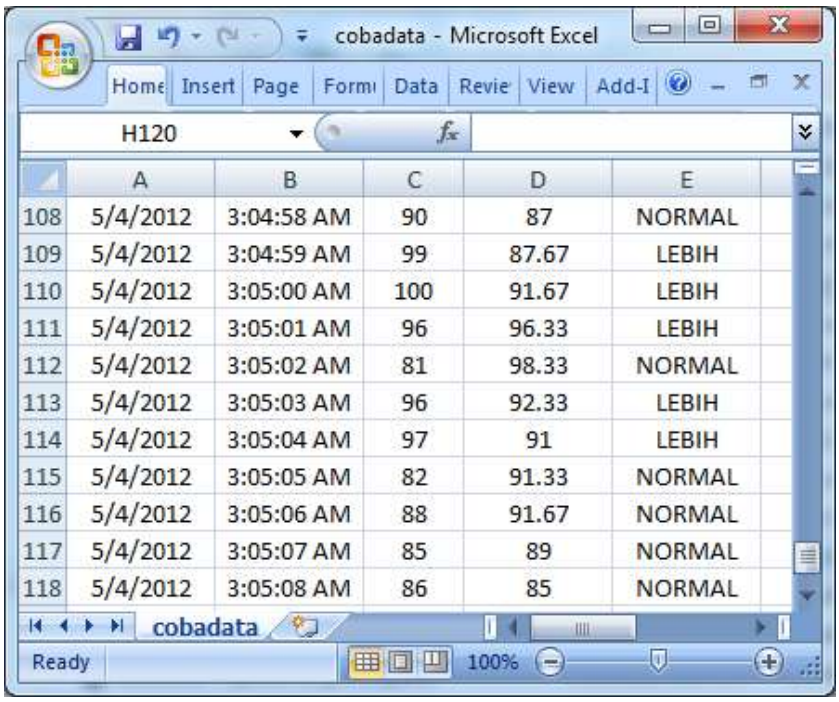
Penyimpanan data ke dalam sebuah file tersebut akan berlangsung ketika tombol Run ditekan, dan berhenti ketika tombol Stop ditekan.

Tidak hanya melakukan penyimpanan, pembaca diharapkan mampu menampilkan nilai data suhu perdetik tersebut pada sebuah grafik di Front Panel.

Kemudian ketika tombol Stop ditekan, sebuah grafik lagi yang lain di Front Panel akan menampilkan data nilai suhu dari awal sampai akhir penyimpanan.

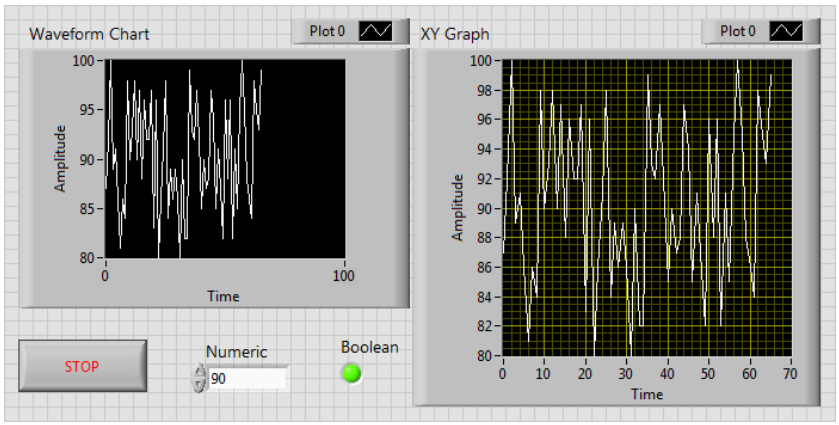
Sebuah LED indikator juga harus ditambahkan untuk memberi peringatan ketika nilai suhu tersebut melebihi batas nilai (LED padam bila tidak melebihi batas, dan LED menyala bila melebihi batas).

Untuk memberikan gambaran mengenai hasil program ini, Gambar 8.90 menunjukkan file yang dihasilkan dari datalogger, yang dibuka dengan Excel. Kemudian Gambar 8.91 menunjukkan tampilan 2 buah Grafik, sebuah LED, batas nilai dan tombol Stop pada Front Panel.



	A	B	C	D	E
108	5/4/2012	3:04:58 AM	90	87	NORMAL
109	5/4/2012	3:04:59 AM	99	87.67	LEBIH
110	5/4/2012	3:05:00 AM	100	91.67	LEBIH
111	5/4/2012	3:05:01 AM	96	96.33	LEBIH
112	5/4/2012	3:05:02 AM	81	98.33	NORMAL
113	5/4/2012	3:05:03 AM	96	92.33	LEBIH
114	5/4/2012	3:05:04 AM	97	91	LEBIH
115	5/4/2012	3:05:05 AM	82	91.33	NORMAL
116	5/4/2012	3:05:06 AM	88	91.67	NORMAL
117	5/4/2012	3:05:07 AM	85	89	NORMAL
118	5/4/2012	3:05:08 AM	86	85	NORMAL

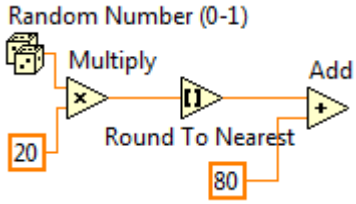
Gambar 8.90 File hasil datalogger yang dibuka dengan Excel



Gambar 8.91 Tampilan Front Panel dengan 2 buah Grafik, sebuah LED indikator, sebuah nilai batas dan sebuah tombol Stop

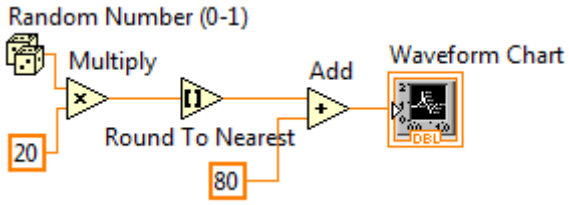
Berikut ini langkah-langkah pembuatan aplikasi di atas:

1. Berhubung interaksi data antara Arduino yang membaca sensor dengan software LabVIEW belum diperkenalkan, maka gunakan data simulasi menggunakan angka random sebagai nilai suhu. Buat nilai suhu dari angka random yang memiliki nilai antara 80 – 100 (bilangan bulat), dengan kode seperti berikut ini:



Gambar 8.92 Simulasi nilai suhu dengan nilai antara 80-100

2. Karena grafik pertama harus menampilkan data nilai suhu setiap detik, dalam arti harus diupdate setiap waktu, maka tipe grafik yang tepat adalah **Waveform Chart** (lihat kembali sub bab 7.2.1).



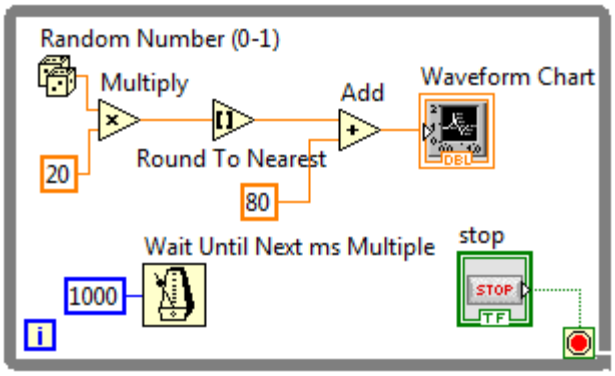
Gambar 8.93 Gunakan Waveform Chart untuk menampilkan titik data satu persatu, atau menampilkan data yang selalu diupdate setiap waktu

3. Tambahkan **While Loop** yang akan membuat program berjalan terus-menerus hingga tombol **Stop** di dalam While Loop ditekan.

4. Kemudian untuk membuat selang waktu 1 detik, gunakan fungsi **Wait Until Next ms Multiple** yang diberi nilai input 1000.

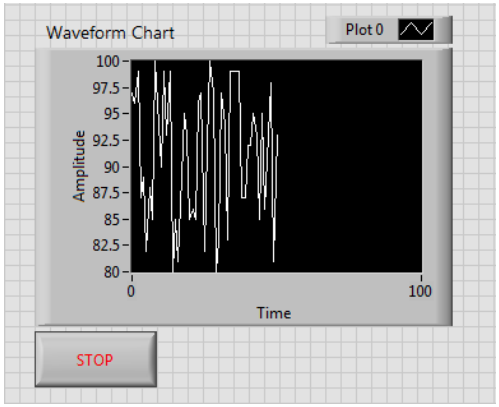
Catatan: fungsi **Wait Until Next ms Multiple** sama dengan **Wait (ms)**, yaitu menunda waktu sebesar nilai inputnya. Hanya bedanya, anggap

waktu saat ini adalah 12:30:00:475 (*jam:menit:detik:milidetik*), maka dengan fungsi **Wait (ms)** sebesar 1000, nilai waktu berikutnya adalah 12:30:01:475, 12:30:02:475, 12:30:03:475 dst-nya. Sedangkan dengan fungsi **Wait Until Next ms Multiple** sebesar 1000, nilai waktu berikutnya adalah 12:30:01:000, 12:30:02:000, 12:30:03:000 dst-nya.



Gambar 8.94 Menambahkan While Loop dan Wait Until Next ms Multiple

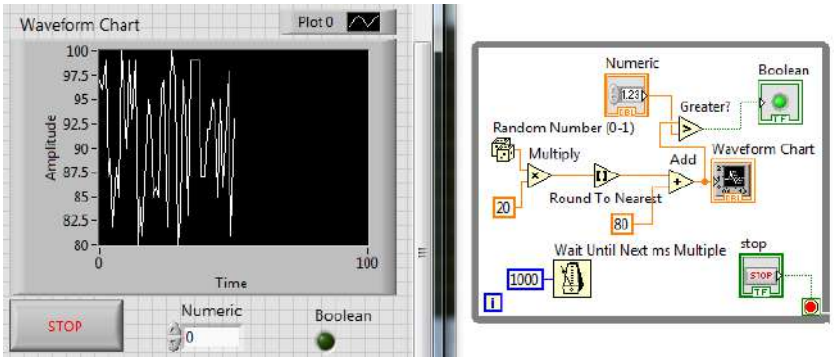
5. Jalankan program dengan menekan tombol **Run**.



Gambar 8.95 Penambahan data setiap detik ditampilkan pada grafik

6. Tambahkan objek **Num Ctrl** sebagai nilai batas dan **Round LED** sebagai LED indikator di jendela Front Panel. Nilai suhu harus

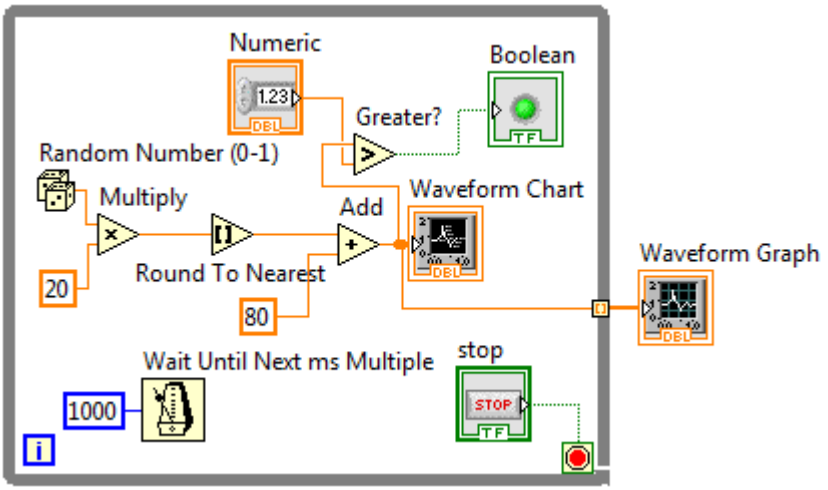
dibandingkan dengan nilai batas ini, dan apabila nilai suhu lebih besar, maka LED harus menyala. Untuk itu tambahkan fungsi **Greater?**, dan hubungkan semuanya seperti gambar berikut.



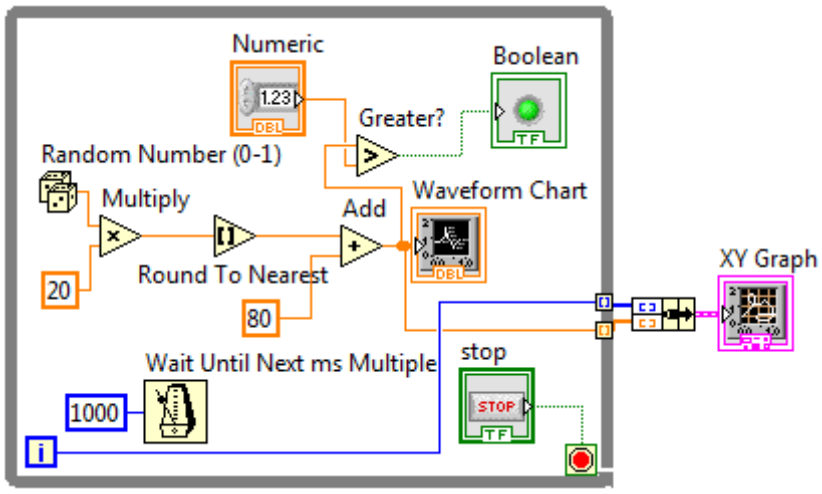
Gambar 8.96 Menambahkan nilai batas dan LED indikator

7. Ketika tombol **Stop** ditekan, maka grafik kedua harus menampilkan semua data keseluruhan dari awal hingga akhir. Karena data hanya diupdate sekali setelah semua data diterima, maka tipe grafik yang bisa digunakan adalah **Waveform Graph** atau **XY Graph**. Tambahkan **Waveform Graph** pada **Front Panel**. Kemudian di jendela Block Diagram, tempatkan icon **Waveform Graph** di luar **While Loop**. Hubungkan output dari data nilai suhu ke **Waveform Graph** menembus dinding **While Loop**. Klik kanan pada terminal tembusan di dinding **While Loop** tersebut, dan pilih **Enable Indexing** untuk mengumpulkan semua data dari awal hingga **While Loop** berhenti.

Catatan: apabila menggunakan **XY Graph**, maka pembaca perlu menambahkan data untuk sumbu X, yang dapat diambilkan dari nilai **Counter (i)**, sedangkan nilai suhu sebagai data untuk sumbu Y.



Gambar 8.97 Menambahkan Waveform Graph dengan input dari nilai suhu yang dibuat Enable Indexing pada dinding While Loop



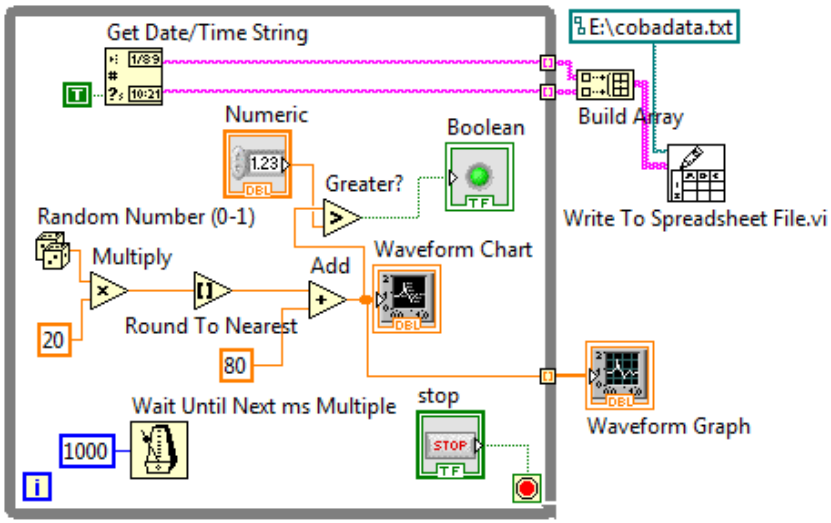
Gambar 8.98 Alternatif grafik kedua menggunakan XY Graph sebagai ganti Waveform Graph, dengan sumbu X diambil dari nilai Counter (i)

8. Berikutnya, adalah membuat datalogger untuk menyimpan data ke sebuah file. Untuk itu tambahkan fungsi **Write To Spreadsheet File** dan tempatkan di luar **While Loop.**

9. Klik kanan pada kaki input **file path** (kaki kiri atas) dari fungsi **Write to Spreadsheet File** dan pilih **Create Constant**. Isi kotak yang muncul dengan lokasi file dan nama file yang akan diciptakan.

10. Isi file pada kolom pertama dan kedua adalah data waktu, yaitu tanggal dan jam. Untuk itu tambahkan fungsi **Get Date/Time String**, dan tempatkan di dalam **While Loop**. Beri nilai **True** pada kaki ***want seconds?*** Kemudian keluarkan kedua output fungsi tersebut keluar dari dinding **While Loop**, dan buat **Enable Indexing** untuk mengumpulkan semua data dari awal hingga **While Loop** berhenti.

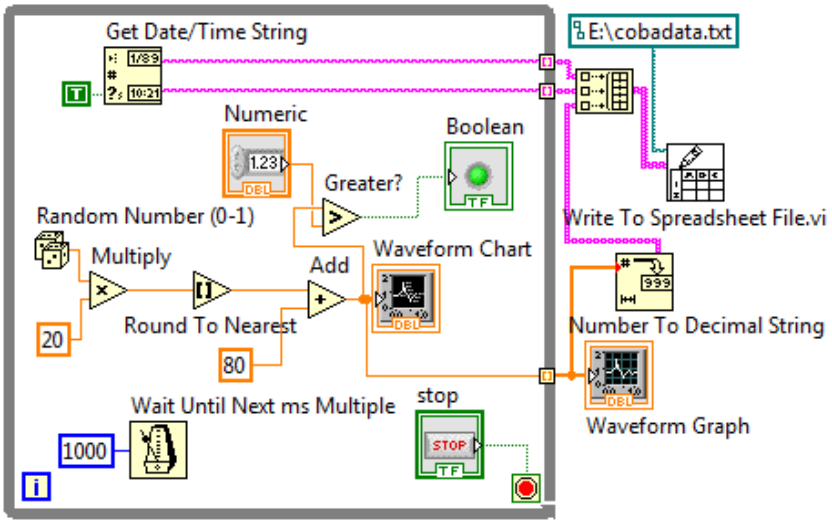
11. Tambahkan fungsi **Build Array** untuk menggabungkan data kolom pertama dan kedua, untuk kemudian diberikan ke fungsi **Write To Spreadsheet File**.



Gambar 8.99 Build Array untuk menggabungkan semua data

12. Data pada kolom ketiga adalah data nilai suhu. Tarik fungsi **Build Array** ke bawah, dan hubungkan terminal output **Enable Indexing** nilai suhu ke input **Build Array**. Namun sebelumnya, lakukan

pengubahan tipe data dari Number menjadi String menggunakan fungsi **Decimal String to Number**.



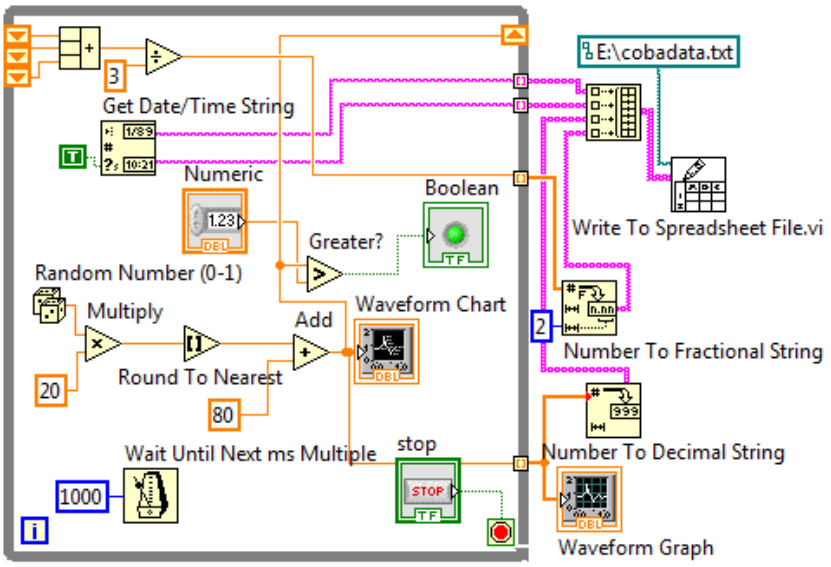
Gambar 8.100 Fungsi Number to Decimal String untuk mengubah tipe data nilai suhu dari Number menjadi String

13. Data pada kolom keempat adalah nilai rata-rata suhu untuk setiap 3 kali pengambilan data. Untuk itu gunakan fungsi **Shift Register**, dan hubungkan output nilai suhu ke terminal kanan **Shift Register**.

14. Kemudian tarik terminal kiri ke bawah hingga muncul tambahan 2 terminal. Tambahkan fungsi **Compound Arithmetic** untuk menjumlahkan ketiga data pada terminal kiri tersebut. Setelah itu gunakan fungsi **Divide** untuk membagi hasil penjumlahan dengan angka 3, sehingga diperoleh nilai rata-rata suhu.

15. Keluarkan output nilai rata-rata suhu dari **While Loop** dengan membuat **Enable Indexing**, dan hubungkan ke input **Build Array**. Namun sebelumnya, lakukan pengubahan tipe data menggunakan

fungsi **Number To Fractional String** dengan kaki input **precision** diberi nilai 2 (yaitu banyaknya angka di belakang koma).



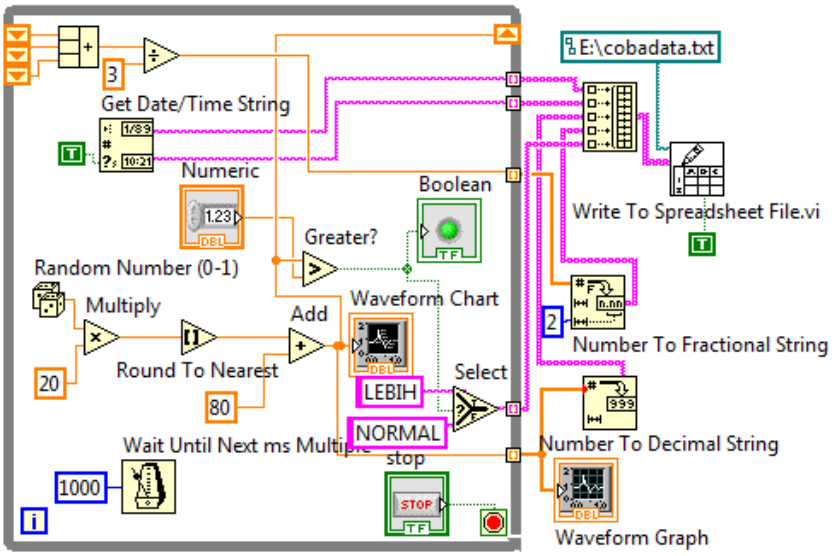
Gambar 8.101 Penambahan nilai rata-rata suhu untuk kolom keempat

16. Data pada kolom kelima adalah catatan apakah nilai suhu melebihi nilai batas atau tidak. Jika melebihi maka dicatat "LEBIH", jika tidak lebih maka dicatat "NORMAL". Untuk itu, tambahkan fungsi **Select**, dengan kaki input **t** diberi **String Constant** yang diisi tulisan LEBIH, kaki input **f** diberi **String Constant** yang diisi tulisan NORMAL, dan kaki input **s** dihubungkan dengan output fungsi **Greater**.

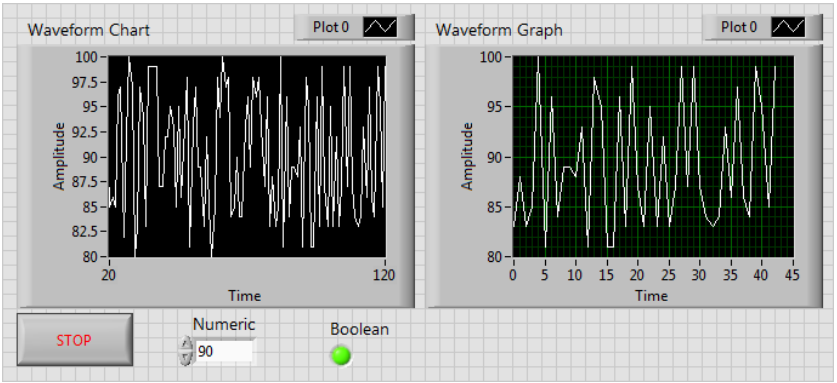
17. Kemudian hubungkan kaki output fungsi **Select** dengan **Build Array** melalui dinding **While Loop** yang dibuat **Enable Indexing**.

18. Terakhir, beri nilai **True** pada kaki input **transpose?** (kaki kiri bawah) dari fungsi **Write To Spreadsheet File** untuk mengubah posisi data.

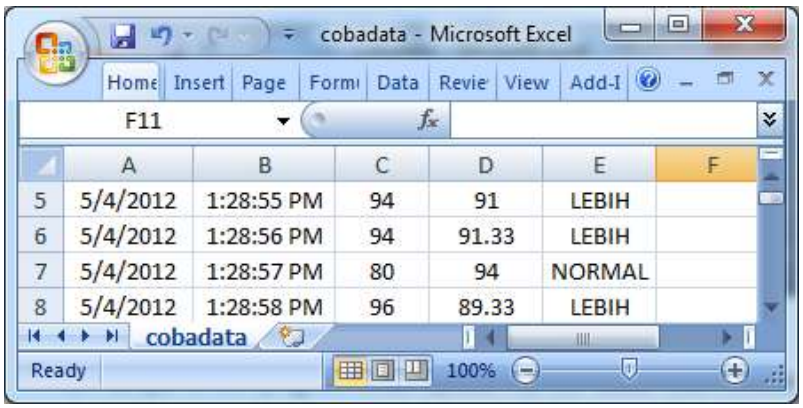
19. Kemudian jalankan program dengan menekan tombol **Run**. Tunggu hingga beberapa saat, dan kemudian tekan tombol **Stop**. Maka hasil program seperti terlihat pada Gambar 8.103 dan 8.104.



Gambar 8.102 Program lengkap untuk contoh aplikasi File I/O, yang menghasilkan datalogger dengan 5 buah kolom data

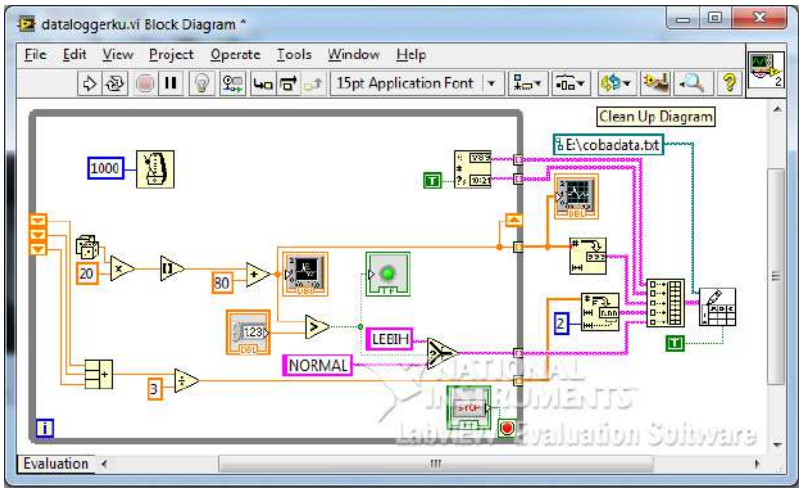


Gambar 8.103 Hasil program menampilkan grafik per detik dan grafik keseluruhan data dari awal dijalankan sampai tombol Stop ditekan



Gambar 8.104 File txt hasil datalooger yang dibuka dengan Excel

20. Sampai di sini program untuk aplikasi file I/O kedua telah selesai. Sebenarnya ada satu tool yang sangat istimewa di LabVIEW yang belum dimanfaatkan, yaitu tool **Clean Up Diagram**. Tool ini sangat membantu ketika kode Block Diagram menjadi begitu rumit seperti benang kusut. Tekan tombol **Clean Up Diagram**, maka Block Diagram akan menjadi rapi secara ajaib.



Gambar 8.105 Dengan menekan tool Clean Up Diagram, kode di Block Diagram menjadi lebih tertata rapi dan bersih dari garis yang putus

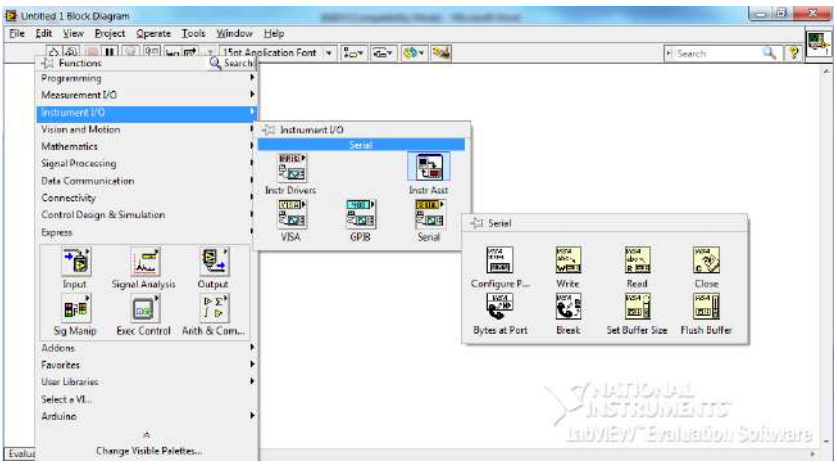
BAB 9
KOMUNIKASI SERIAL LABVIEW

9.1 Komunikasi Serial LabVIEW

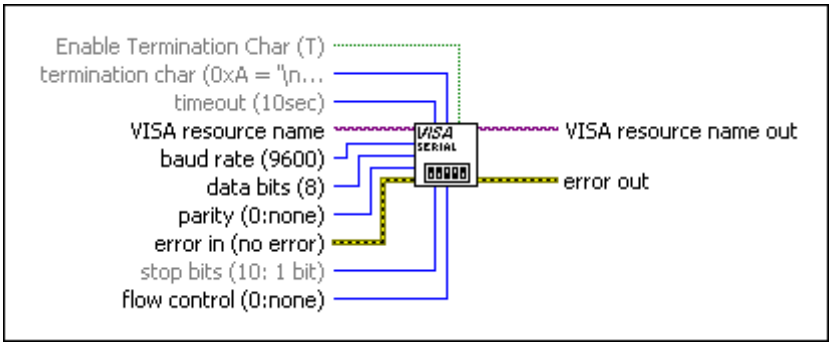
Apabila pada Bab 4 telah dibahas bagaimana membuat Arduino bisa berkomunikasi secara serial dengan komputer melalui **Serial Monitor**, maka pada Bab 9 ini akan dibahas bagaimana membuat Arduino bisa berkomunikasi secara serial dengan komputer melalui **LabVIEW**.

Untuk melihat fungsi-fungsi komunikasi serial LabVIEW, klik kanan jendela Block Diagram, pilih **Instrument I/O** dan kemudian pilih **Serial**. Berikut keterangan singkat dari fungsi-fungsi tersebut:

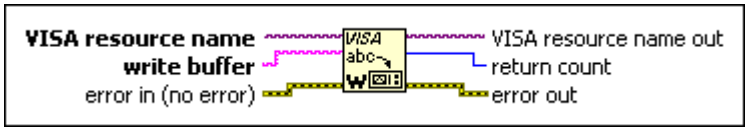
1. **VISA Configure Serial Port**. Fungsi ini digunakan untuk mengatur seting komunikasi serial di awal sebelum komunikasi dilangsungkan, termasuk saluran yang digunakan (***VISA resource name***), kecepatan komunikasi (***baud rate***), jumlah data bit, stop bit, dll.
2. **VISA Write**. Fungsi ini digunakan untuk menulis atau mengirimkan data dari **write buffer** ke suatu alat atau hardware interface yang ditentukan oleh ***VISA resource name***.



Gambar 9.1 Fungsi-fungsi untuk komunikasi serial

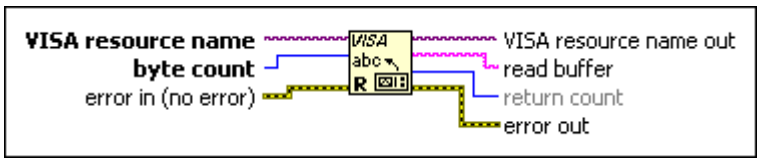


Gambar 9.2 VISA Configure Serial Port



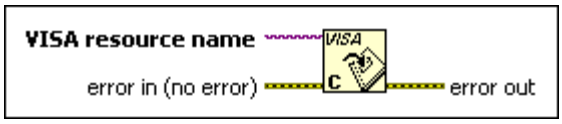
Gambar 9.3 VISA Write

3. **VISA Read**. Fungsi ini digunakan untuk membaca sejumlah byte data dari alat atau hardware interface yang ditentukan oleh ***VISA resource name***.



Gambar 9.4 VISA Read

4. **VISA Close**. Fungsi ini digunakan untuk menutup komunikasi dengan alat yang ditentukan oleh ***VISA resource name***.

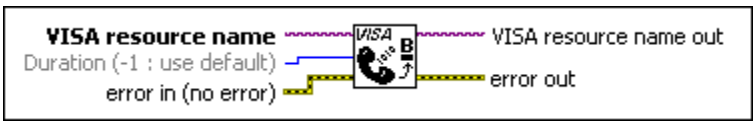


Gambar 9.5 VISA Close

5. **VISA Bytes at Serial Port**. Fungsi ini digunakan untuk menghitung banyaknya byte yang terdapat dalam buffer di saluran port serial yang ditentukan.

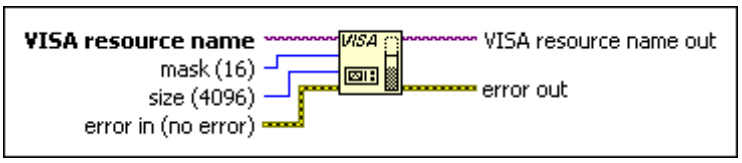
Gambar 9.6 VISA Bytes at Serial Port

6. **VISA Serial Break**. Fungsi ini digunakan untuk mengirimkan sebuah jeda (*break*) selama beberapa waktu (*duration*) pada saluran port serial yang ditentukan oleh ***VISA resource name***.



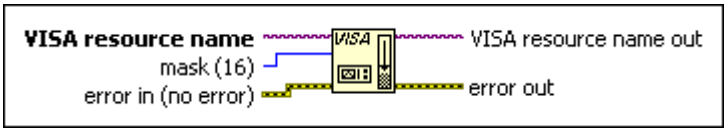
Gambar 9.7 VISA Serial Break

7. **VISA Set I/O Buffer Size**. Fungsi ini digunakan untuk mengatur ukuran Buffer I/O untuk kirim dan terima data dalam jumlah byte.



Gambar 9.8 VISA Set I/O Buffer Size

8. **VISA Flush I/O Buffer**. Fungsi ini digunakan untuk membersihkan isi Buffer dengan mengirimkan semua isinya ke alat yang ditentukan oleh ***VISA resource name.***



Gambar 9.9 VISA Flush I/O Buffer

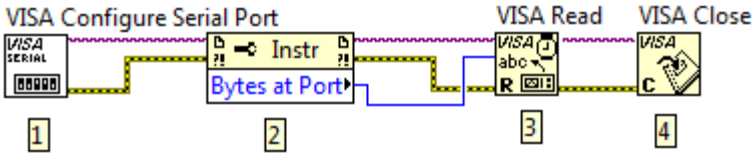
Agar lebih jelas mengenai penggunaan fungsi-fungsi tersebut, berikut ini disajikan contoh aplikasi komunikasi serial LabVIEW dengan Arduino. Contoh-contoh aplikasi di sini sama seperti contoh aplikasi pada Bab 4, hanya bedanya, tidak lagi menggunakan Serial Monitor, sebagai gantinya menggunakan LabVIEW. Dengan begitu, diharapkan pembaca akan lebih memahami komunikasi serial LabVIEW dengan Arduino.

9.2 Contoh #1: LabVIEW menampilkan data ASCII Table yang dikirim dari Arduino

Berikut langkah-langkah pembuatannya:

1. Ulangi langkah-langkah di Contoh #1 pada Bab 4 hingga **Serial Monitor** menampilkan data **ASCII Table** yang dikirimkan Arduino.
2. Untuk membaca data ASCII Table dari Arduino dan menampilkannya di jendela Front Panel, maka LabVIEW membutuhkan 3 buah fungsi; fungsi pertama untuk mengatur seting komunikasi sekaligus membukanya, fungsi kedua untuk membaca data yang dikirimkan,

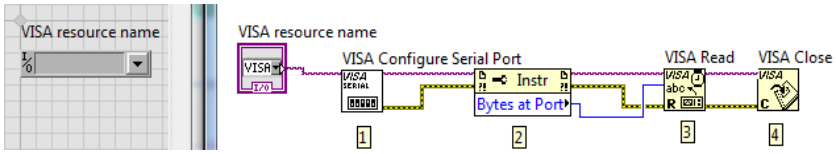
dan fungsi ketiga untuk menutup komunikasi. Ketiga fungsi tersebut adalah **VISA Configure Serial Port**, **VISA Read** dan **VISA Close**. Berhubung fungsi **VISA Read** membutuhkan input berapa banyak byte yang akan dibaca, maka tambahkan fungsi **VISA Bytes Serial At Port** untuk mendapatkan banyaknya byte yang diterima. Kemudian hubungkan keempat fungsi Serial tersebut seperti berikut:



Gambar 9.10 Susunan fungsi Serial untuk menerima data

3. Klik kanan kaki input ***VISA resource name*** di fungsi **VISA Configure Serial Port**, dan pilih **Create** kemudian **Control**, untuk bisa memilih saluran port yang sedang digunakan oleh Arduino di Front Panel.

Catatan: pilih **Create Control** apabiila menginginkan nilai dapat dimasukkan dari jendela Front Panel. Pilih **Create Constant** apabila menginginkan nilai masukan yang tetap, dan tidak lagi bisa diubah-ubah oleh pengguna (tidak muncul di Front Panel). Pilih **Create Indicator** apabila menginginkan suatu nilai dapat ditampilkan di Front panel.

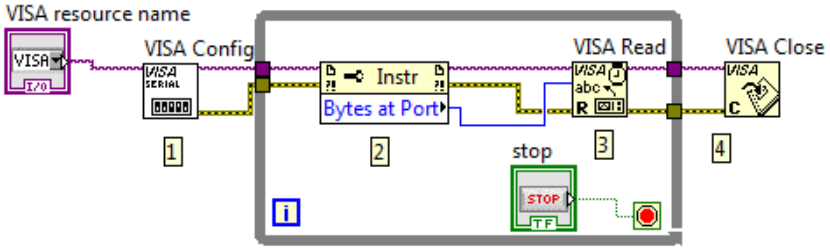


Gambar 9.11 Create Control pada kaki input VISA resource name

4. Untuk ***baudrate***, karena secara default sebesar 9600 (sama seperti baudrate Arduino), maka biarkan kaki inputnya tidak terhubung.

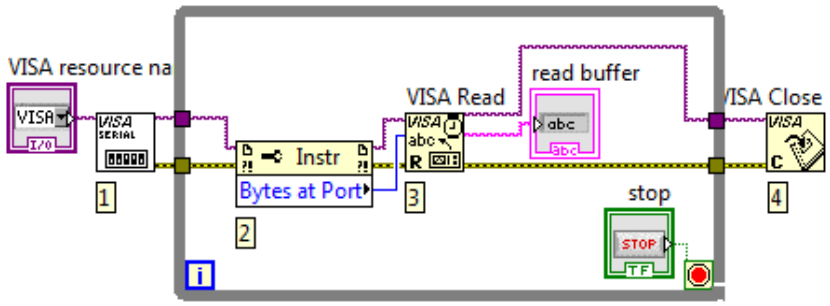
Catatan: kaki input sebuah fungsi yang dibiarkan tidak terhubung dengan nilai apapun, secara otomatis akan mengambil nilai defaultnya.

5. Berbeda dengan fungsi **VISA Configure Serial Port** yang hanya cukup sekali dilakukan, fungsi **VISA Read** perlu dilakukan terus-menerus. Untuk itu tambahkan struktur perulangan **While Loop** pada fungsi **VISA Read** dan juga **VISA Bytes Serial At Port**. Sedangkan fungsi **VISA Close** sama seperti **VISA Configure Serial Port**, yaitu hanya sekali dilakukan, untuk itu tempatkan di luar **While Loop**.



Gambar 9.12 Penambahan While Loop pada VISA Read dan Byte At Port

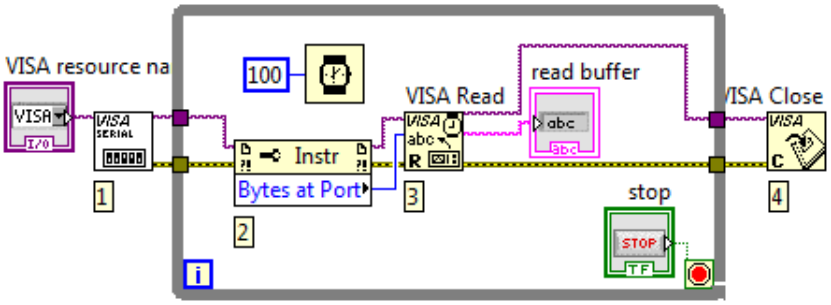
6. Berikutnya, untuk menampilkan pembacaan data oleh **VISA Read**, tambahkan sebuah **String Indicator**, yang dilakukan dengan cara meng-klik kanan kaki output **read buffer** (di sisi kanan) **VISA Read**, dan kemudian pilih **Create Indicator**.



Gambar 9.13 Klik kanan kaki read buffer dan pilih Create Indicator

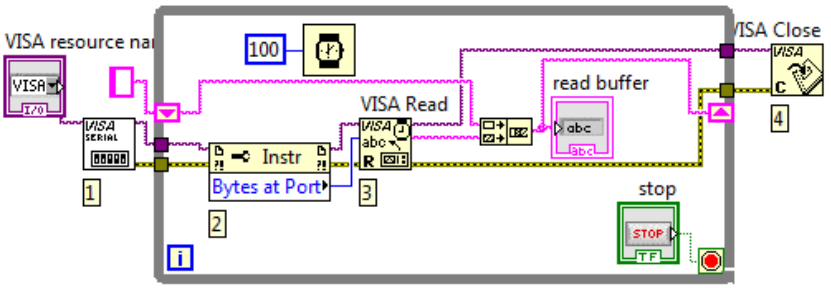
7. Berikutnya, tambahkan fungsi **Wait(ms)**, dan beri nilai input sebesar 100. Fungsi **Wait(ms)** ini sangat penting, karena tanpa diberi tunda

waktu, data akan ditampilkan dengan sangat cepat pada **String Indicator**. Dengan diberi tunda waktu, data menjadi tampak jelas.



Gambar 9.14 Tambahkan Wait(ms) sebesar 100 untuk membuat tampilan data pada String Indicator menjadi terlihat dengan jelas

8. Pada jendela Front Panel, tekan tombol **Run**, pilih **saluran COM** yang digunakan, dan perhatikan **String Indicator**. Tampak data ASCII Table ditampilkan, hanya sayangnya masih sebaris demi sebaris. Untuk menampilkan data ASCII Table secara utuh, maka tambahkan **Shift Register** untuk menyimpan data, dan fungsi **Concatenate String**, untuk menggabungkan semua data yang tersimpan tersebut.

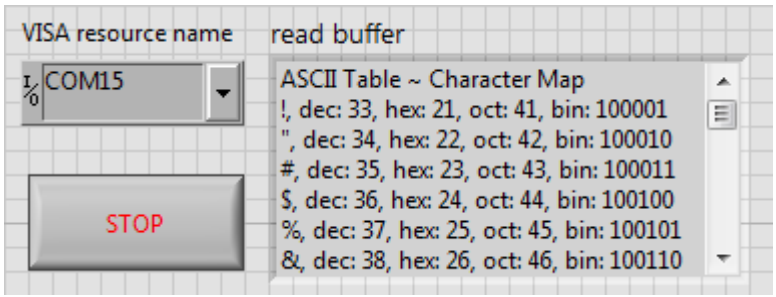


Gambar 9.15 Tambahkan Shift Register dan Concatenate Strings untuk menampilkan semua data ASCII Table pada String Indicator

9. Beri nilai awal **Shift Register** dengan **Empty String Constant**, atau dengan meng-klik kanan terminal kiri, dan kemudian memilih **Create**

Constant. Dengan cara ini, setiap kali program dijalankan, **String Indicator** akan menampilkan kotak yang kosong.

10. Untuk memunculkan data yang berada di bawah batas kotak **String Indicator**, gunakan **Vertical ScrollBar**, yang dapat dimunculkan dengan cara meng-klik kanan **String Indicator**, dan kemudian memilih **Visible Items**, **Vertical ScrollBar**.

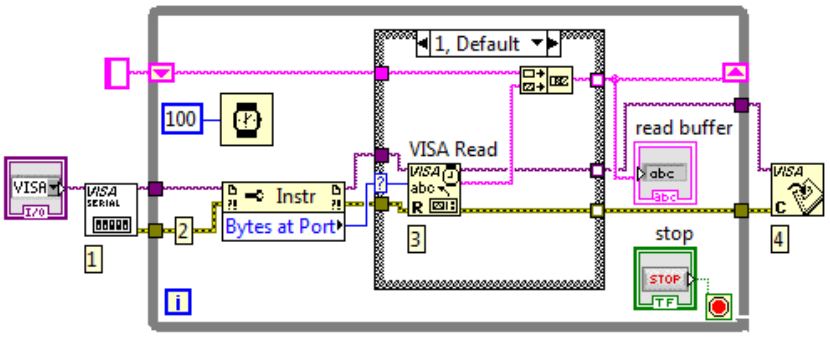


Gambar 9.16 Klik kanan String Indicator, pilih Visible Items dan kemudian Vertical ScrollBar untuk melihat data di bawah batas kotak

11. Terakhir, apabila diinginkan komunikasi serial secara otomatis berhenti ketika semua data ASCII Table telah selesai diterima oleh LabVIEW, maka tambahkan **Case Structure** pada fungsi **VISA Read** dan **Concatenate String**, yang digunakan untuk memeriksa apakah masih ada data ASCII Table atau tidak.

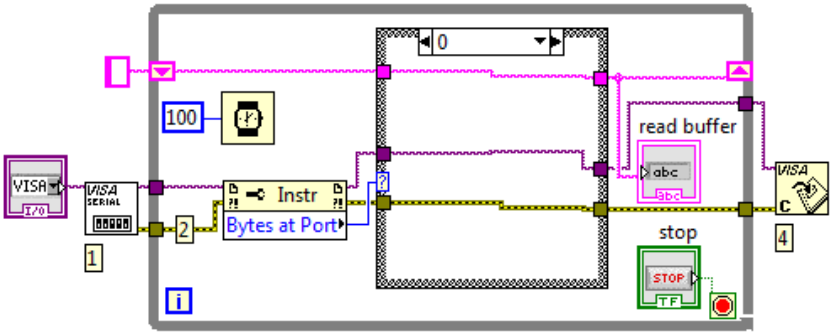
12. Hubungkan output **VISA Bytes Serial At Port** dengan terminal **selector Case Structure**, maka label **Case Structure** akan berubah dari **True/False** menjadi **1/0**.

13. Berhubung jumlah bytes bisa lebih dari 1, maka buat agar label 1 menjadi **Default**, dengan cara meng-klik kanan label 1 tersebut, kemudian memilih **Make This The Default Case**.



Gambar 9.17 Case Structure dengan Default Case di label 1

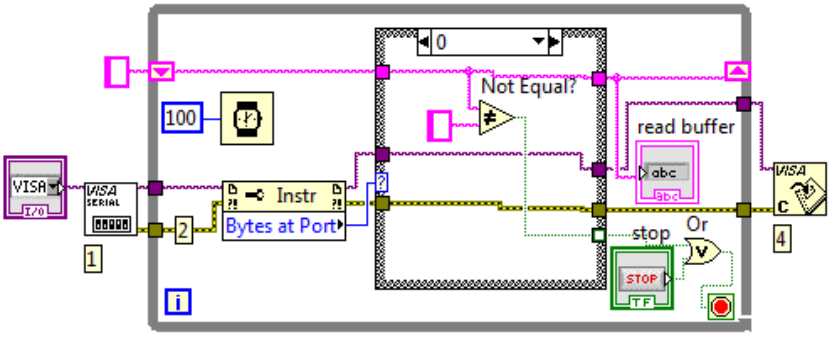
14. Kemudian untuk **blok Case 0**, hubungkan terminal kiri dengan terminal kanan sehingga tidak ada kotak yang berwarna putih.



Gambar 9.18 Menghubungkan terminal kiri dengan kanan di blok Case 0

15. Komunikasi, dalam hal ini pembacaan data, diinginkan berhenti ketika semua data telah diterima. Apabila data telah diterima semua, berarti nilai pada **VISA Bytes Serial at Port** akan sebesar 0, di mana pada nilai ini, **blok Case 0** aktif. Namun nilai 0 pada **VISA Bytes Serial at Port** juga terjadi ketika belum ada data yang diterima. Cara mudah untuk mengetahui apakah sudah ada data diterima atau belum adalah dengan menambahkan fungsi **Not Equal** (tidak sama dengan) yang membandingkan data pada String Indicator dengan **Empty String Constant**. Jika output fungsi **Not Equal** itu **False**, ini berarti belum ada

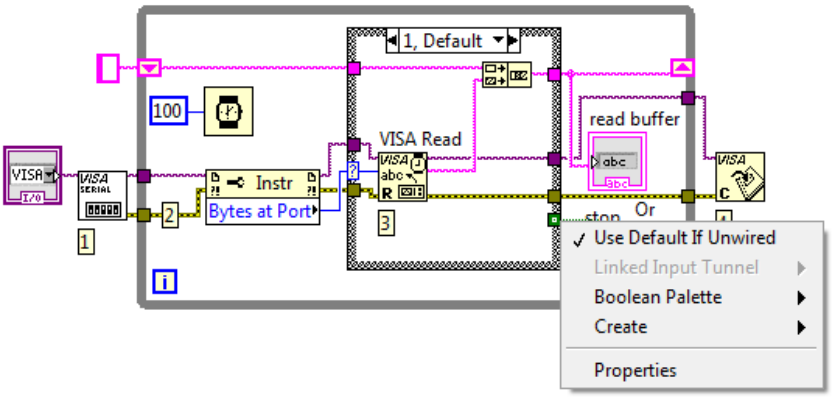
data yang diterima pada **String Indicator**. Tetapi jika outputnya **True**, berarti sudah ada data yang diterima pada **String Indicator**. Dengan menambahkan fungsi **Not Equal** ini di dalam **blok Case 0**, dan menghubungkan outputnya dengan **Loop Condition** (kotak dengan bulatan merah di tengahnya), akan membuat komunikasi secara otomatis berhenti setelah semua data diterima. Tambahkan fungsi **OR** untuk menggabungkan input dari tombol **Stop** dengan output fungsi **Not Equal**.



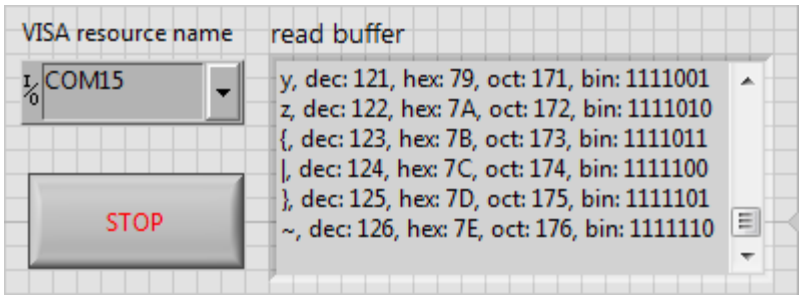
Gambar 9.19 Menambahkan fungsi Not Equal di blok Case 0, dan fungsi OR untuk menggabungkan output fungsi Not Equal dengan tombol Stop

16. Buat kotak putih pada dinding **blok Case** yang muncul karena fungsi **Not Equal** menjadi berwarna hijau dengan memberi nilai **False** atau dengan meng-klik kanan, dan pilih **Use Default if Unwired**.

17. Jalankan program dengan menekan tombol **Run**, pilih saluran **COM** yang sedang digunakan, dan perhatikan isi **String Indicator**. Seharusnya setelah nilai **dec** mencapai **126**, diikuti satu baris kosong, barulah program komunikasi berhenti secara otomatis. Sampai di sini contoh aplikasi #1 ASCII Table, dengan LabVIEW sebagai pengganti Serial Monitor telah berhasil dilakukan.



Gambar 9.20 Klik kanan pada kotak putih, pilih Use Default If Unwired



Gambar 9.21 Tampak setelah nilai dec mencapai 126 diikuti satu baris kosong, program secara otomatis berhenti

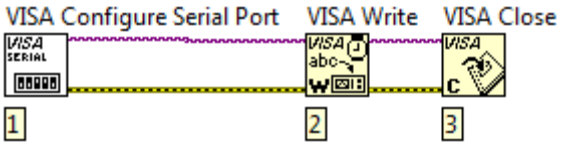
9.3 Contoh #2: Menyalakan LED dari LabVIEW

Berikut langkah-langkah pembuatannya:

1. Ulangi langkah-langkah di Contoh #2 pada Bab 4 hingga LED menyala ketika pada kolom **Send** di **Serial Monitor** diketikkan angka 1, dan LED padam ketika diketikkan angka lain.

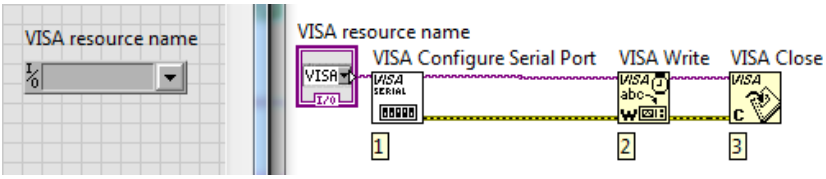
2. Untuk membuat LED menyala, maka LabVIEW harus mengirimkan karakter 1, dan karakter 0 agar LED padam. Untuk itu LabVIEW membutuhkan 3 buah fungsi; fungsi pertama untuk mengatur seting komunikasi sekaligus membukanya (**VISA Configure Serial Port**),

fungsi kedua untuk mengirimkan karakter (**VISA Write**), dan fungsi ketiga untuk menutup komunikasi (**VISA Close**).



Gambar 9.22 Susunan fungsi Serial untuk mengirimkan data

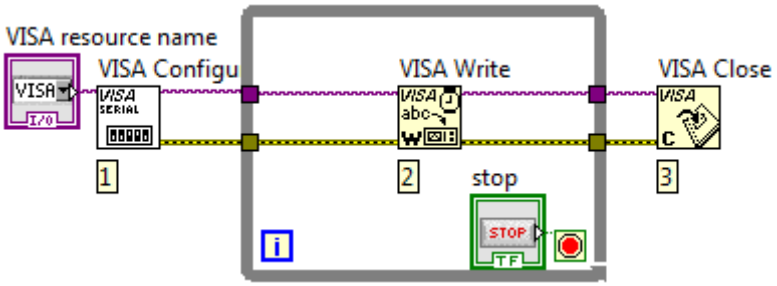
3. Klik kanan kaki input ***VISA resource name*** di fungsi **VISA Configure Serial Port**, dan pilih **Create** kemudian **Control**, untuk bisa memilih saluran port yang sedang digunakan oleh Arduino di Front Panel. Untuk ***baudrate***, karena secara default sebesar 9600 (sama seperti baudrate Arduino), maka biarkan kaki inputnya tidak terhubung.



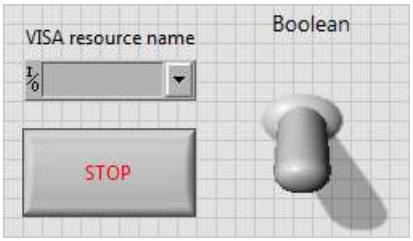
Gambar 9.23 Create Control pada kaki input VISA resource name

4. Berbeda dengan fungsi **VISA Configure Serial Port** dan **VISA Close** yang hanya cukup sekali dilakukan, fungsi **VISA Write** perlu dilakukan terus-menerus. Untuk itu tambahkan struktur perulangan **While Loop** pada fungsi **VISA Write**.

5. Berikutnya, untuk memudahkan pengiriman karakter 0 dan 1, tambahkan objek **Toggle Switch** di Front Panel. Diinginkan ketika **Toggle Switch** dituas ke atas, maka karakter 1 dikirimkan, dan ketika dituas ke bawah, maka karakter 0 dikirimkan.

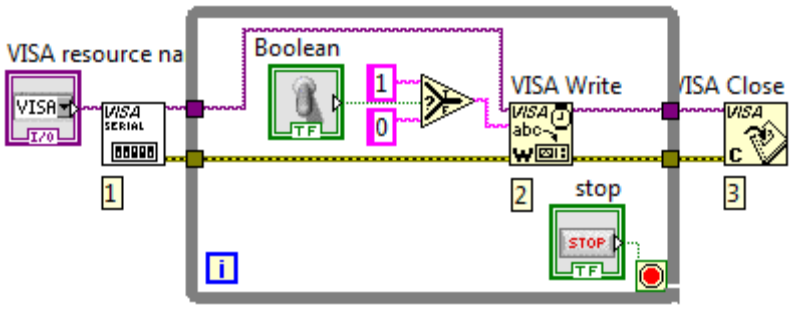


Gambar 9.24 Penambahan While Loop pada VISA Write



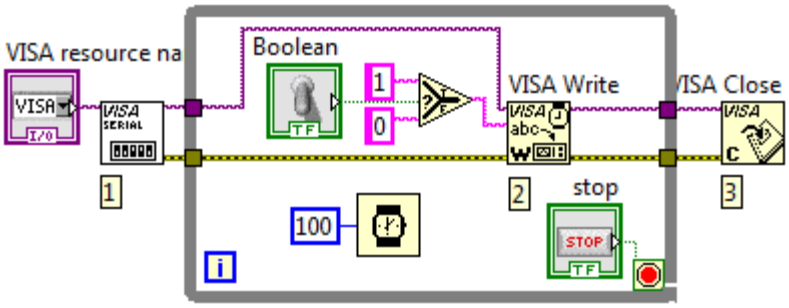
Gambar 9.25 Toggle Switch untuk memudahkan pengiriman karakter

6. Tambahkan fungsi **Select** di dalam **While Loop** dengan kaki input **s** terhubung ke output icon **Toggle Switch**, kaki input **t** terhubung dengan **String Constant** yang berisi karakter **1**, dan kaki input **f** terhubung dengan **String Constant** yang berisi karakter **0**, dan output dari fungsi **Select** terhubung ke kaki **write buffer VISA Write**.



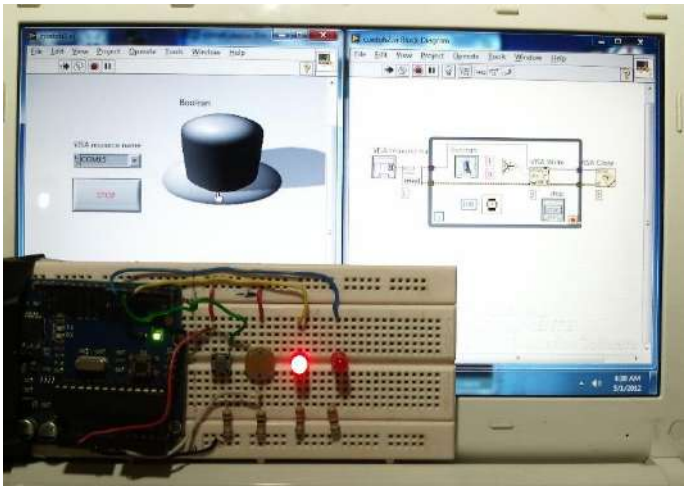
Gambar 9.26 Toggle Switch terhubung ke fungsi Select, ke write buffer

7. Berikutnya, tambahkan fungsi **Wait(ms)**, dan beri nilai input sebesar 100. Fungsi **Wait(ms)** ini sangat penting, karena tanpa diberi tunda waktu, maka pengiriman data akan gagal atau tidak terkirimkan.



Gambar 9.27 Tambahkan Wait(ms) sebesar 100 untuk memastikan data yang dikirimkan LabVIEW benar-benar diterima Arduino secara utuh

8. Jalankan program dengan menekan tombol **Run**, pilih saluran COM yang digunakan, kemudian tuas **Toggle Switch** ke atas dan ke bawah, perhatikan apakah LED menyala ketika tuas **Toggle Switch** ke atas? dan padam ketika tuas **Toggle Switch** ke bawah?



Gambar 9.28 LED menyala ketika tuas Toggle Switch digerakkan ke atas

9.4 Contoh #3: LabVIEW membaca tombol dan menampilkan statusnya dengan Round LED Indicator di Front Panel

Berikut langkah-langkah pembuatannya:

1. Ulangi langkah-langkah di Contoh #3 pada Bab 4 hingga **Serial Monitor** menampilkan angka 1 ketika tombol di D5 ditekan, dan angka 0 ketika tombol di D5 tidak ditekan.

2. Untuk memastikan bahwa pengiriman data dari Arduino benar-benar diterima di LabVIEW dengan baik, maka untuk seterusnya, gunakan aturan-aturan berikut ini dalam pemrograman di Arduino:

   - Selalu gunakan perintah **Serial.write** untuk mengirimkan data, sebagai ganti **Serial.print** atau **Serial.println**, karena selain lebih efisien, juga memastikan bahwa data yang dikirimkan adalah karakter ASCII sebesar 1 byte setiap kali pengiriman.

   - Tambahkan delay sebesar 100 ms untuk memastikan data yang terkirim benar-benar diterima oleh LabVIEW.

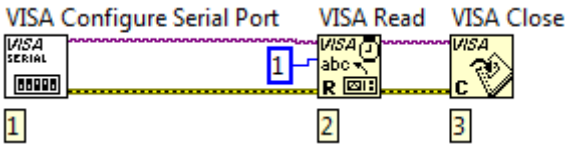
   Dengan 2 aturan tersebut, maka modifikasi program Arduino menjadi seperti Gambar 9.29, dan **Upload** kembali ke Arduino.

3. Kemudian, untuk membaca status tombol dari Arduino, maka LabVIEW membutuhkan 3 buah fungsi; fungsi **VISA Configure Serial Port** untuk mengatur seting komunikasi sekaligus membukanya, fungsi **VISA Read** untuk membaca data yang dikirimkan, dan fungsi **VISA Close** untuk menutup komunikasi.

4. Berhubung fungsi **VISA Read** membutuhkan input berapa banyak byte yang akan dibaca, dan untuk status sebuah tombol hanya dibutuhkan 1 karakter saja, maka beri nilai input byte angka 1 seperti Gambar 9.30.

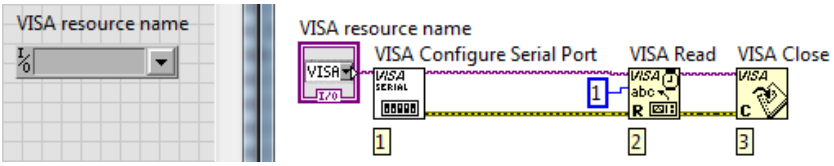
```
contoh_ketiga_Bab_4 §
void setup(){
  Serial.begin(9600);   // baudrate sebesar 9600
  pinMode(5,INPUT);  }  // kaki D5 sebagai INPUT
void loop(){
  byte tombol=digitalRead(5); // baca tombol
  if(tombol==HIGH) {          // jika ditekan
    Serial.write('1');}       // kirim ASCII 1
  else  {                     // jika tidak,
    Serial.write('0');}       // kirim ASCII 0
  delay(100);}                // beri tunda 100ms
```

Gambar 9.29 Modifikasi program Contoh #3 di Bab 4, gunakan Serial.write dan tambahkan delay(100)



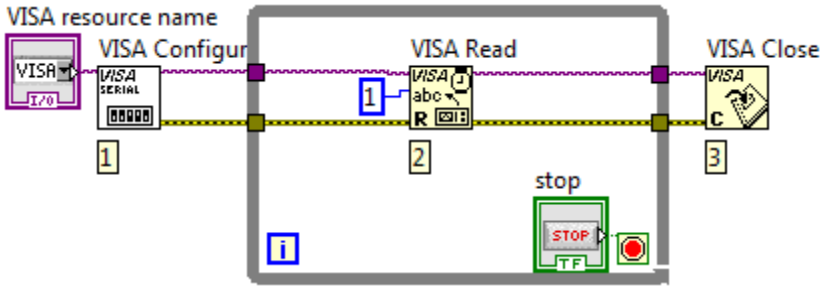
Gambar 9.30 Susunan fungsi Serial untuk menerima data 1 byte

5. Klik kanan kaki input ***VISA resource name*** di fungsi **VISA Configure Serial Port**, dan pilih **Create** kemudian **Control**, untuk bisa memilih saluran port yang sedang digunakan oleh Arduino dari Front Panel.



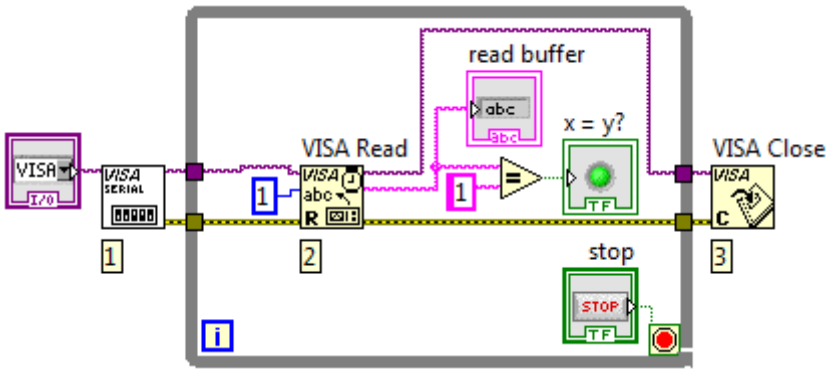
Gambar 9.31 Create Control pada kaki input VISA resource name

6. Tambahkan **While Loop** pada fungsi **VISA Read**, karena harus berulangkali membaca data selama komunikasi sedang berlangsung.



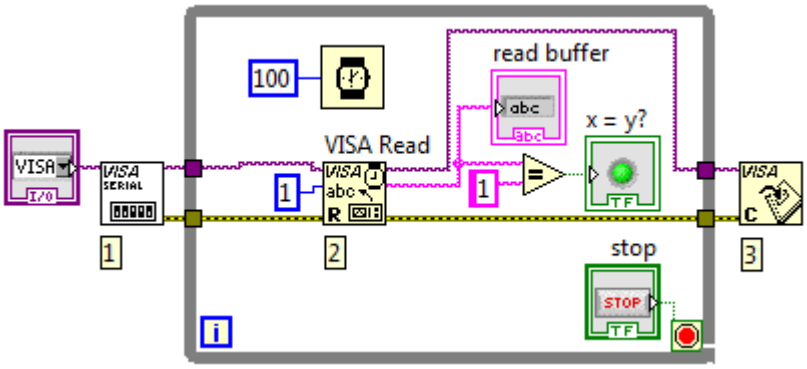
Gambar 9.32 Menambahkan While Loop pada VISA Read

7. Klik kanan kaki **read buffer VISA Read**, dan pilih **Create Indicator** untuk melihat data yang diterima LabVIEW. Kemudian tambahkan fungsi **Equal** untuk membandingkan output **read buffer** tersebut dengan **String Constant 1** atau **ASCII 1**, jika sama bernilai **True** dan jika tidak sama bernilai **False**. Nilai output inilah yang akan menyalakan **LED Indicator**. Untuk memunculkan **LED Indicator**, klik kanan pada output fungsi **Equal**, dan pilih **Create Indicator**.



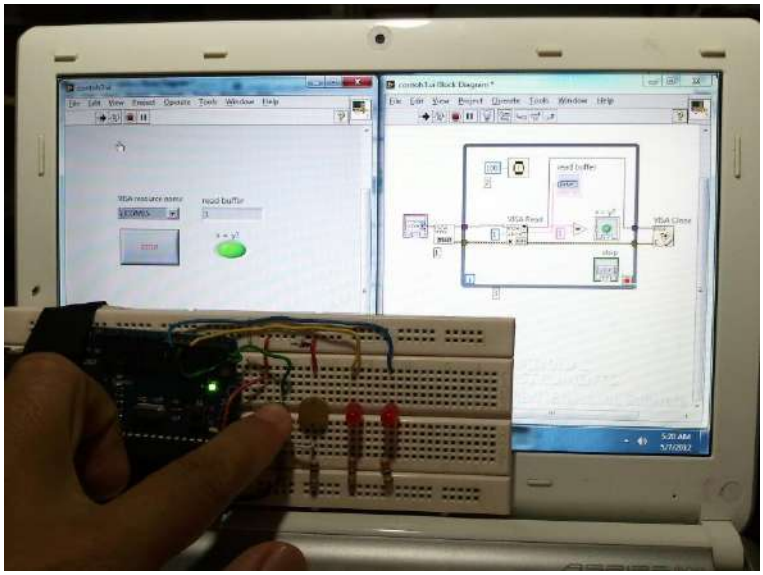
Gambar 9.33 Menambahkan LED Indicator pada output VISA Read

9. Berikutnya, tambahkan fungsi **Wait(ms)**, dan beri nilai input sebesar 100. Fungsi **Wait(ms)** ini sangat penting, karena tanpa diberi tunda waktu, data menjadi tidak terbaca karena tidak utuh atau terpotong. Dengan diberi tunda waktu, pembacaan data menjadi lebih baik.



Gambar 9.34 Penambahan Wait(ms) sebesar 100ms

10. Jalankan program dengan menekan tombol **Run**, pilih saluran **COM** yang digunakan, kemudian tekan tombol di kaki D5 Arduino, apakah **Round LED Indicator** berubah warnanya (menyala)? Lepas tombol, apakah **Round LED Indicator** kembali ke warna semula (padam)?



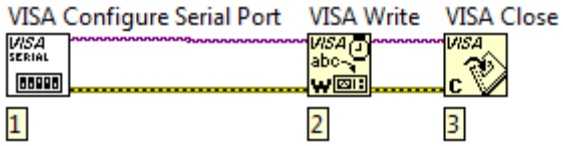
Gambar 9.35 Fot hasil program: tombol ditekan, Round LED indicator menyala, tombol dilepas, Round LED indicator padam

9.5 Contoh #4: Mengatur intensitas cahaya LED dari LabVIEW

Berikut langkah-langkah pembuatannya:

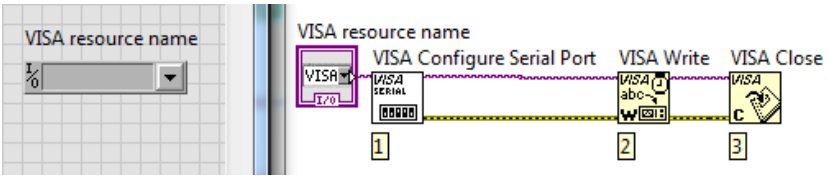
1. Ulangi langkah-langkah di Contoh #4 pada Bab 4 dari langkah pertama hingga langkah terakhir.

2. Untuk mengatur intensitas cahaya LED yang terhubung dengan Arduino, maka LabVIEW harus mengirimkan karakter ASCII 1 byte dengan nilai desimal dari 0 sampai 255; 0 untuk LED padam, dan 255 untuk LED menyala sangat terang. Untuk itu LabVIEW membutuhkan 3 buah fungsi; fungsi **VISA Configure Serial Port** untuk mengatur seting komunikasi sekaligus membukanya, fungsi **VISA Write** untuk mengirimkan karakter, dan **VISA Close** untuk menutup komunikasi.



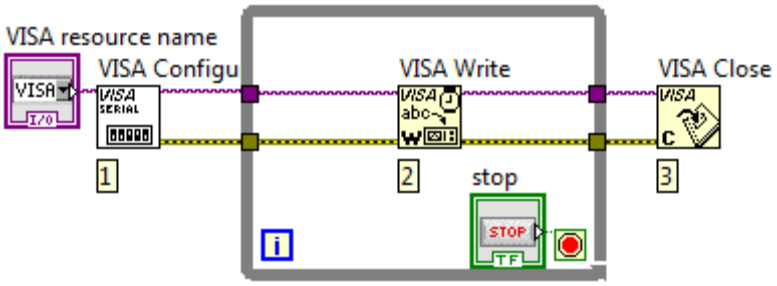
Gambar 9.36 Susunan fungsi Serial untuk mengirimkan data

2. Klik kanan dan pilih **Create Control** untuk kaki input ***VISA resource name*** di fungsi **VISA Configure Serial Port**. Untuk ***baudrate***, karena secara default sebesar 9600 (sama seperti baudrate Arduino), maka biarkan kaki inputnya tidak terhubung.



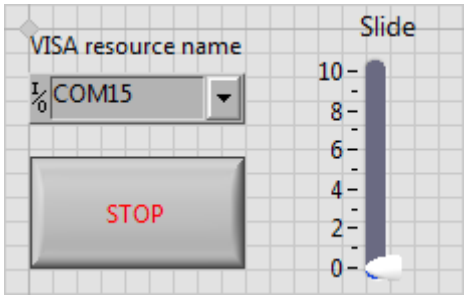
Gambar 9.37 Create Control pada kaki input VISA resource name

3. Berbeda dengan fungsi **VISA Configure Serial Port** dan **VISA Close** yang hanya cukup sekali dilakukan, fungsi **VISA Write** perlu dilakukan terus-menerus. Untuk itu tambahkan struktur perulangan **While Loop** pada fungsi **VISA Write**.



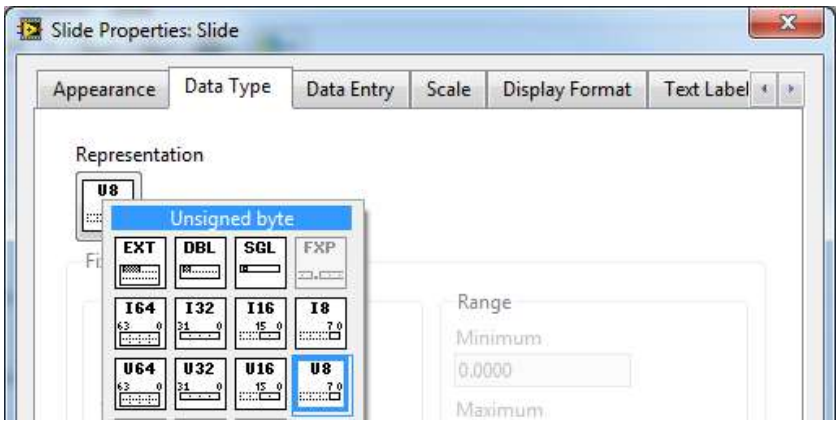
Gambar 9.38 Penambahan While Loop pada VISA Write

4. Berikutnya, untuk memudahkan pengiriman karakter dengan nilai desimal dari 0-255, ambil objek **Vertical Pointer Slide** dari kategori **Numeric Controls** dan dan tempatkan pada Front Panel.



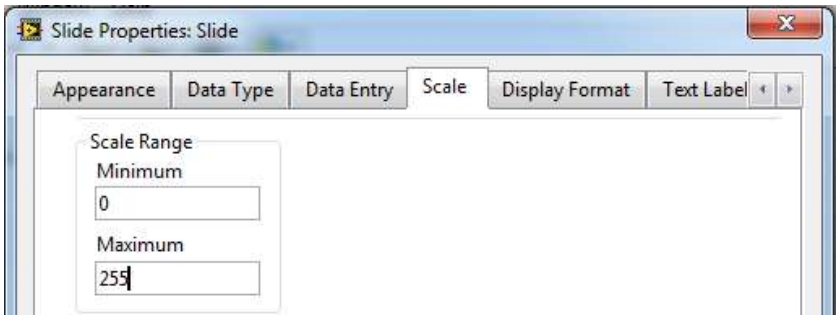
Gambar 9.39 Tempatkan objek Vertical Pointer Slide pada Front Panel

5. Klik kanan pada objek **Vertical Pointer Slide**, dan pilih **Properties**. Pada jendela **Slide Properties** yang terbuka, pilih tab **Data Types**. Kemudian klik pada icon di **Representation**, dan ganti tipe datanya dari **Double Precision** (**DBL**) menjadi **Unsigned Byte 8 bit** (**U8**).



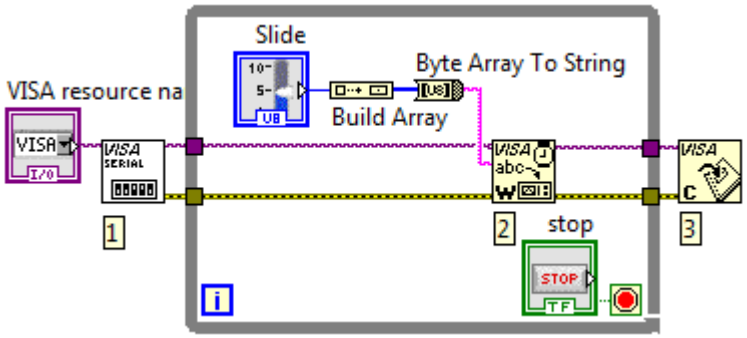
Gambar 9.40 Mengganti tipe data dari DBL menjadi U8

6. Masih pada jendela **Slide Properties**, pilih tab **Scale**, kemudian ganti nilai jangkauan **Maximum** dari 10 menjadi 255.



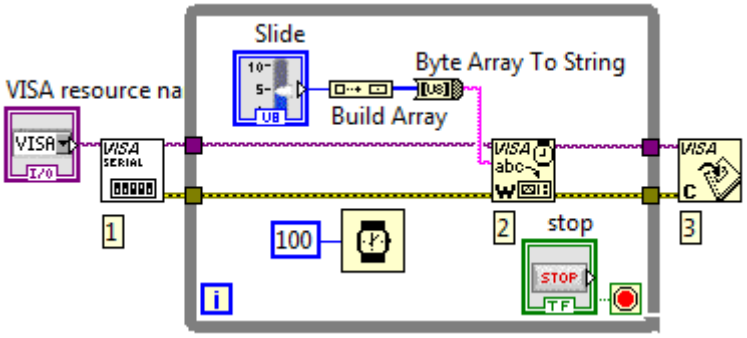
Gambar 9.41 Mengubah nilai jangkauan Maximum dari 10 menjadi 255

7. Pada jendela Block Diagram, karena input **write buffer VISA Write** harus bertipe data **String**, sedangkan nilai yang dihasilkan oleh **Vertical Pointer Slide** bertipe data **Byte (ASCII)**, maka lakukan pengubahan dari **Byte** menjadi **String**. Fungsi untuk mengubah tipe data dari Byte menjadi String ini membutuhkan 2 buah fungsi, yaitu fungsi **Byte Array To String**, dan fungsi **Build Array** untuk membuat data Byte menjadi Array Byte.



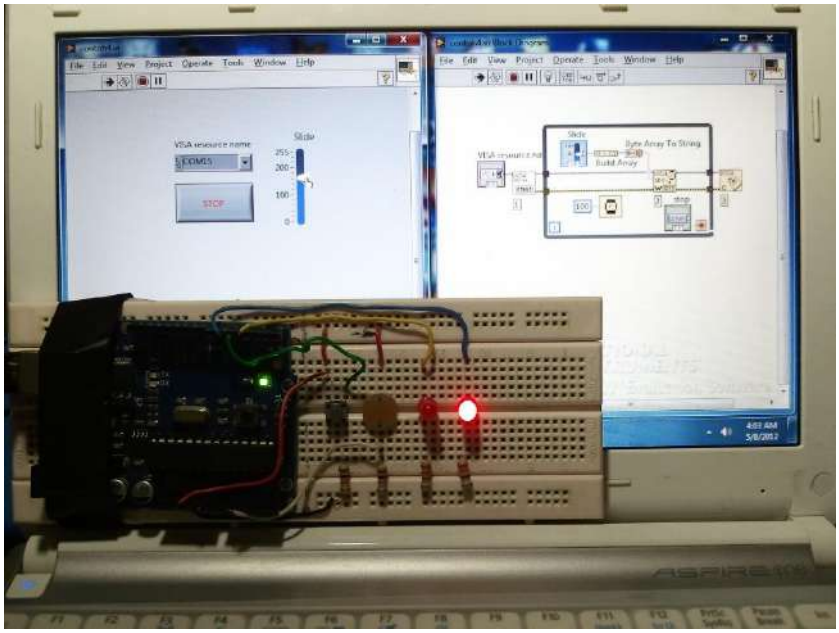
Gambar 9.42 Menambahkan fungsi Build Array dan Byte Array To String untuk menghubungkan data Vertical Pointer Slide ke Visa Write

8. Berikutnya, tambahkan fungsi **Wait(ms)**, dan beri nilai input sebesar 100. Fungsi **Wait(ms)** ini sangat penting, karena tanpa diberi tunda waktu, maka pengiriman data akan gagal atau tidak terkirimkan.



Gambar 9.43 Tambahkan Wait(ms) sebesar 100 untuk memastikan data yang dikirimkan LabVIEW benar-benar diterima Arduino secara utuh

9. Jalankan program dengan menekan tombol **Run**, pilih saluran **COM** yang digunakan, kemudian geser **Pointer** ke atas dan ke bawah, apakah LED di kaki Arduino D6 bisa diatur intensitas cahayanya menjadi semakin terang ketika **Pointer** digeser ke atas, dan menjadi semakin gelap ketika **Pointer** digeser ke bawah?



Gambar 9.44 Foto hasil program: bila Pointer dinaikkan ke atas, maka LED menyala semakin terang, bila diturunkan LED semakin gelap

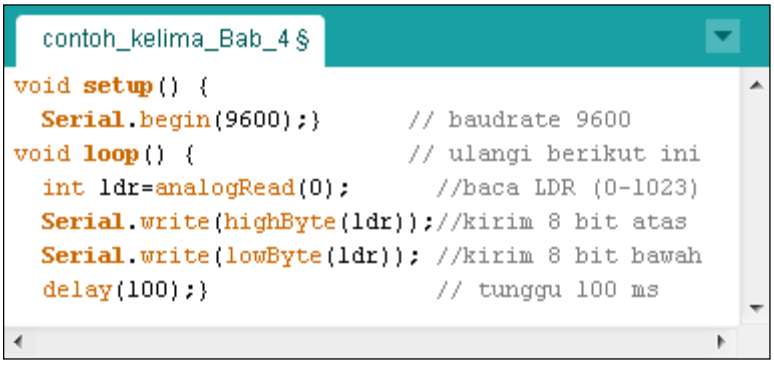
9.6 Contoh #5: LabVIEW membaca nilai analog LDR dan menampilkan nilainya dengan Gauge di Front Panel

Berikut langkah-langkah pembuatannya:

1. Ulangi langkah-langkah di Contoh #5 pada Bab 4 hingga **Serial Monitor** menampilkan angka yang akan semakin kecil ketika LDR ditutup dengan tangan, dan semakin besar ketika LDR dibuka.
2. Seperti telah dituliskan sebelumnya, untuk memastikan bahwa pengiriman data dari Arduino benar-benar diterima di LabVIEW dengan baik, maka gunakan aturan-aturan berikut ini dalam pemrograman di Arduino:

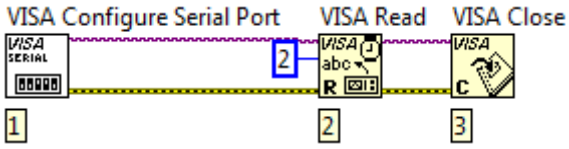
- Selalu gunakan perintah **Serial.write** untuk mengirimkan data, sebagai ganti **Serial.print** atau **Serial.println**, karena selain lebih efisien, juga memastikan bahwa data yang dikirimkan adalah karakter ASCII sebesar 1 byte setiap kali pengiriman.
- Untuk data yang dihasilkan oleh ADC Arduino, berhubung ukurannya 10 bit, maka harus dipecah menjadi 2 buah data 8 bit, dengan fungsi **highByte()** dan **lowByte()**.
- Tambahkan delay sebesar 100 ms untuk memastikan data yang terkirim benar-benar diterima oleh LabVIEW.

Dengan 3 aturan tersebut, maka modifikasi program Arduino menjadi seperti berikut, dan kemudian **Upload** kembali ke Arduino.



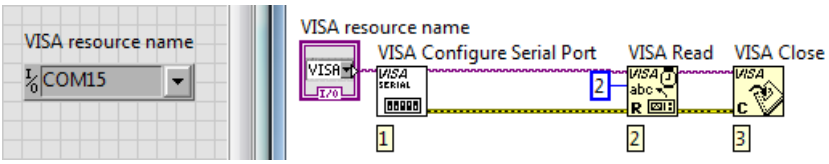
Gambar 9.45 Modifikasi program Contoh #5 di Bab 4, gunakan fungsi untuk memecah nilai dari 10 bit menjadi 2 buah data 8 bit (highByte dan lowByte), yang kemudian dikirimkan dengan Serial.write()

8. Kemudian, untuk membaca nilai analog LDR dari Arduino, maka LabVIEW membutuhkan 3 buah fungsi; fungsi **VISA Configure Serial Port** untuk mengatur seting komunikasi sekaligus membukanya, fungsi **VISA Read** untuk membaca data yang dikirimkan, dan fungsi **VISA Close** untuk menutup komunikasi. Karena ada 2 buah data 8 bit yang dikirim, maka beri angka 2 untuk input **byte count**.



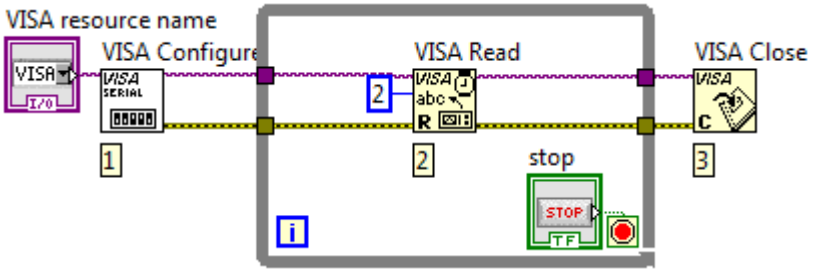
Gambar 9.46 Susunan fungsi Serial untuk membaca data 2 byte

9. Klik kanan dan pilih **Create Control** pada kaki input ***VISA resource name*** di fungsi **VISA Configure Serial Port**.



Gambar 9.47 Create Control pada kaki input VISA resource name

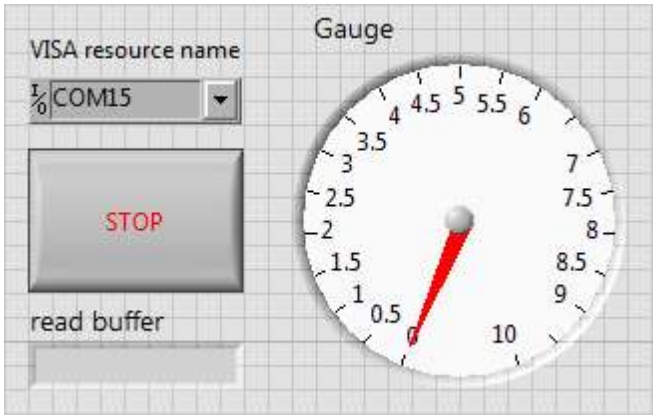
10. Tambahkan **While Loop** pada fungsi **VISA Read**, karena harus berulangkali membaca data selama komunikasi sedang berlangsung.



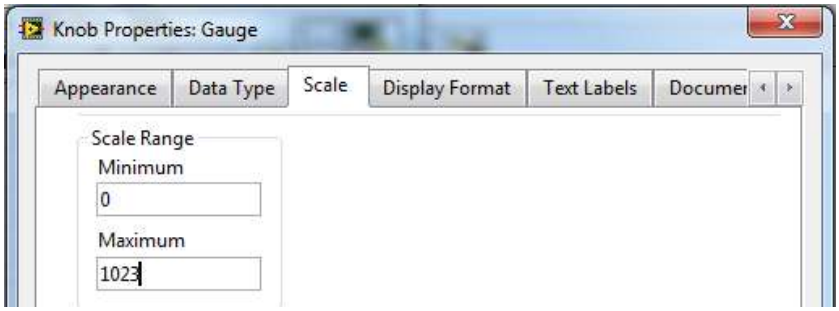
Gambar 9.48 Menambahkan While Loop pada VISA Read

11. Klik kanan kaki **read buffer VISA Read**, dan pilih **Create Indicator**, maka akan muncul **String Indicator**.

12. Pada jendela Front Panel, ambil objek **Gauge** dari kategori **Numeric Indicators** di palet Controls. Kemudian klik kanan **Gauge**, pilih **Properties**. Pada Tab **Scale**, ganti nilai jangkauan maksimum dari **10** menjadi **1023**.



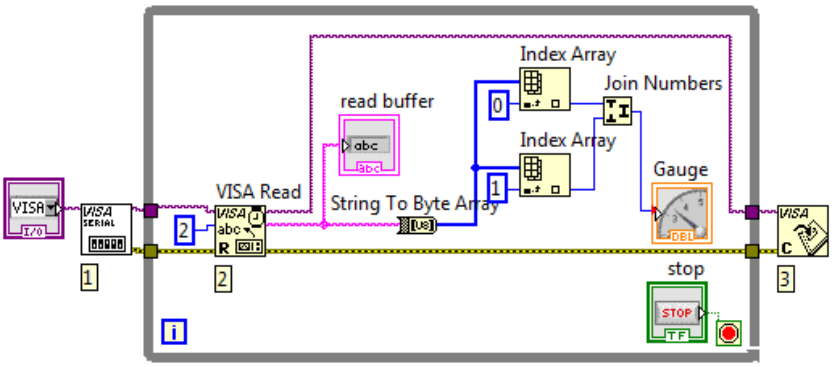
Gambar 9.48 Menambahkan objek Gauge di Front Panel



Gambar 9.49 Mengganti nilai jangkauan Maksimum dari 10 menjadi 1023

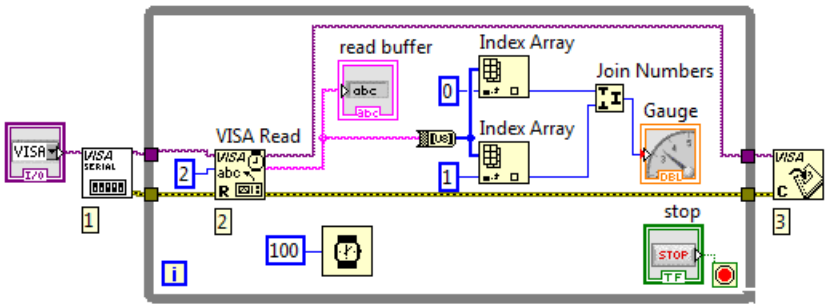
13. Data dari **VISA Read** bertipe **String (ASCII)**, sedangkan **Gauge** bertipe **Numeric**. Untuk menghubungkan keduanya, diperlukan fungsi konversi yang mengubah **String** menjadi **Numeric Byte**. Untuk itu tambahkan fungsi **String to Byte Array**.

14. Karena input **Gauge** harus data tunggal, bukan Array, maka lakukan pengubahan Array menjadi data tunggal dengan cara menambahkan 2 buah fungsi **Index Array** untuk mengambil data 8 bit pertama dan 8 bit kedua. Berikutnya tambahkan fungsi **Join Numbers** untuk menggabungkan keduanya sesuai dengan bobot nilainya.



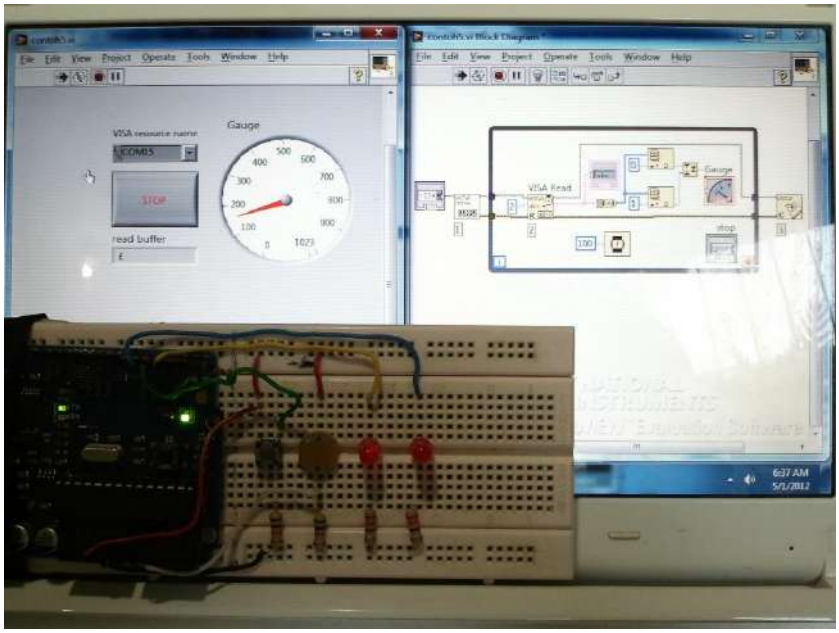
Gambar 9.50 Fungsi String To Byte Array, Index Array dan Join Numbers untuk mengubah data String menjadi data Numeric Gauge

11. Berikutnya, tambahkan fungsi **Wait(ms)**, dan beri nilai input sebesar 100. Fungsi **Wait(ms)** ini sangat penting, karena tanpa diberi tunda waktu, data menjadi tidak terbaca karena tidak utuh atau terpotong. Dengan diberi tunda waktu, pembacaan data menjadi lebih baik.



Gambar 9.51 Tambahkan Wait(ms) untuk memastikan LabVIEW benar-benar membaca data dari Arduino secara utuh

10. Jalankan program dengan menekan tombol **Run**, pilih saluran **COM** yang digunakan, kemudian perhatikan nilai yang ditunjukkan oleh jarum **Gauge** saat LDR ditutup dengan tangan, dibuka dan kemudian diberi cahaya, apakah nilainya juga berubah semakin besar ketika LDR mendapat cahaya yang semakin banyak?



Gambar 9.52 Foto hasil program: jarum Gauge menunjukkan nilai yang sebanding dengan besarnya cahaya yang diterima oleh LDR

9.7 Contoh #6: Gabungan

Berikut langkah-langkah pembuatannya:

1. Ulangi langkah-langkah di Contoh #6 pada Bab 4 dari langkah pertama hingga langkah terakhir.

2. Seperti telah dituliskan sebelumnya, untuk memastikan bahwa pengiriman data dari Arduino benar-benar diterima di LabVIEW dengan baik, maka gunakan aturan-aturan berikut ini dalam pemrograman di Arduino:

    - Selalu gunakan perintah **Serial.write** untuk mengirimkan data, sebagai ganti **Serial.print** atau **Serial.println**, karena selain lebih efisien, juga memastikan bahwa data yang dikirimkan adalah karakter ASCII sebesar 1 byte setiap kali pengiriman.

- Untuk data yang dihasilkan oleh ADC Arduino, berhubung ukurannya 10 bit, maka harus dipecah menjadi 2 buah data 8 bit, dengan fungsi **highByte()** dan **lowByte()**.
- Tambahkan delay sebesar 100 ms untuk memastikan data yang terkirim benar-benar diterima oleh LabVIEW.

Dengan 3 aturan tersebut, maka modifikasi program Arduino dan lakukan **Upload** kembali ke Arduino dengan program berikut ini:

contoh_keenam_Bab_9 §

```
void setup() {
  Serial.begin(9600);  // baudrate 9600
  pinMode(5,INPUT);    // kaki D5 input tombol
  pinMode(4,OUTPUT);   // kaki D4 output LED
  pinMode(6,OUTPUT);}  // kaki D6 output LED
void loop() {
  if(Serial.available()) {
    int datakul=Serial.read();
    int dataku2=Serial.read();
    if (datakul=='1')
      {digitalWrite(4,HIGH);}
      else
      {digitalWrite(4,LOW);}
    analogWrite(6,dataku2);}
  int tombol=digitalRead(5); // baca tombol
  int ldr=analogRead(0);     // baca LDR
  Serial.write(tombol);    // kirim nilai tombol
  Serial.write(highByte(ldr));// kirim nilai LDR
  Serial.write(lowByte(ldr)); // kirim nilai LDR
  delay(100);}          // tunggu 100 ms
```

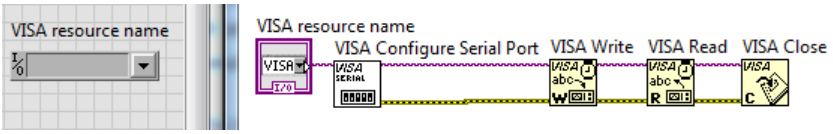
Gambar 9.53 Modifikasi program Contoh #6 di Bab 4

3. Untuk LabVIEW, karena LabVIEW tidak hanya menerima data, tetapi juga mengirimkan data, maka membutuhkan 4 buah fungsi; fungsi

VISA Configure Serial Port untuk mengatur seting komunikasi sekaligus membukanya, fungsi **VISA Write** untuk mengirimkan data ke Arduino, fungsi **VISA Read** untuk membaca data dari Arduino, dan fungsi **VISA Close** untuk menutup komunikasi.

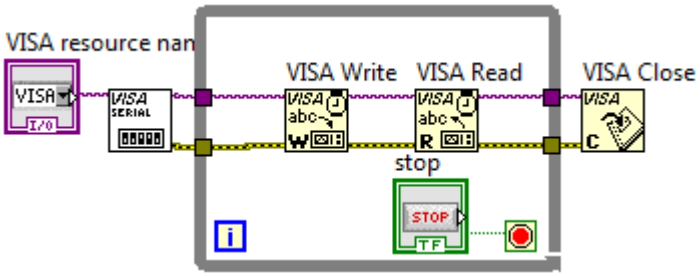
Gambar 9.54 Susunan fungsi Serial untuk mengirim dan menerima data

4. Klik kanan dan pilih **Create Control** pada kaki input ***VISA resource name*** di fungsi **VISA Configure Serial Port**.



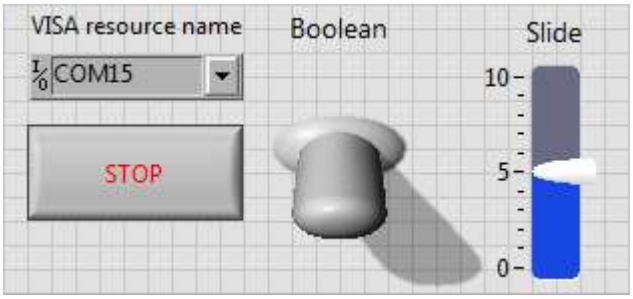
Gambar 9.55 Create Control pada kaki input VISA resource name

5. Tambahkan **While Loop** pada fungsi **VISA Write** dan **VISA Read**, karena keduanya harus bekerja terus-menerus selama komunikasi sedang berlangsung.



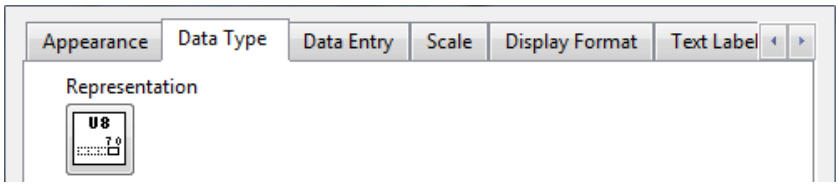
Gambar 9.56 Menambahkan While Loop pada VISA Write dan VISA Read

6. Pada jendela Front Panel, tempatkan objek **Toggle Switch** dan objek **Vertical Pointer Slide**.



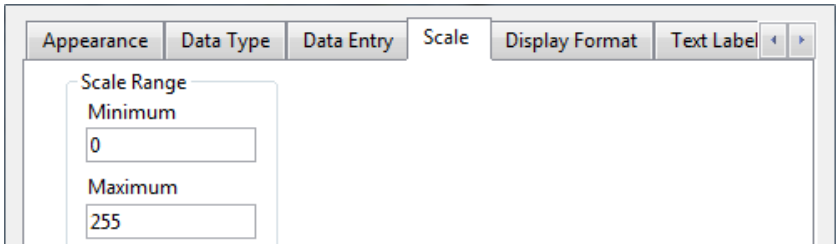
Gambar 9.57 Objek Toggle Switch dan Vertical Pointer Slide

7. Klik kanan pada objek **Vertical Pointer Slide**, pilih **Properties**, pilih tab **Data Types**. Kemudian klik pada icon **Representation**, ganti tipe data dari **Double Precision** (**DBL**) menjadi **Unsigned Byte 8 bit** (**U8**).



Gambar 9.58 Mengganti tipe data dari DBL menjadi U8

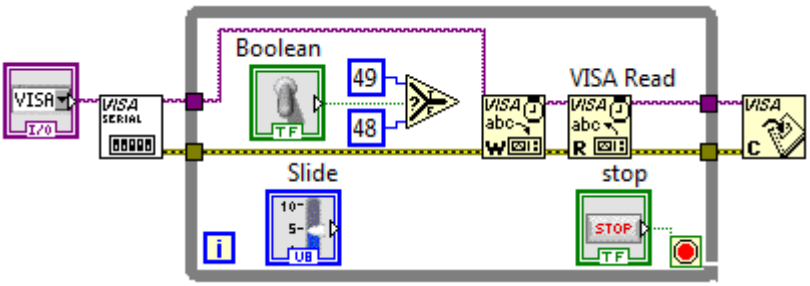
8. Masih pada jendela **Slide Properties**, pilih tab **Scale**, kemudian ganti nilai jangkauan **Maximum** dari 10 menjadi 255.



Gambar 9.59 Mengubah nilai jangkauan Maximum dari 10 menjadi 255

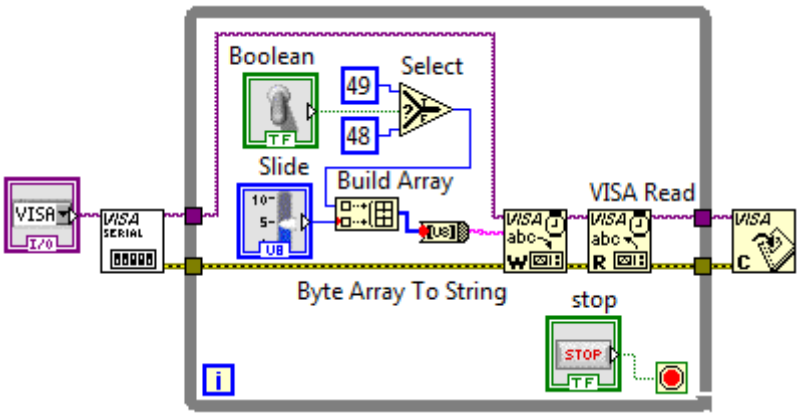
9. Pada jendela Block Diagram, tambahkan fungsi **Select** pada **Toggle Switch**. Kemudian ambil 2 buah **Numeric Constant**, dan beri nilai 49 dan 48, dan hubungkan masing-masing dengan kaki input **t** dan **f**

fungsi **Select**. Mengapa angka 49 dan 48? Karena angka 49 dan 48 merupakan angka desimal untuk kode ASCII 1 dan 0.



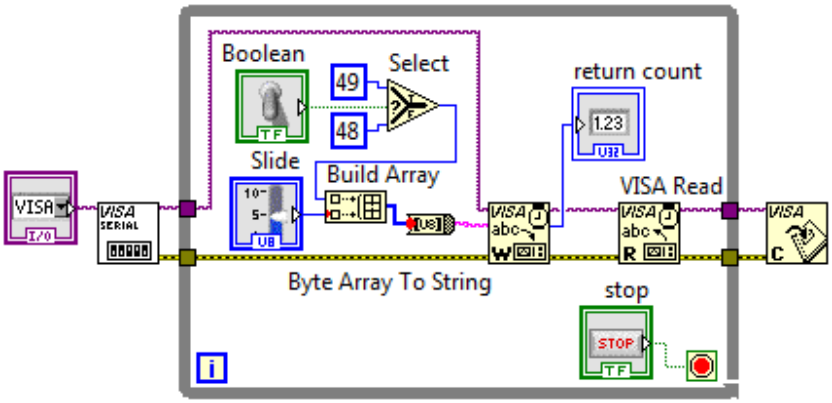
Gambar 9.60 Menambahkan fungsi Select dengan input angka 49 dan 48

10. Karena **Toggle Switch** dan **Vertical Pointer Slide** sama-sama mengirimkan data, maka gabungkan output keduanya menggunakan fungsi **Build Array**. Kemudian tambahkan fungsi **Byte Array to String**, dan hubungkan ke **write buffer VISA Write**.



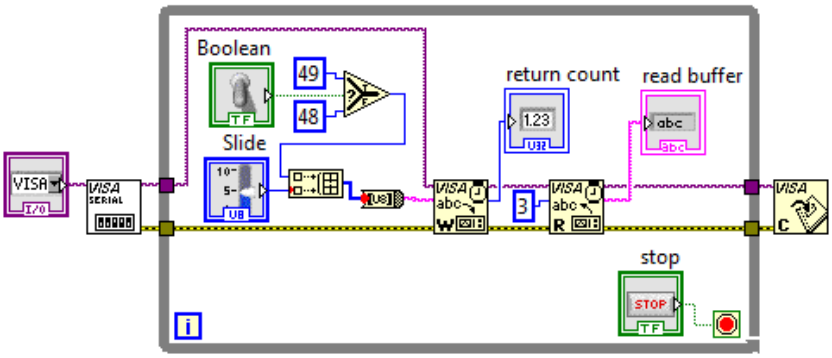
Gambar 9.61 Menambahkan fungsi Build Array dan Byte Array to String

11. Untuk memastikan jumlah data yang dikirimkan, tambahkan Indicator pada kaki output **return count VISA Write**, dengan cara meng-klik kanan kaki tersebut, dan memilih **Create Indicator**.



Gambar 9.59 Menambahkan Indicator pada kaki return count VISA Read

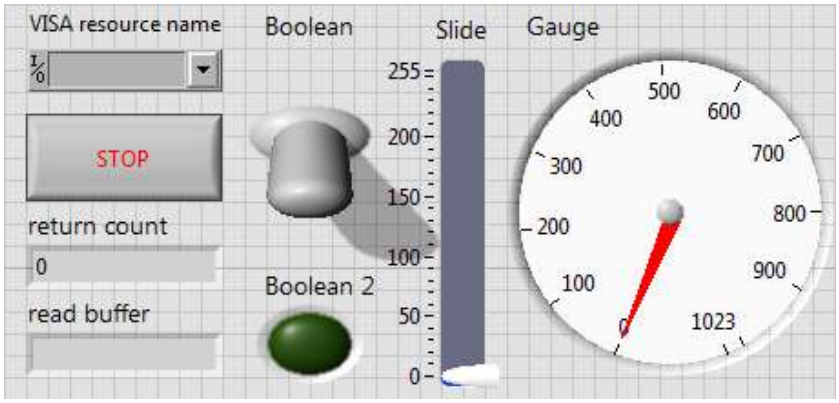
12. Untuk **VISA Read**, karena ada 3 buah data 8 bit yang dikirimkan Arduino, maka beri nilai 3 pada kaki input **byte count VISA Read**. Kemudian pada kaki **read buffer VISA Read**, tambahkan Indicator dengan meng-klik kanan kaki tersebut dan memilih **Create Indicator**.



Gambar 9.60 Menambahkan nilai 3 pada kaki input byte count dan Indicator pada kaki output read buffer VISA Read

13. Diinginkan dari 3 byte data yang diterima tersebut, 1 byte pertama ditampilkan dengan indicator **LED**, dan 2 byte berikutnya ditampilkan dengan indicator **Gauge**. Untuk itu ambil objek **Round LED** dan objek **Gauge** dari palet Controls dan tempatkan di Front Panel. Klik kanan

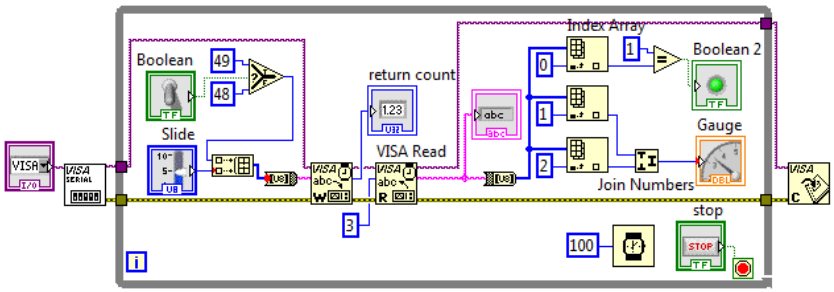
pada objek **Gauge**, pilih **Properties**, pada tab **Scale**, dan ganti nilai jangkauan maksimum dari 10 menjadi 1023.



Gambar 9.61 Menambahkan objek Round LED dan objek Gauge

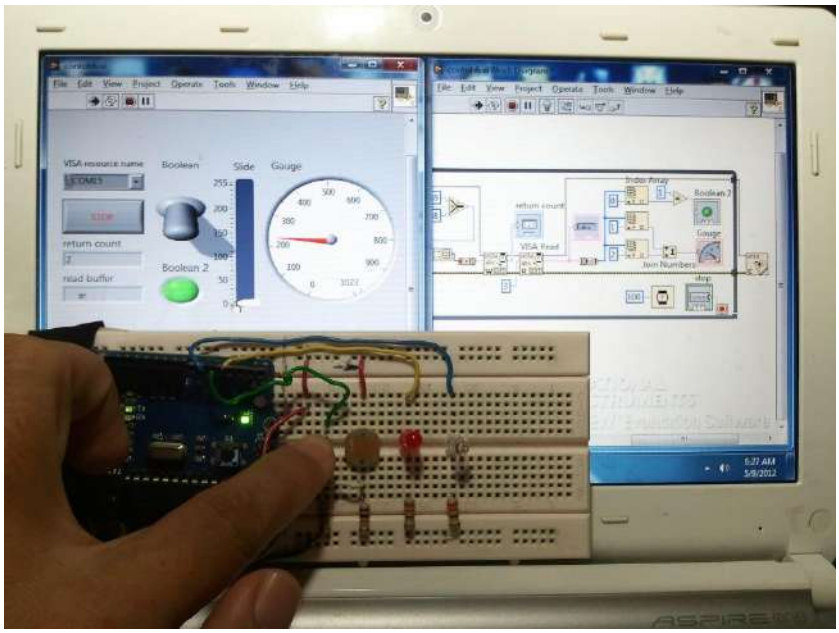
14. Untuk menghubungkan output fungsi **VISA Read** yang bertipe data **String** menjadi **Numeric**, tambahkan fungsi **String to Byte Array**, dilanjutkan dengan fungsi **Index Array**, untuk memecah Array menjadi 3 buah data tunggal. Untuk data tunggal pertama, bandingkan nilainya dengan 1 menggunakan fungsi **Equal**, dan hubungkan outputnya ke indicator **Round LED**. Untuk data tunggal kedua dan ketiga, tambahkan fungsi **Join Numbers** yang akan menggabungkan keduanya sesuai bobot nilainya, dan hubungkan outputnya ke indicator **Gauge**.

15. Berikutnya, tambahkan fungsi **Wait(ms)**, dan beri nilai input sebesar 100. Fungsi **Wait(ms)** ini sangat penting, karena tanpa diberi tunda waktu, data menjadi tidak terbaca karena tidak utuh atau terpotong.



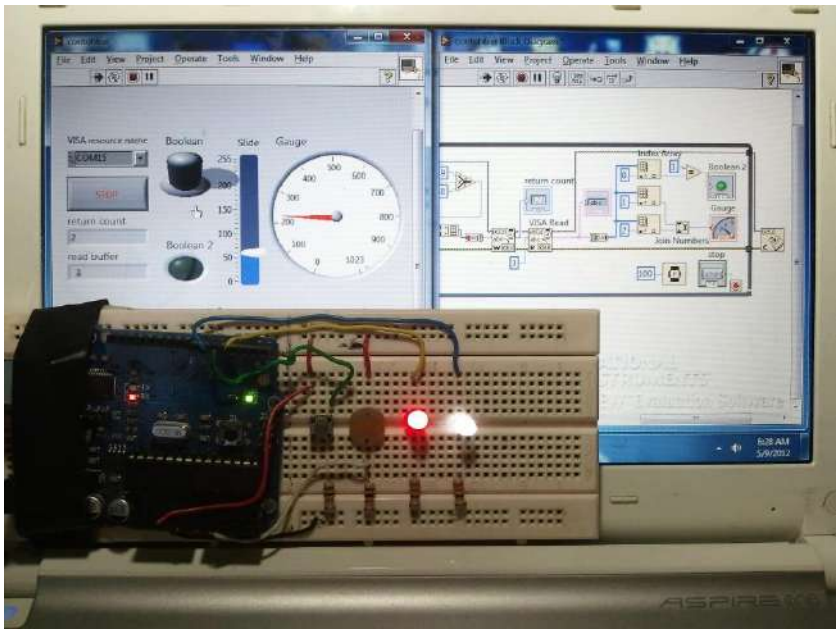
Gambar 9.62 program lengkap untuk aplikasi gabungan

16. Jalankan program dengan menekan tombol **Run**, pilih saluran **COM** yang digunakan, kemudian perhatikan nilai yang ditunjukkan oleh jarum **Gauge** saat LDR ditutup dengan tangan, dibuka dan kemudan diberi cahaya, apakah nilainya juga berubah semakin besar ketika LDR mendapat cahaya yang semakin banyak? Tekan dan lepas tombol di Arduino, apakah **Round LED** berubah warnanya?



Gambar 9.63 Tampak Round LED menyala ketika tombol ditekan

17. Kemudian tuas **Toggle Switch** ke atas dan ke bawah, apakah LED pertama di Arduino menyala dan kemudian padam? Geser **Vertical Pointer Slide** ke atas hingga maksimum, apakah LED kedua di Arduino menyala semakin terang? Geser **Vertical Pointer Slide** ke bawah hingga minimum, apakah LED kedua di Arduino semakin redup hingga akhirnya padam?



Gambar 9.64 Tampak LED menyala ketikaToggle Switch dituas ke atas

BAB 10
FIRMATA

10.1 Apa itu Firmata

Firmata adalah sebuah protocol yang ditulis pada Arduino, untuk memudahkan komunikasi serial Arduino dengan software-software komputer, termasuk LabVIEW. Mudah dalam arti pemrograman tidak lagi dilakukan di kedua sisi, tetapi hanya di satu sisi, yaitu di software komputer saja, sedangkan di Arduino, tidak perlu lagi diprogram, selama **Firmata** masih ada di dalamnya.

Keuntungan menggunakan **Firmata**:

1. Kecepatan komunikasi yang bisa mencapai **115200** bps (bandingkan dengan kecepatan default komunikasi serial, yang hanya **9600** bps).
2. **Firmata** ini membuat Arduino seperti layaknya sebuah **DAQ Card**, di mana semua kaki IO Arduino telah difungsikan secara tetap, sehingga hanya tinggal memasukkannya pada program di software komputer. Dengan cara seperti ini, setiap kali program dimodifikasi, atau butuh penambahan kaki IO, tidak perlu lagi memprogram Arduino (seperti halnya dilakukan di Bab 9), tetapi cukup memprogram di software komputer saja.

3. Untuk aplikasi komunikasi serial yang membutuhkan penggunaan kaki IO Arduino yang cukup banyak, baik difungsikan sebagai input maupun output, digital maupun analog (termasuk PWM), atau yang membutuhkan penambahan fungsi-fungsi komunikasi yang lebih rumit seperti I2C dan SPI, maka penggunaan **Firmata** lebih menguntungkan, karena semuanya sudah tersedia fungsinya, hanya tinggal mengaktifkannya. Sebaliknya bila tanpa **Firmata**, maka pembaca akan berurusan dengan penulisan kode program yang panjang, baik di sisi Arduino maupun di sisi komputer.

Kerugian menggunakan **Firmata**:

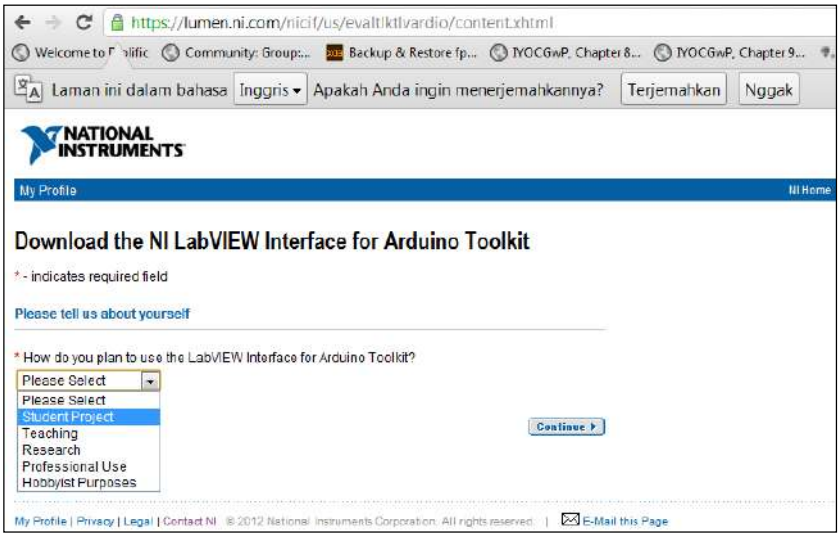
1. Kaki **IO** Arduino tidak lagi fleksibel, karena telah difungsikan secara tetap, sehingga untuk aplikasi-aplikasi khusus, atau untuk fungsi-fungsi yang belum tersedia, seperti pembacaan sensor ultrasonic, dimana membutuhkan pembacaan pulsa yang sangat cepat, akan sulit dilakukan. Namun ke depannya ada kemungkinan bisa dilakukan, apabila fungsinya sudah dimasukkan ke dalam Firmata.

10.2 Instalasi Firmata

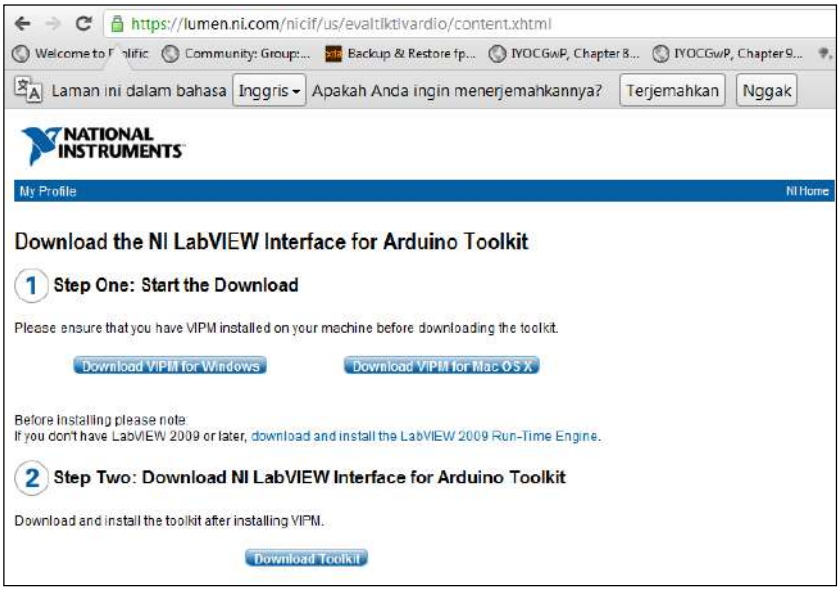
Fungsi-fungsi Firmata Arduino untuk LabVIEW dapat ditemukan secara gratis di Internet dengan nama **LabVIEW Interface for Arduino** (**LIFA**).

Berikut langkah-langkah untuk menginstal **LIFA**:

1. Mula-mula, gunakan mesin pencari Google, ketikkan kata kunci “labview interface for arduino”, kemudian buka alamat yang ditunjukkan Google, maka muncul halaman seperti Gambar 10.1.
2. Pilih rencana penggunaan LIFA, misalnya untuk **Student Project**. Kemudian klik tombol **Continue**, maka akan muncul langkah-langkah untuk men-download LIFA seperti ditunjukkan pada Gambar 10.2.



Gambar 10.1 Memilih rencana penggunaan LIFA



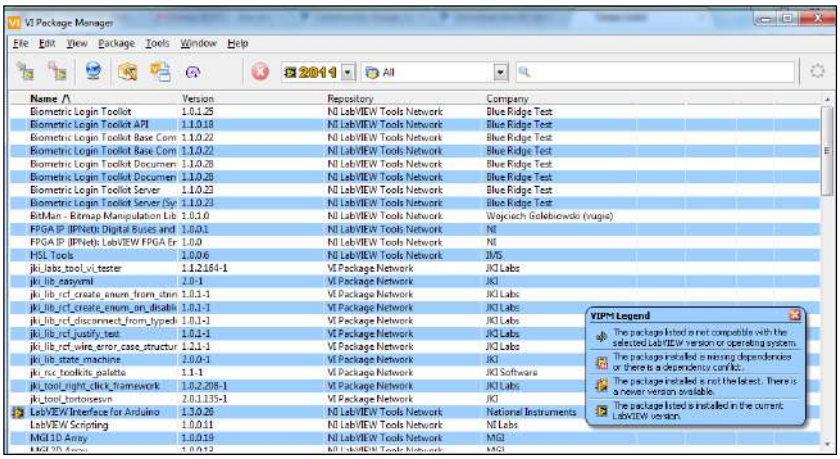
Gambar 10.2 Langkah-langkah download LIFA

3. Klik pada tombol **Download VIPM for Windows**.

4. Setelah download berhasil dilakukan, klik 2 kali pada file tersebut untuk meng-instal **VIPM**. **VIPM** adalah software aplikasi yang membantu mendownload library atau paket yang diperlukan dari internet dan sekaligus menginstalnya pada software LabVIEW.

Gambar 10.3 VIPM (VI Package Manager), untuk membantu mendownload dan menginstal paket yang diperlukan

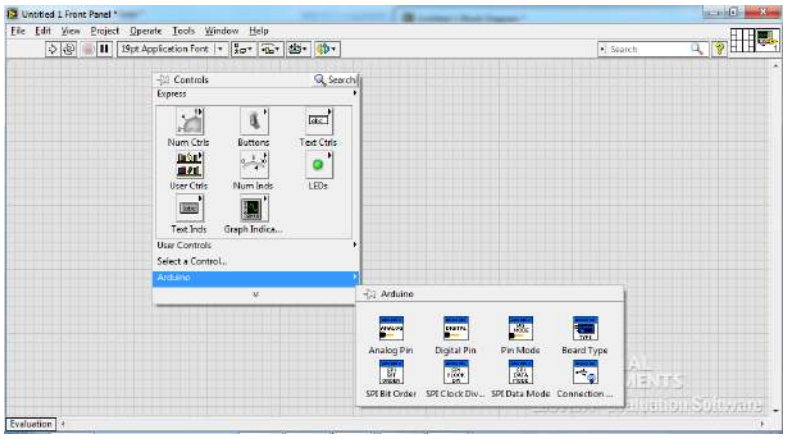
5. Setelah instalasi berhasil dilakukan, jalankan aplikasi VIPM. Maka akan muncul daftar paket yang tersedia dan bisa diinstal.



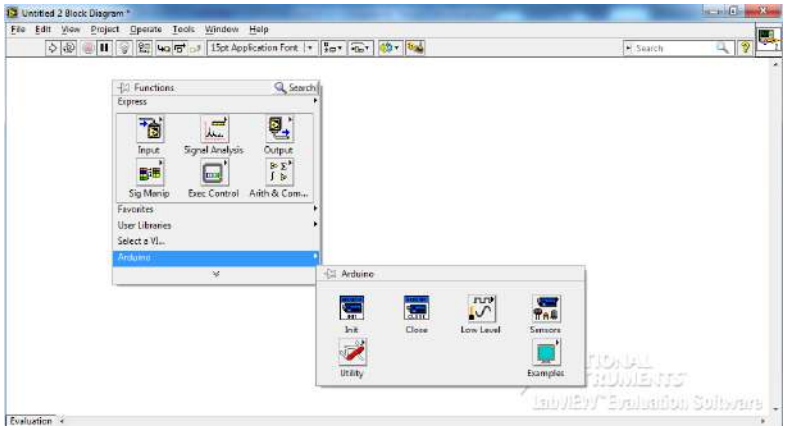
Gambar 10.4 Daftar paket yang tersedia dan bisa diinstal

6. Pilih paket **LabVIEW Interface for Arduino**, kemudian klik pada tombol **Instal and Upgrade Package(s)** di kiri atas Toolbar. Maka VIPM akan mengambil paket tersebut dari Internet, dan kemudian mendownload serta menginstalnya pada software LabVIEW.

7. Setelah instalasi selesai, pembaca dapat menemukan paket atau toolkit Arduino tersebut di palet Controls dan di palet Functions.



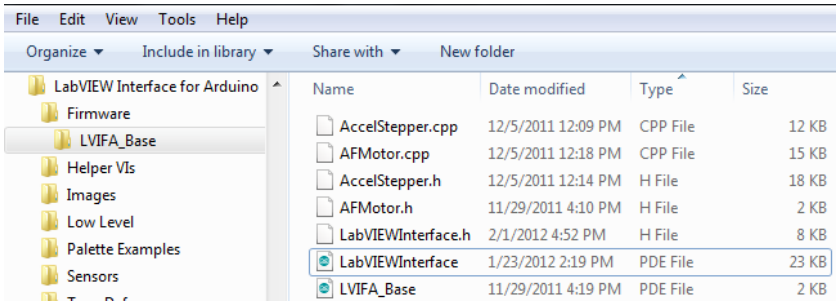
Gambar 10.5 Toolkit Arduino dapat ditemukan di palet Controls



Gambar 10.6 Fungsi Firmata Arduino dapat diambil di palet Function

8. Setelah fungsi-fungsi Firmata atau LabVIEW Interface for Arduino toolkit telah terpasang pada LabVIEW, maka berikutnya adalah memasang atau meng-Upload kode Firmata (*LVIFA Firmware*) pada Arduino. Upload kode Firmata ini hanya dilakukan sekali, dan kemudian untuk seterusnya, Arduino tidak perlu lagi diprogram, cukup di LabVIEW saja. Kode Firmata ini dapat diambil di direktori:

   C:\Program Files\National Instruments\LabVIEW2011\vi.lib\LabVIEW Interface for Arduino\Firmware\LVIFA_Base



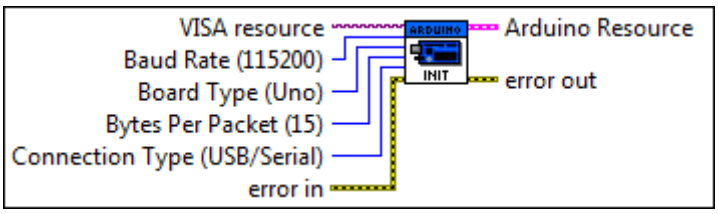
Gambar 10.7 File kode Firmata untuk Arduino

9. Klik 2 kali file **LVIFA_Base**, maka akan membuka software Arduino. Kompilasi file dengan menekan tombol **Verify**. Setelah selesai, tekan tombol **Upload**. Jangan lupa untuk menghubungkan Arduino dengan komputer sebelum tombol **Upload** ditekan. Apabila muncul pesan **Done Uploading**, maka Firmata telah berhasil dipasang di Arduino.

10.3 Fungsi-fungsi Toolkit Arduino

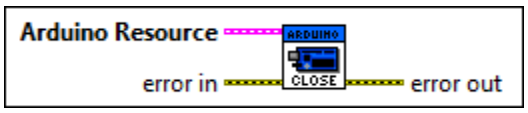
Berikut keterangan singkat dari beberapa fungsi toolkit Arduino (LIFA):

9. **LIFA Init**. Fungsi ini digunakan untuk mengatur seting komunikasi dengan Arduino, termasuk port yang digunakan (***VISA resource***), tipe Arduino (***Board Type***), dll, dan kemudian mengkoneksikannya.



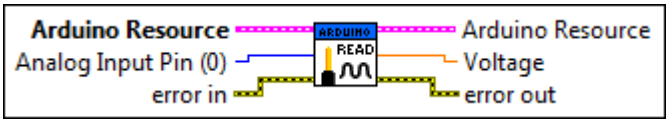
Gambar 10.8 LIFA Init

10. **LIFA Close**. Fungsi ini untuk menutup komunikasi dengan Arduino.



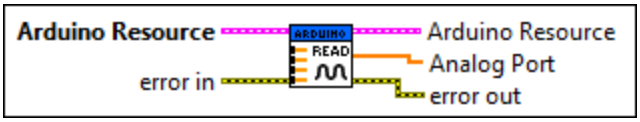
Gambar 10.9 LIFA Close

11. **LIFA Analog Read Pin**. Fungsi ini digunakan untuk membaca nilai tegangan analog (0-5V) pada kaki analog Arduino yang sesuai dengan **Analog Input Pin**.



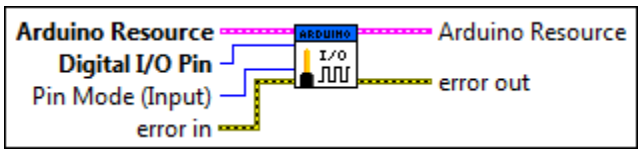
Gambar 10.10 LIFA Analog Read Pin

12. **LIFA Analog Read Port**. Fungsi ini digunakan untuk membaca semua tegangan analog (0-5V) pada kaki port analog Arduino (A0 – A5).



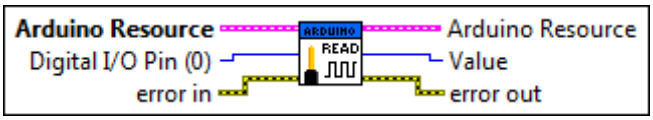
Gambar 10.11 LIFA Analog Read Port

13. **LIFA Set Digital Pin Mode**. Fungsi ini digunakan untuk mengatur kaki digital Arduino untuk dijadikan sebagai INPUT atau OUTPUT.



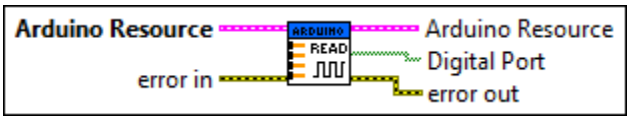
Gambar 10.12 LIFA Set Digital Pin Mode

14. **LIFA Digital Read Pin**. Fungsi ini digunakan untuk membaca nilai digital (0/1) pada kaki digital Arduino yang sesuai dengan **Digital I/O Pin**, asalkan kaki tersebut diset INPUT dengan **Set Digital Pin Mode**.



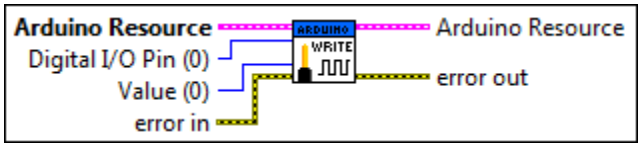
Gambar 10.13 LIFA Digital Read Pin

15. **LIFA Digital Read Port**. Fungsi ini digunakan untuk membaca nilai digital pada kaki port digital Arduino (D0 – D13), asalkan kaki-kaki tersebut diset (dengan fungsi **Set Digital Pin Mode**) sebagai INPUT.



Gambar 10.14 LIFA Digital Read Port

16. **LIFA Digital Write Pin**. Fungsi ini untuk mengirimkan nilai digital (0/1) ke kaki digital Arduino yang sesuai dengan **Digital I/O Pin**, asalkan kaki tersebut dibuat OUTPUT dengan **Set Digital Pin Mode**.

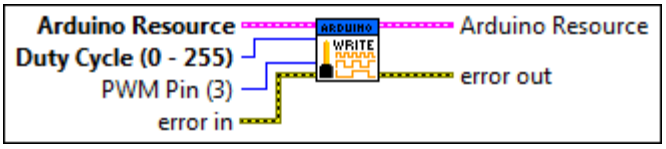


Gambar 10.15 LIFA Digital Write Pin

17. **LIFA Digital Write Port**. Fungsi ini digunakan untuk mengirimkan nilai digital pada kaki port digital Arduino (D0 – D13), asalkan kaki-kaki tersebut dijadikan OUTPUT dengan fungsi **Set Digital Pin Mode**.

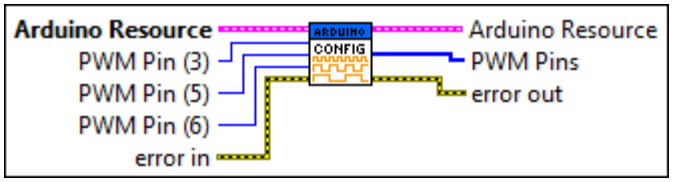
Gambar 10.16 LIFA Digital Write Port

18. **LIFA PWM Write Pin**. Fungsi ini digunakan untuk mengirimkan nilai PWM atau **Duty Cycle** (dengan nilai antara 0-255) ke sebuah kaki PWM yang sesuai dengan **PWM Pin**.



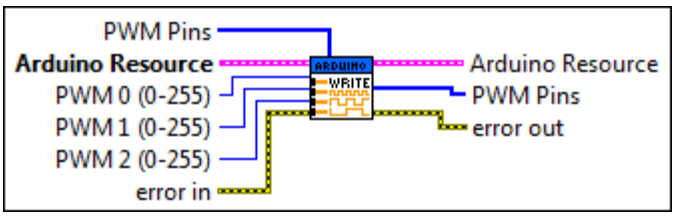
Gambar 10.17 LIFA PWM Write Pin

19. **LIFA PWM Configure Port**. Fungsi ini digunakan untuk mengatur kaki-kaki digital Arduino menjadi kaki PWM, di mana output fungsi ini digunakan untuk input **PWM Pins** pada fungsi **PWM Write Port**.



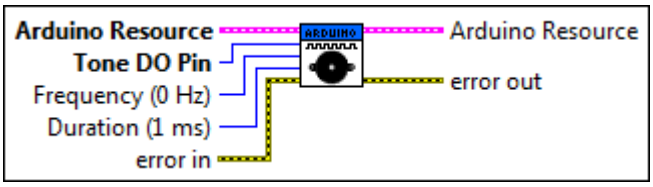
Gambar 10.18 LIFA PWM Configure Port

20. **LIFA PWM Write Port**. Fungsi ini digunakan untuk mengirimkan nilai PWM (0-255) pada kaki-kaki yang sesuai dengan **PWM Pins**, dimana kaki-kaki **PWM Pins** ini diperoleh dari fungsi **PWM Configure Port**.



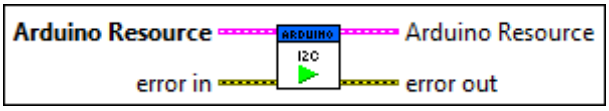
Gambar 10.19 LIFA PWM Write Port

21. **LIFA Tone**. Fungsi ini digunakan untuk membangkitkan gelombang kotak dengan frekuensi yang dapat diatur, pada duty cycle sebesar 50%. Fungsi ini dapat dihentikan dengan membuat frekuensinya sebesar 0 Hz. Fungsi ini hanya dapat digunakan untuk satu kaki pada satu waktu. Untuk lebih dari satu kaki, maka harus bergantian.



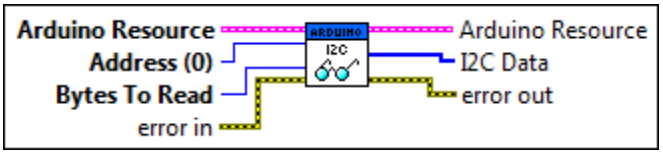
Gambar 10.20 LIFA Tone

22. **LIFA I2C Init**. Fungsi ini digunakan untuk membuka komunikasi I2C.



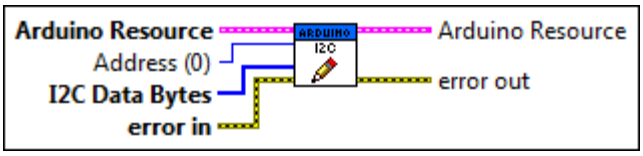
Gambar 10.21 LIFA I2C Init

23. **LIFA I2C Read**. Fungsi ini digunakan untuk membaca data dari I2C.



Gambar 10.22 LIFA I2C Read

24. **LIFA I2C Write**. Fungsi ini digunakan untuk mengirimkan data ke I2C.

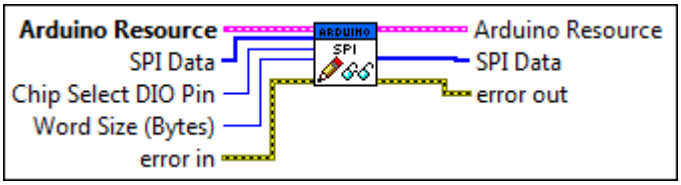


Gambar 10.23 LIFA I2C Write

25. **LIFA SPI Init**. Fungsi ini digunakan untuk membuka komunikasi SPI.

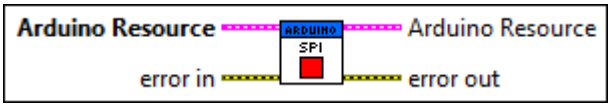
Gambar 10.24 LIFA SPI Init

26. **LIFA SPI Send Receive**. Fungsi ini digunakan untuk mengirim/ menerima data pada komunikasi SPI.



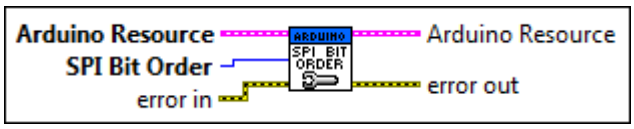
Gambar 10.25 LIFA SPI Send Receive

27. **LIFA SPI Close**. Fungsi ini untuk mematikan komunikasi SPI.



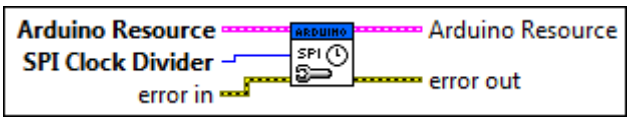
Gambar 10.26 LIFA SPI Close

28. **LIFA SPI Set Bit Order**. Fungsi ini digunakan untuk mengatur urutan bit pada komunikasi SPI, apakah bit MSB dulu atau LSB dulu.



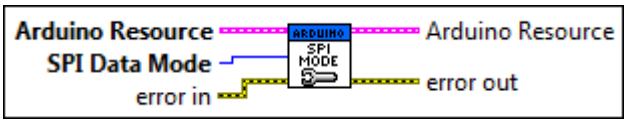
Gambar 10.27 LIFA SPI Set Bit Order

29. **LIFA SPI Set Clock Divider**. Fungsi ini untuk mengatur Clock Divider.



Gambar 10.28 LIFA SPI Set Clock Divider

30. **LIFA SPI Set Data Mode**. Fungsi ini digunakan untuk mengatur mode data SPI, yaitu: **Mode 0** (polaritas 0 fase 0), **Mode 1** (polaritas 0 fase 1), **Mode 2** (polaritas 1 fase 0) atau **Mode 3** (polaritas 1 fase 1).

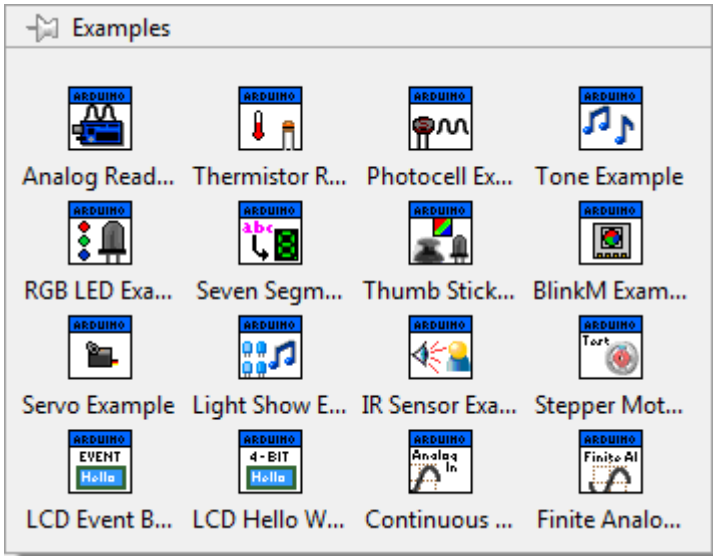


Gambar 10.29 LIFA SPI Data Mode

10.4 Contoh Aplikasi Firmata

Contoh-contoh aplikasi Firmata ini dapat pembaca temukan dengan membuka palet **Functions** di jendela Block Diagram, kemudian pilih kategori **Arduino**, dilanjutkan dengan **Examples**, maka akan terlihat berbagai contoh aplikasi, seperti ditunjukkan pada Gambar 10.30.

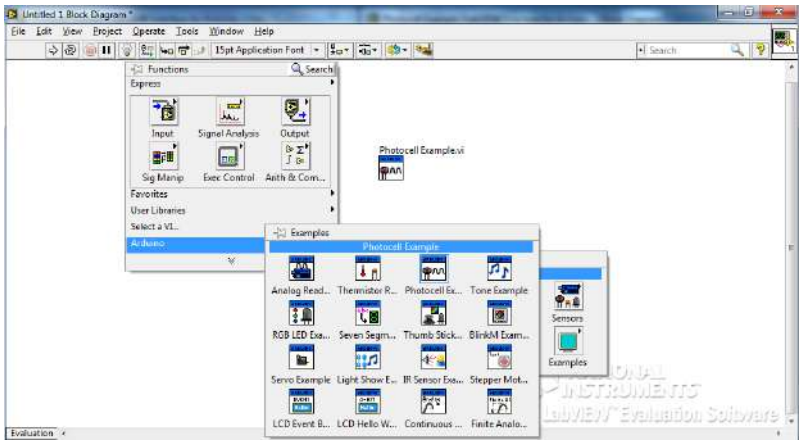
Contoh aplikasi yang tersedia mulai dari pembacaan sinyal analog potensio, pembacaan beberapa sensor (thermistor, LDR, joystick dan infrared) hingga pengendalian LED, buzzer dan tampilan (7-segmen dan LCD) serta aktuator (motor servo dan stepper).



Gambar 10.30 Contoh-contoh aplikasi pada kategori Examples

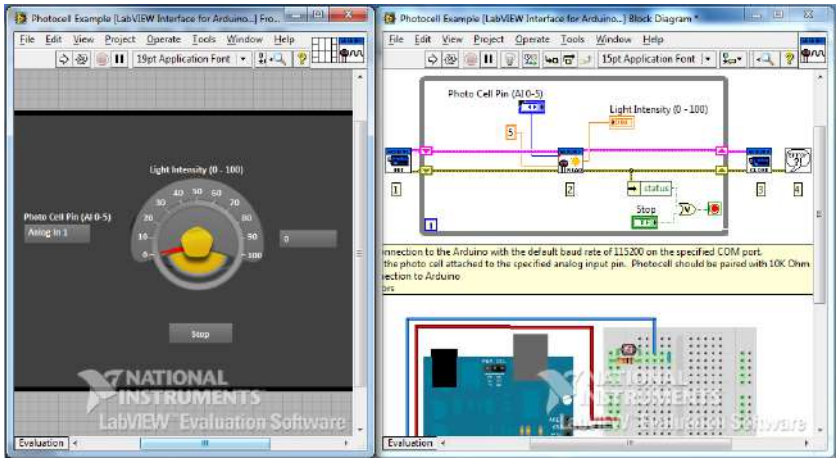
Berikut langkah-langkah untuk menjalankan contoh aplikasi tersebut:

1. Ambil salah satu icon dalam kategori **Examples**, dan tempatkan di jendela Block Diagram. Sebagai contoh di sini, diambil icon **Photocell Example** (photocell = LDR).



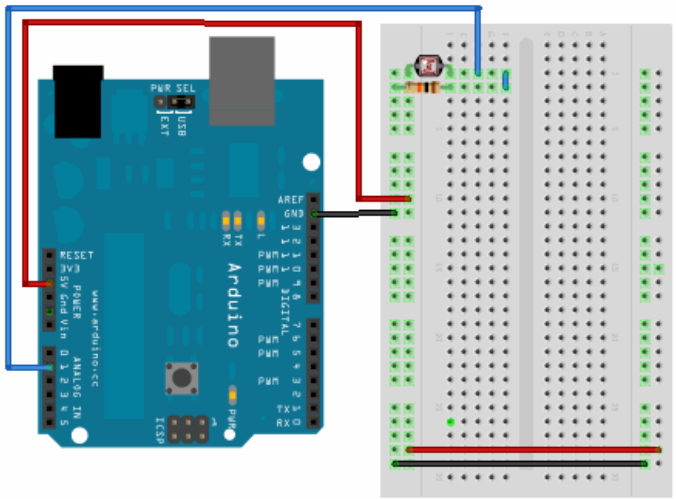
Gambar 10.31 Mengambil icon Photocell Example

2. Klik 2 kali pada icon tersebut untuk membuka jendela Front Panel dan Block Diagram dari icon tersebut.



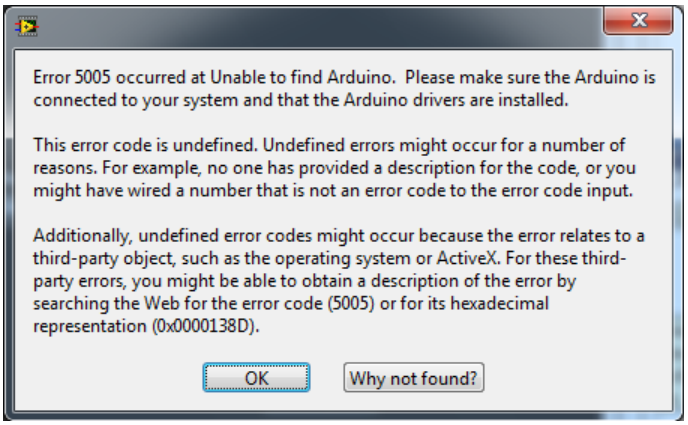
Gambar 10.32 Membuka Front Panel dan Block Diagram icon Photocell

3. Susun rangkaian alat sesuai dengan gambar pengawatan rangkaian pada jendela Block Diagram.



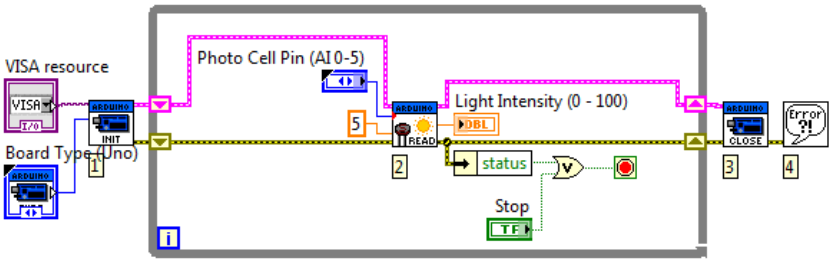
Gambar 10.33 Gambar pengawatan rangkaian pada Block Diagram

4. Jalankan program dengan menekan tombol Run. Namun sebelumnya, pastikan bahwa kode Firmata (*LVIFA Firmware*) telah di-Upload ke dalam Arduino, dan kabel USB Arduino menghubungkan Arduino dengan komputer. Maka ketika tombol Run ditekan, akan muncul pesan error berikut ini:



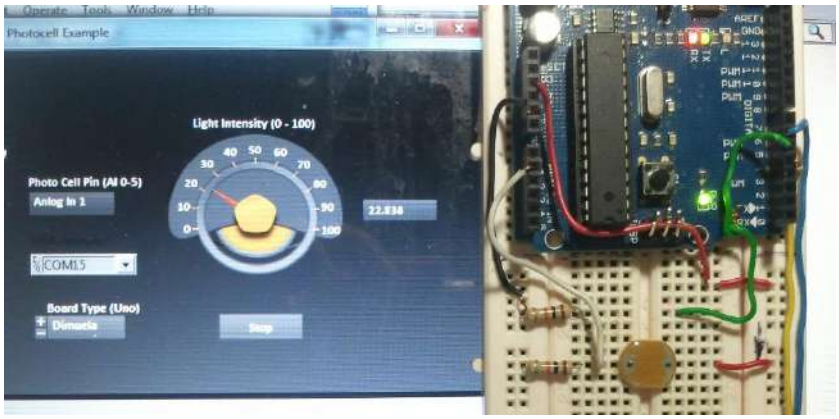
Gambar 10.34 Muncul Error akibat port dan tipe Arduino tidak diketahui

5. Error di atas terjadi karena program belum memasukkan saluran port serial dan tipe Arduino yang digunakan secara benar. Untuk itu tambahkan pada Block Diagram, yaitu pada icon **LIFA Init**, **Create Control** pada kaki **VISA Resource** dan kaki **Board Type**.



Gambar 10.35 Menambahkan Create Control pada kaki VISA resource dan kaki Board Type pada fungsi LIFA Init

6. Buka Front Panel, jalankan program kembali dengan menekan tombol **Run**. Kemudian pada kotak ***drop down***, pilih saluran COM yang sesuai dan tipe Board Arduino yang digunakan, serta pastikan **Photocell Pin** AI **0-5** sesuai dengan kaki Arduino yang terhubung dengan LDR. Maka seharusnya jarum **Light Intensity** akan bergerak mengikuti besarnya cahaya yang diterima oleh LDR.



Gambar 10.36 Foto hasil program: jarum pada Ligh Intensity menunjukkan nilai sebanding dengan cahaya yang diterima oleh LDR

Catatan: perhatikan bahwa dengan Firmata, komunikasi menjadi sangat cepat dan tetap stabil, sekalipun di dalam **While Loop** tidak diberi tunda waktu **Wait(ms)** dan dengan **Baudrate** sebesar 115200 bps (default).

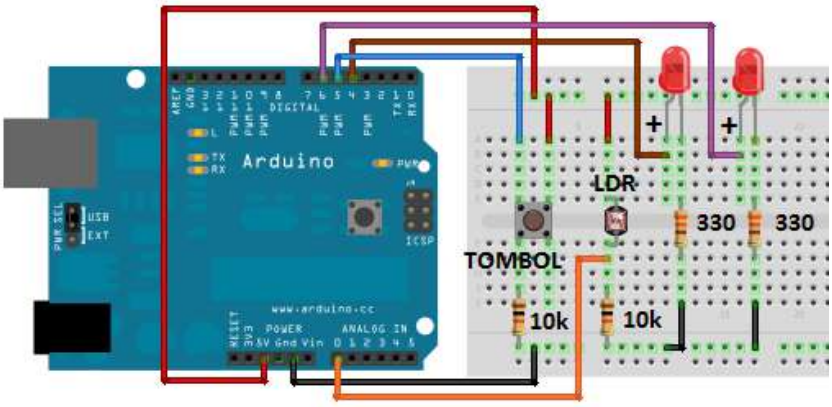
Catatan: hal yang menarik berikutnya adalah adanya icon **status Error**, yang di-OR-kan dengan tombol **Stop** ke terminal **Loop Condition**. Dengan cara ini, setiap kali terjadi **Error**, seperti contohnya pada gambar 10.34, secara otomatis program akan berhenti. Icon **status Error** tersebut dapat dibuat dengan meng-klik kanan kawat Error, dan memilih **Cluster, Class & Variant Pallete**, dan kemudian **Unbundle by Name**.

Untuk contoh-contoh aplikasi yang lain, diserahkan kepada pembaca untuk mencobanya.

10.5 Aplikasi Firmata pada Rangkaian Gabungan

Untuk memperjelas komunikasi serial menggunakan Firmata, maka berikut ini diberikan langkah-langkah pembuatan aplikasi Firmata untuk rangkaian gabungan pada contoh #6 di Bab 4.

11 Susun rangkaian gabungan seperti contoh #6 di Bab 4:



Gambar 10.37 Rangkaian gabungan

12 Hubungkan hardware Arduino ke komputer dengan kabel USB.

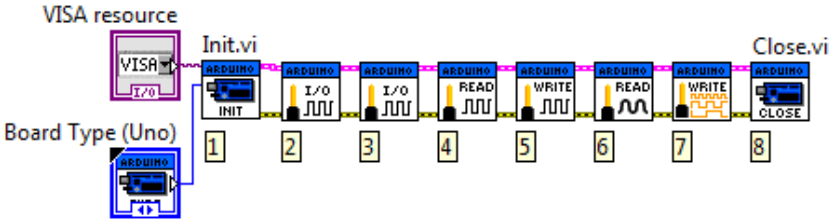
13 Apabila kode Firmata belum ditanamkan ke Arduino, maka tekan tombol **Upload** di software Arduino, untuk menanamkan kode Firmata (*LVIFA Firmware*) ke hardware Arduino.

14 Buka LabVIEW, pilih **New Blank VI**.

15 Pada jendela Block Diagram, tambahkan 8 buah icon fungsi LIFA Arduino, secara berturut-turut: 1. **Init**, 2. **Set Digital Pin Mode**, 3. **Set Digital Pin Mode**, 4. **Digital Read Pin**, 5. **Digital Write Pin**, 6. **Analog Read Pin**, 7. **PWM Write Pin**, 8. **Close**, dan hubungkan satu sama lain secara berurutan seperti gambar berikut ini.

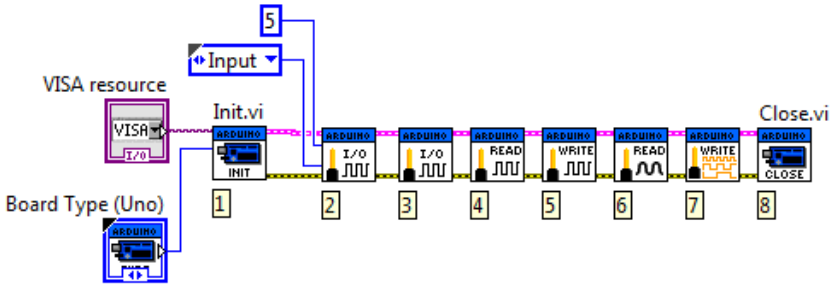
Gambar 10.38 Icon Fungsi-fungsi LIFA Arduino

16 Tambahkan **Create Control** pada kaki **VISA Resource** dan kaki **Board Type** pada fungsi **LIFA Init**.



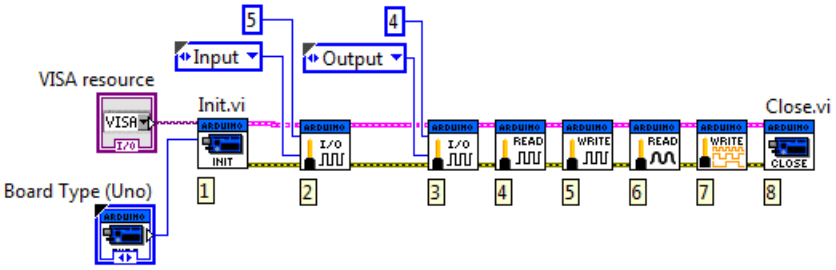
Gambar 10.39 Create Control pada VISA Resource dan Board Type

17 Tambahkan **Create Constant** pada kedua kaki input fungsi **LIVA Set Digital Pin Mode** (no. 2). Fungsi ini sama seperti instruksi **pinMode** pada software Arduino. Karena kaki D5 Arduino dibuat sebagai input untuk membaca tombol, maka isi kotak **Constant** dengan angka 5 pada kaki **IO pin**, dan pilih **Input** untuk kaki **mode pin**.



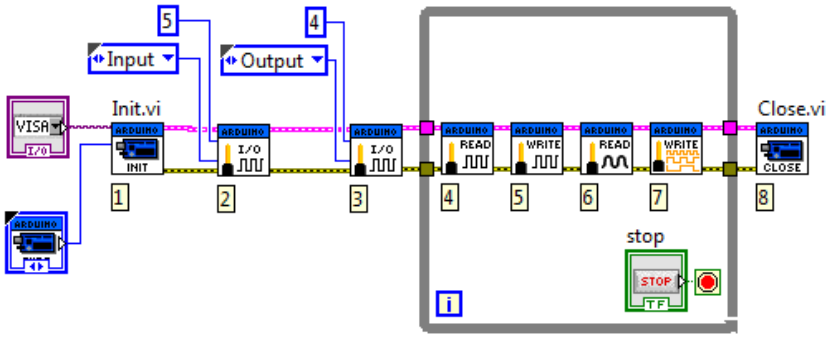
Gambar 10.40 Create Constant pada kedua kaki input fungsi Set Digital Pin Mode, yang pertama untuk I/O pin, yang kedua untuk mode pin.

18 Ulangi untuk fungsi **Set Digital Pin Mode** berikutnya (no. 3). Karena kaki D4 Arduino dibuat sebagai output untuk menghidupkan LED, maka tambahkan **Create Constant** pada kedua kaki, dan isi kotak Constant pertama dengan angka 4 untuk kaki **IO pin**, dan kotak Constant kedua dengan tipe **Output** untuk kaki **mode pin**.



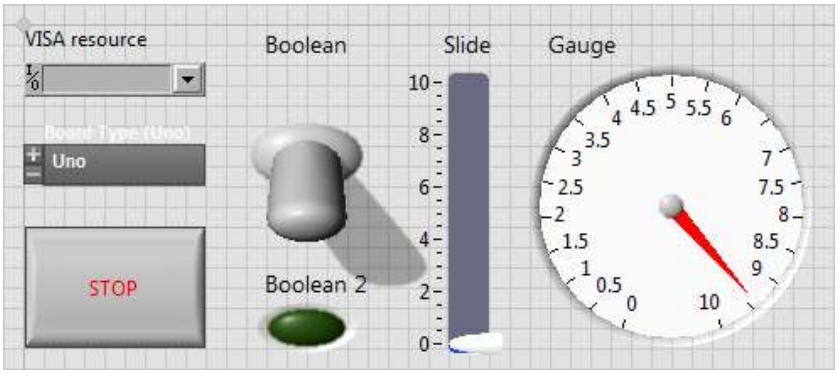
Gambar 10.41 Create Constant pada kedua kaki input fungsi Set Digital Pin Mode, yang pertama untuk I/O pin, yang kedua untuk mode pin.

19 Untuk fungsi-fungsi berikutnya (no. 4 – no. 7), tambahkan **While Loop**, karena fungsi-fungsi tersebut harus bekerja terus-menerus hingga komunikasi dihentikan.



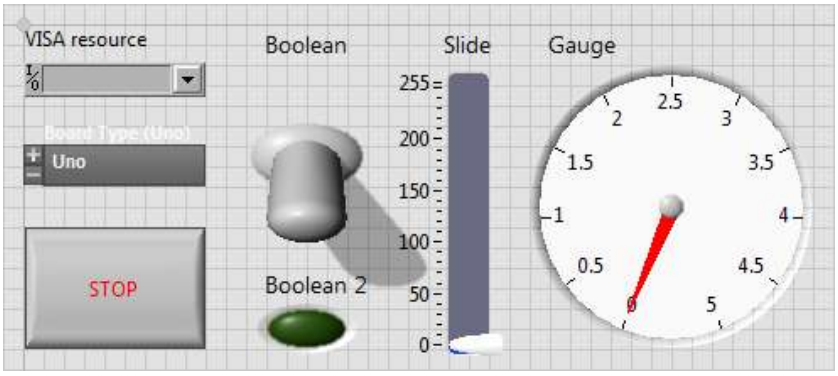
Gambar 10.42 Tambahkan While Loop pada fungsi no. 4 – no. 7

20 Buka jendela Front Panel, dan tambahkan 4 buah objek, yaitu **Toggle Switch**, **Round Led**, **Vertical Pointer Slide** dan **Gauge**.



Gambar 10.43 Tambahkan 4 buah objek, yaitu Toggle Switch, Round Led, Vertical Pointer Slide dan Gauge.

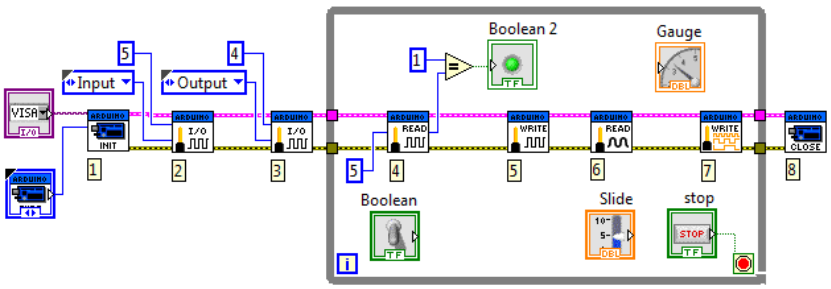
21 Ganti nilai jangkauan skala pada **Vertical Pointer Slide**, dari 0-10 menjadi 0-255. Kemudian ganti nilai jangkauan skala pada **Gauge**, dari 0-10 menjadi 0-5. Mengapa 0-5? Karena keluaran dari fungsi **LIFA Analog Read Pin** adalah nilai tegangan antara 0-5V.



Gambar 10.44 Mengganti nilai skala Vertical Pointer Slide dari 0-10 menjadi 0-255 dan Gauge dari 0-10 menjadi 0-5

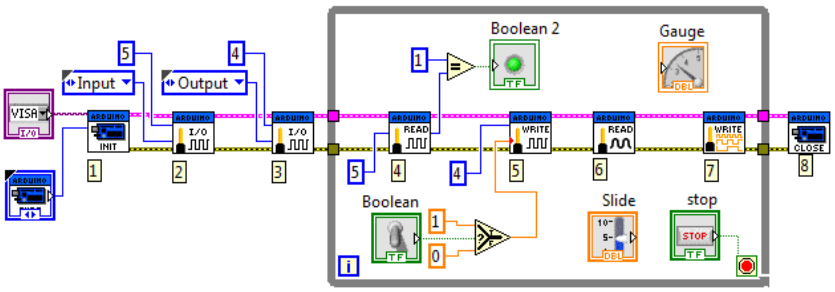
22 Buka jendela Block Diagram, pada kaki input **I/O pin** fungsi **Digital Read Pin** (fungsi no. 4), tambahkan angka 5. Mengapa 5? Karena kaki Arduino yang digunakan untuk membaca tombol adalah D5.

23 Kemudian tambahkan fungsi **Equal** untuk membandingkan nilai output **value** dari fungsi **Digital Read Pin** dengan nilai 1. Apabila nilainya sama dengan 1, maka nyalakan **Round Led**.



Gambar 10.45 Fungsi Equal untuk membandingkan output Digital Read Pin dengan nilai 1, bila sama maka nyalakan Round Led

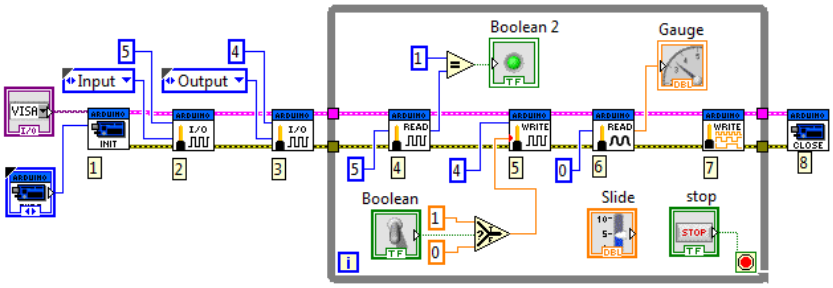
24 Pada kaki input **I/O pin** fungsi **Digital Write Pin** (fungsi no. 5), tambahkan angka 4. Mengapa 4? Karena kaki Arduino yang digunakan untuk menyalakan LED adalah D4. Kemudian pada kaki input **value**, tambahkan fungsi **Select** untuk memberikan angka 1 ke kaki input tersebut, ketika **Toggle Switch** digeser ke atas (True), dan angka 0 ketika **Toggle Switch** digeser ke bawah (False).



Gambar 10.46 Tambahkan fungsi Select untuk memberi 1 ketika Toggle Switch dituas ke atas, dan 0 ketika Toggle Switch dituas ke bawah

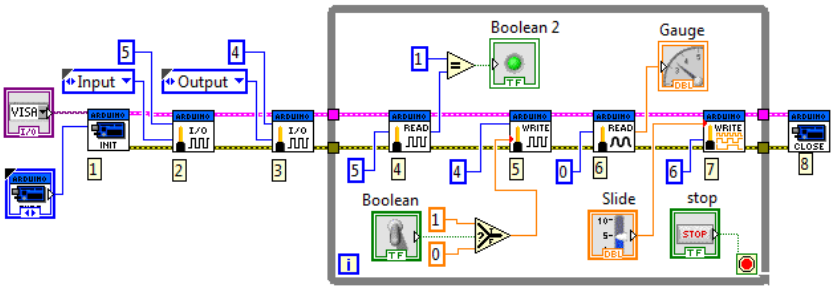
25 Pada kaki **analog input pin** fungsi **Analog Read Pin** (fungsi no. 6), beri angka 0. Mengapa 0? Karena kaki Arduino yang digunakan untuk

membaca sinyal analog LDR adalah kaki A0. Kemudian hubungkan kaki output **voltage** fungsi tersebut dengan **Gauge**.



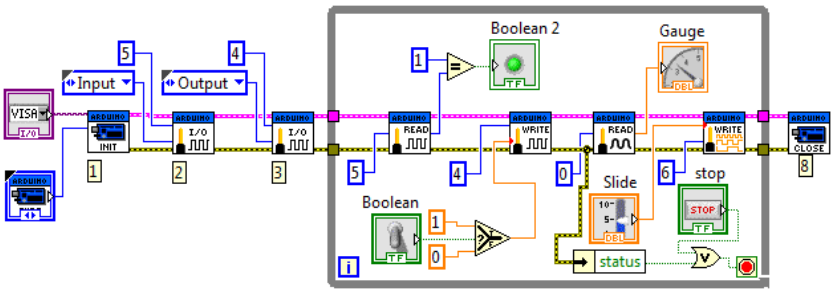
Gambar 10.47 Beri kaki input fungsi Analog Read Pin angka 0 (A0) dan hubungkan kaki output fungsi tersebut dengan Gauge

26 Hubungkan icon **Slide** dengan kaki input **duty cycle** fungsi **PWM Write Pin** (fungsi no. 7), dan beri angka 6 pada kaki input **PWM pin**. Mengapa 6? Karena kaki PWM Arduino yang digunakan adalah D6.



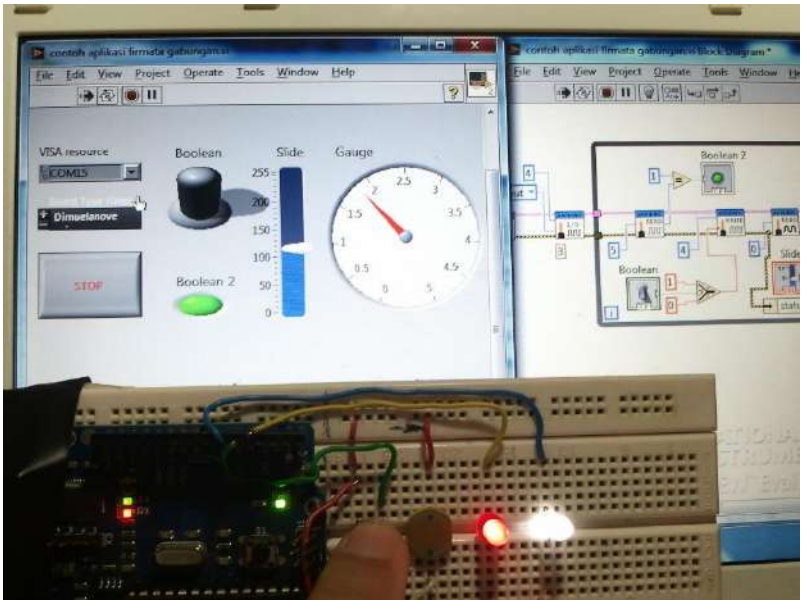
Gambar 10.48 Hubungkan icon Slide dengan kaki duty cycle fungsi PWM Write Pin, dan beri angka 6 pada kaki input PWM pin

27 Terakhir, untuk menghentikan program ketika terjadi Error, tambahkan **status Error**, yang di-OR-kan dengan tombol **Stop** ke terminal **Loop Condition**. Icon **status Error** tersebut dapat dibuat dengan meng-klik kanan kawat Error, dan memilih **Cluster, Class & Variant Pallete**, dan dilanjutkan dengan **Unbundle by Name**.



Gambar 10.49 Tambahkan status Error yang di-OR-kan dengan tombol Stop ke Loop Condition, untuk menghentikan program ketika Error

28 Jalankan program dengan menekan tombol Run, dan amati perubahan pada rangkaian gabungan dan tampilan pada layar ketika masing-masing inputnya, digerakkan atau diubah nilainya.



Gambar 10.50 Foto hasil program

BAB 11
ULTRASONIK RADAR

11.1 Apa itu Ultrasonik Radar

Sebagai proyek akhir dari buku ini, akan dibuat 2 buah alat, yaitu Ultrasonik Radar (Bab 11), dan Infrared Radar (Bab 12). Ultrasonik Radar akan dibuat menggunakan fungsi-fungsi komunikasi serial LabVIEW atau **VISA**, sedangkan Infrared Radar akan dibuat menggunakan fungsi-fungsi komunikasi serial Firmata LabVIEW atau **LIFA**.

Yang dimaksud dengan Ultrasonik Radar di sini adalah sebuah pengukur jarak sederhana menggunakan sensor Jarak Ultrasonik, yang memetakan jarak objek-objek yang berada di dalam daerah jangkauannya. Umumnya sensor Ultrasonic memiliki jangkauan hingga lebih dari 3 m.

Aplikasi dari Ultrasonik Radar ini salah satunya adalah untuk navigasi **AGV** (*Autonomous Guided Vehicle*), yaitu suatu kendaraan tanpa awak, yang biasanya digunakan di industri, untuk memindahkan material produksi dari satu lokasi ke lokasi yang lain. Tentu saja untuk AGV yang sebenarnya, dibutuhkan sensor yang lebih baik dan jauh jangkauannya, seperti sensor jarak dengan Laser misalnya, yang bisa mendeteksi jarak hingga puluhan meter dengan tingkat ketelitian yang tinggi.

Namun sensor Ultrasonik dipilih di sini, karena harganya yang relatif murah dan banyak tersedia di pasaran (untuk informasi toko yang menyediakan, silahkan melihat pada Lampiran buku ini).

11.2 Spesifikasi Alat

Sesuai namanya, maka diinginkan alat Ultrasonik Radar ini memiliki spesifikasi atau kemampuan sebagai berikut:

1. Mampu mendeteksi jarak semua objek yang berada dalam radius daerah jangkauannya.
2. Mampu memetakan hasil pendeteksian tersebut dalam bentuk gambar atau grafik.
3. Mampu menyimpan hasil pemetaan tersebut dalam bentuk file.
4. Mampu memberikan indikator suara untuk jarak objek tertentu.

11.3 Cara Pembuatan

Untuk bisa menghasilkan kemampuan sesuai spesifikasi di atas, dibutuhkan 3 jenis komponen, yaitu Arduino (1 buah), motor mini servo (2 buah) dan sensor Ultrasonik (1 buah).

Kemudian agar bisa mendeteksi posisi objek-objek di sekitarnya tempelkan sensor Ultrasonik di atas baling-baling motor mini servo, agar sensor Ultrasonik tersebut dapat diputar oleh motor mini servo.

Motor mini servo yang digunakan adalah 2 buah, karena satu motor mini servo hanya bisa memutar 180 derajat, sehingga untuk bisa memutar 360 derajat, dibutuhkan 2 buah motor mini servo. Lebih jelasnya mengenai cara pemasangan sensor Ultrasonik pada kedua motor, dapat diihat pada Gambar 11.1.

Gambar 11.1 Pemasangan sensor Ultrasonik pada kedua motor

Untuk memudahkan pembaca, maka langkah pembuatan akan dilakukan secara bertahap, dengan perincian sebagai berikut:

1. Tahap pertama: melakukan pembacaan jarak objek menggunakan sensor Ultrasonik yang hasilnya ditampilkan dalam bentuk grafik di software LabVIEW.
2. Tahap kedua: mengendalikan kedua motor mini servo secara otomatis (berputar bolak-balik) dan secara manual dengan memutar objek Dial di software LabVIEW.
3. Tahap ketiga: melakukan pembacaan jarak objek di semua sudut putar sensor Ultrasonik dan memetakan hasilnya pada grafik dengan koordinat polar di software LabVIEW.
4. Tahap keempat: menambahkan fungsi untuk menyimpan hasil pembacaan jarak dan sudut dalam bentuk teks dan gambar.

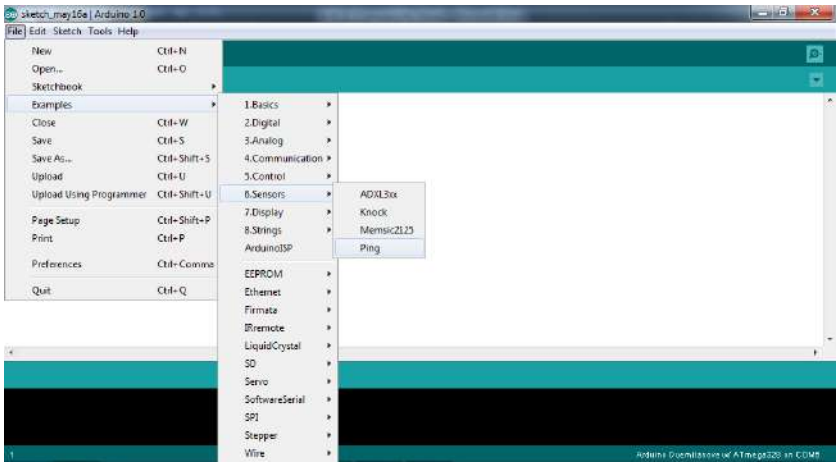
5. Tahap kelima: membangkitkan suara yang besarnya frekuensi suara tersebut ditentukan oleh jarak objek terhadap sensor, semakin dekat semakin tinggi frekuensinya.

Berikut satu-persatu uraian dari tahapan pembuatan tersebut.

11.4 Tahap Pertama

Berikut langkah-langkah pembuatan Tahap Pertama:

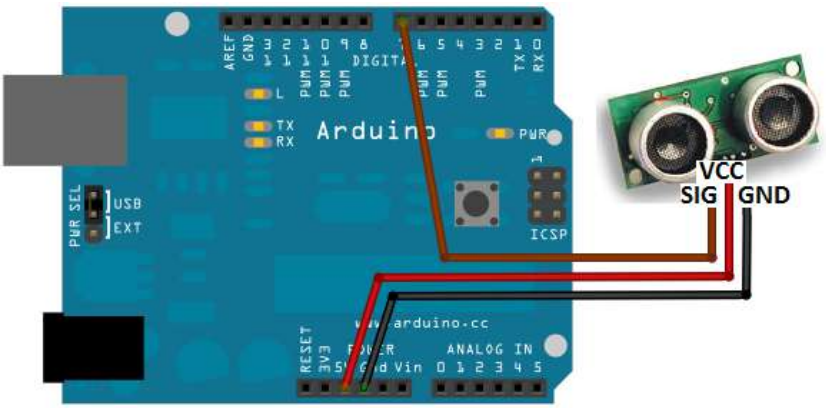
21. Hubungkan hardware Arduino dengan komputer melalui kabel USB.
22. Buka Software Arduino, pada menu **File**, pilih **Examples**, dilanjutkan **Sensors**, kemudian **Ping** (catatan: Ping adalah nama sebuah produk sensor jarak ultrasonik yang dibuat oleh Parallax, memiliki 3 kaki).



Gambar 11.2 Pilih Files >> Examples >> Sensors >> Ping

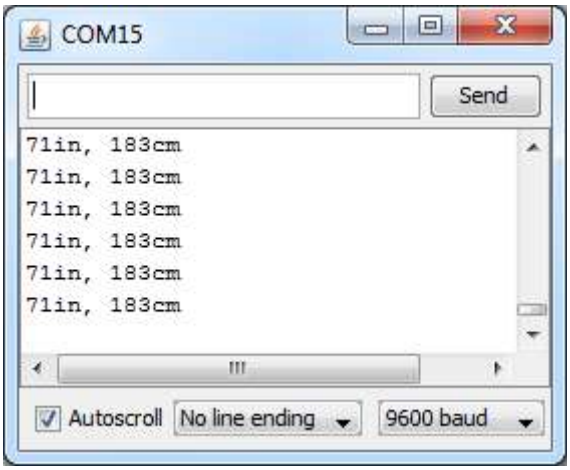
23. Setelah program Ping terbuka, tekan tombol **Verify** dan kemudian tekan tombol **Upload**. Namun sebelum tombol **Upload** ditekan, pastikan **Board** yang dipilih pada menu **Tools** sudah sesuai dengan yang digunakan.

24. Hubungkan kaki **SIG**, **VCC** dan **GND** sensor secara berturut-turut ke kaki **D7**, **5V** dan **GND** Arduino.



Gambar 11.3 Menghubungkan kaki SIG, VCC, GND sensor ke Arduino

25. Buka Serial Monitor dengan menekan tombol kaca pembesar. Maka seharusnya Serial Monitor menampilkan hasil pembacaan jarak dalam satuan inchi dan cm seperti berikut:



Gambar 11.4 Serial Monitor menampilkan hasil pembacaan jarak objek terhadap sensor dalam satuan inchi dan cm

26. Untuk memastikan komunikasi Arduino dengan LabVIEW berjalan dengan baik, maka modifikasi program **Ping** di atas sehingga data yang dikirimkan menjadi ASCII Byte seperti program berikut ini:

tahap_pertama

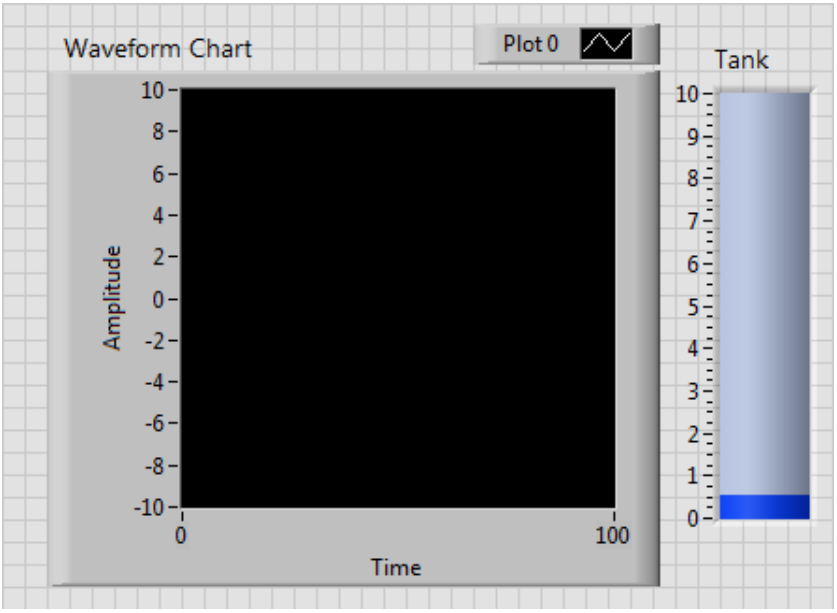
```
void setup()
{ Serial.begin(9600);}
void loop()
{ pinMode(7, OUTPUT);
  digitalWrite(7, LOW);
  delayMicroseconds(2);
  digitalWrite(7, HIGH);
  delayMicroseconds(5);
  digitalWrite(7, LOW);
  pinMode(7, INPUT);
  long duration = pulseIn(7, HIGH);
  long cm = duration/58;
  Serial.write(highByte(cm));
  Serial.write(lowByte(cm));
  delay(100);  }
```

Gambar 11.5 Modifikasi program Ping sehingga mengirimkan data Byte

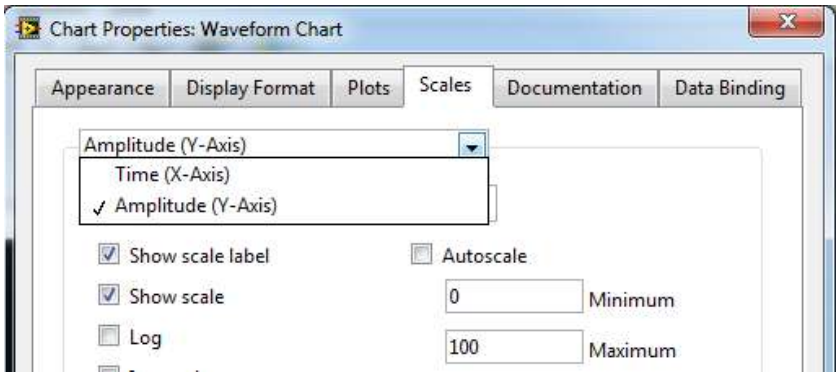
27. Perhatikan program modifikasi Gambar 11.5 di atas, di mana karena nilai variabel **cm** bisa lebih dari 255, maka variabel **cm** tersebut dibagi menjadi 2 buah data 8 bit dengan fungsi highByte() dan lowByte(). **Upload** program tersebut ke Arduino.

28. Buka LabVIEW, dan ambil objek **Waveform Chart** (kategori **Graph**) dan objek **Tank** (kategori **Num Inds**) dari palet Controls dan tempatkan pada **Front Panel** seperti Gambar 11.6.

29. Klik kanan **Waveform Chart**, kemudian pilih **Properties**, pilih **Scales**, pilih **Y axis** dan kemudian hilangkan tanda cek pada **Autoscale**. Berikutnya beri nilai **0** untuk Minimum dan **100** untuk Maximum.



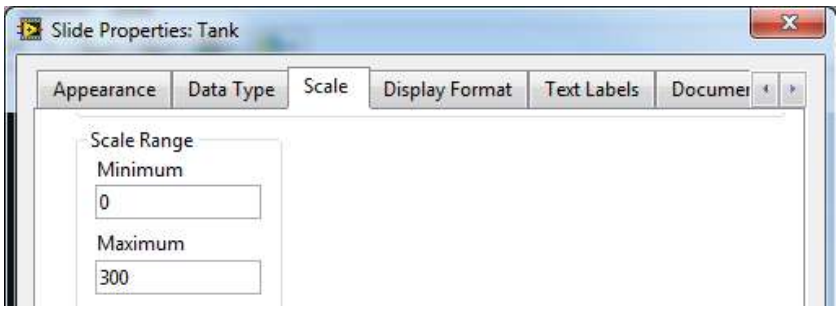
Gambar 11.6 Tempatkan objek Waveform Chart dan Tank di Front Panel



Gambar 11.7 Buat skala sumbu Y Waveform Chart menjadi 0–100 (tetap)

30. Klik kanan objek **Tank**, pilih **Properties**, pilih **Scale**, dan ganti nilai Maximum pada **Scale Range** dari 10 menjadi 100.

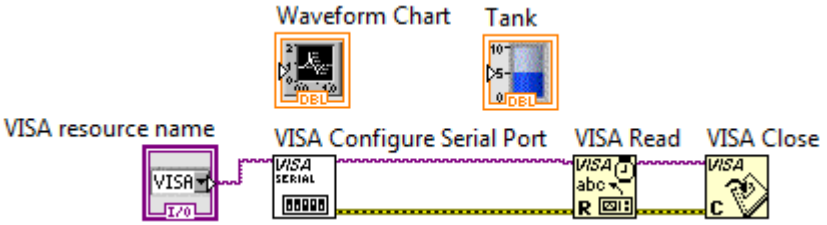
31. Buka jendela Block Diagram, dan tambahkan fungsi **VISA Configure Serial Port**, **VISA Read** dan **VISA Close**.



Gambar 11.8 Buat skala Tank dari 0 – 10 menjadi 0 – 100

Gambar 11.9 Tambahkan fungsi VISA Configure, Read dan Close

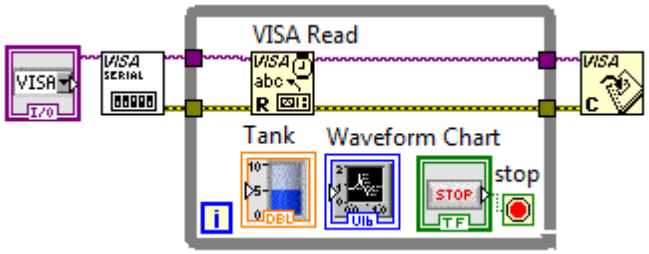
32. **Create Control** pada kaki input ***VISA resource name*** fungsi **VISA Configure Serial Port**. Kemudian hubungkan **VISA Configure Serial Port**, **VISA Read** dan **VISA Close** seperti berikut.



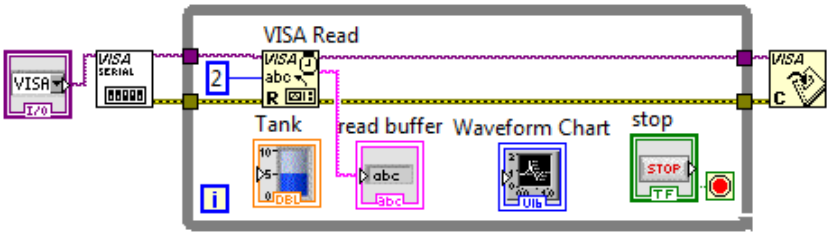
Gambar 11.10 Tambahkan fungsi VISA Configure, Read dan Close

33. Tambahkan **While Loop** pada fungsi **VISA Read**, dan masukkan icon **Waveform Chart** dan icon **Tank** seperti terlihat pada Gambar 11.11.

34. **Create Constant** pada kaki input ***byte Count*** dan isi dengan nilai 2. Mengapa 2? karena data yang dikirimkan dari Arduino sebanyak 2 byte. Kemudian **Create Indicator** pada kaki output ***read buffer*** untuk melihat data ASCII yang diterima tersebut di jendela Front Panel.

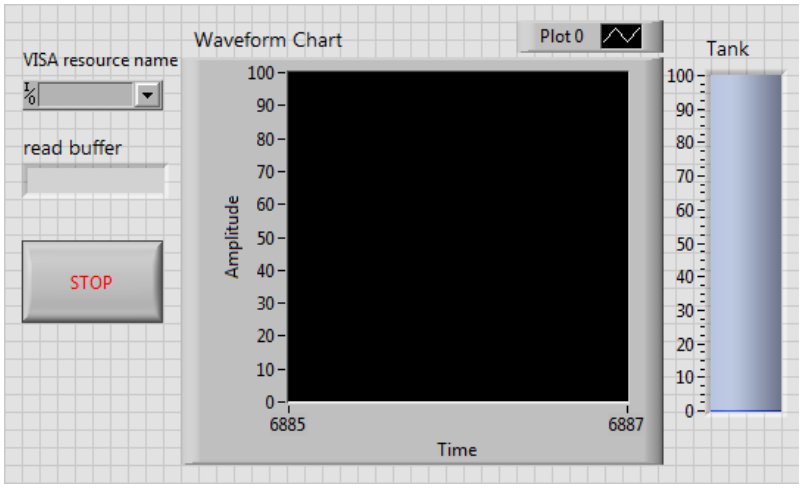


Gambar 11.11 Menambahkan While Loop pada fungsi VISA Read dan memasukkan icon Waveforrm Chart dan Tank ke dalamnya



Gambar 11.12 Create Constant pada kaki input dan beri nilai 2, kemudian Create Indicator pada kaki output read buffer

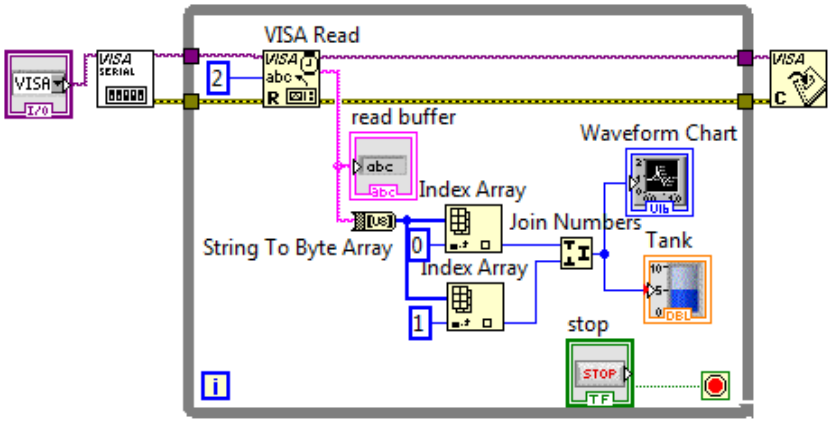
35. Susun objek-objek di jendela Front Panel seperti gambar berikut.



Gambar 11.13 Menata objek-objek yang akan ditampilkan di Front panel

36. Gunakan fungsi **String to Byte Array** diikuti dengan fungsi **Index Array** dan dilanjutkan dengan fungsi **Join Numbers** untuk mengubah data 2 buah karakter ASCII (dari output fungsi **VISA Read**) menjadi data sebuah angka desimal.

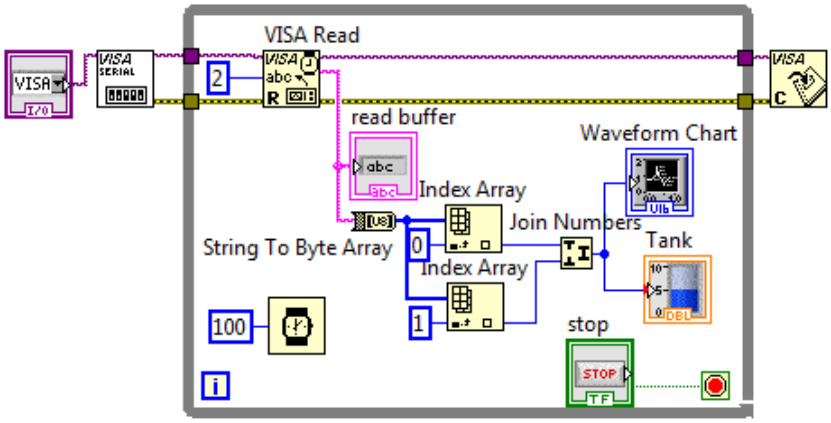
37. Data angka desimal tersebut selanjutnya ditampilkan pada grafik **Waveform Chart** dan indicator **Tank**. Untuk itu hubungkan output **Join Numbers** dengan icon **Waveform Chart** dan icon **Tank**.



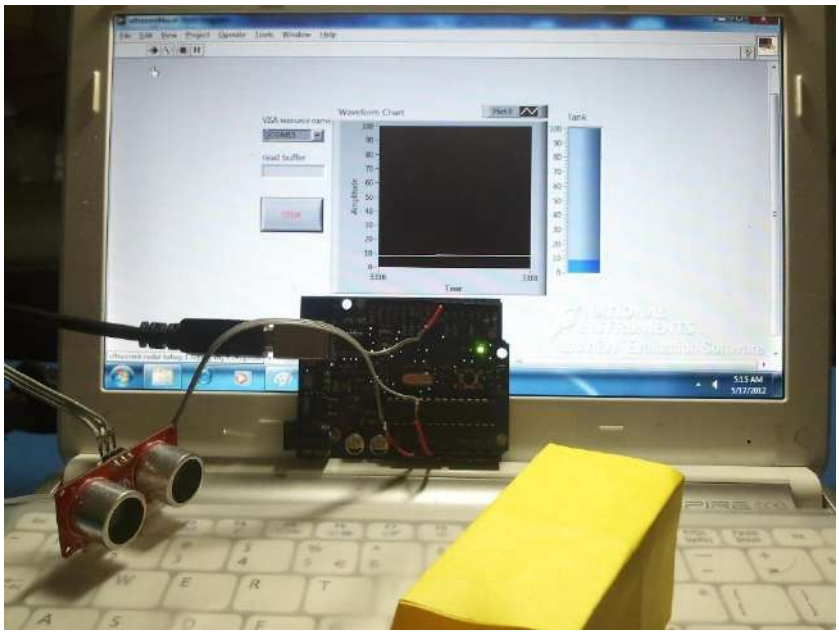
Gambar 11.14 Mengubah 2 byte ASCII menjadi sebuah angka desimal

38. Tambahkan fungsi **Wait(ms)** dengan nilai sebesar 100 ms, untuk memastikan bahwa data yang dikirimkan dari Arduino diterima secara utuh, seperti ditunjukkan pada Gambar 11.15.

39. Pilih saluran COM yang digunakan pada kotak ***VISA resource name*** di Front Panel. Kemudian jalankan program dengan menekan tombol **Run**. Perhatikan bahwa **Waveform Chart** dan **Tank** menampilkan hasil pembacaan jarak oleh sensor Ultrasonik, seperti ditunjukkan pada foto di Gambar 11.16. Sebagai catatan, nilai jarak yang ditampilkan tersebut dalam satuan cm.



Gambar 11.15 Menambahkan tunda waktu sebesar 100ms untuk memastikan komunikasi berjalan dengan baik



Gambar 11.16 Waveform Chart dan Tank menampilkan jarak objek terhadap sensor Ultrasonik sebesar kurang lebih 9 cm

40. Tekan tombol Stop untuk menghentikan program Tahap Pertama.

11.5 Tahap Kedua

Berikut langkah-langkah pembuatan Tahap Kedua:

1. Buka software Arduino dan ketikkan program berikut ini:

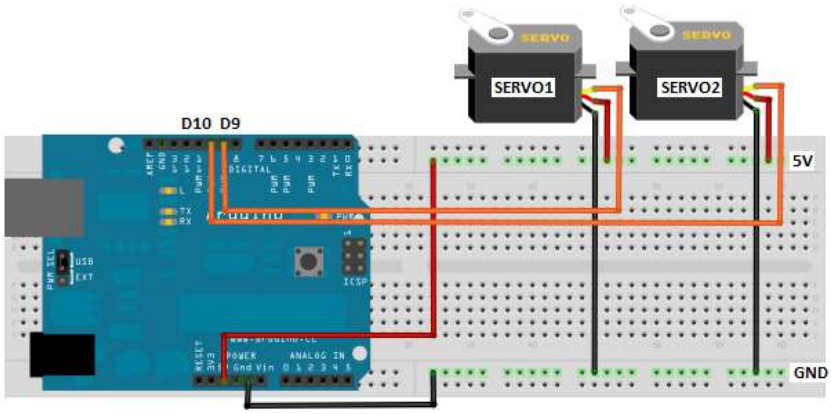
tahap_kedua §

```
#include <Servo.h>
Servo myservo1;
Servo myservo2;
byte data1;
byte data2;
void setup()
{ Serial.begin(9600);
  myservo1.attach(9);
  myservo2.attach(10);}
void loop()
{ if(Serial.available())
  {data1=Serial.read();
   data2=Serial.read();}
  myservo1.write(data1);
  myservo2.write(data2);
  delay(100);}
```

Gambar 11.17 Program Arduino pada Tahap Kedua

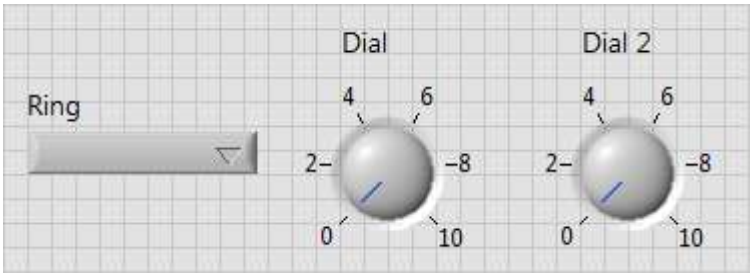
2. Program di atas pada dasarnya menunggu 2 byte data dari komputer. Data byte pertama untuk menggerakkan motor servo1, dan data byte kedua untuk menggerakkan motor servo2. **Compile** dan **Upload** program tersebut ke Arduino.

3. Hubungkan kabel oranye masing-masing motor servo tersebut ke kaki D9 dan D10 Arduino. Sedangkan kabel merah dan kabel hitam masing-masing servo ke kaki 5V dan GND Arduino. Lebih jelasnya perhatikan gambar pengawatan rangkaian berikut ini.



Gambar 11.18 Dua buah motor mini servo dihubungkan ke Arduino

4. Buka jendela Front Panel LabVIEW dan tempatkan objek **Menu Ring** dari kategori **Ring&Enum** di palet Controls. Tempatkan pula 2 buah objek **Dial** dari kategori **Numeric Controls**.

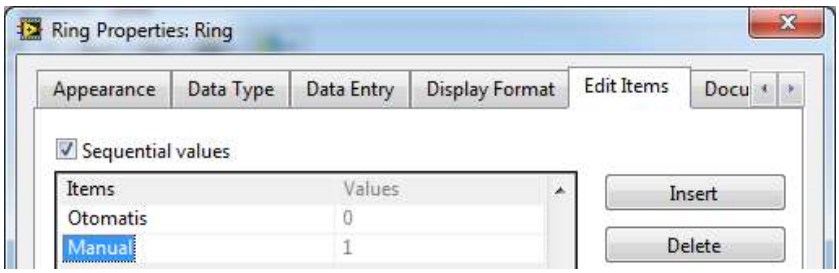


Gambar 11.19 Tempatkan objek Menu Ring dan 2 buah Dial

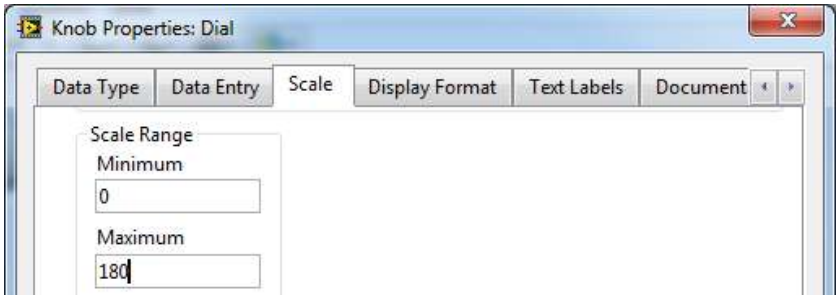
5. Klik kanan pada objek **Menu Ring**, dan pilih **Edit Items**. Pada daftar **Items** yang muncul dan masih kosong di kotak **Edit Items**, ketik **Otomatis**, kemudian ketik **Manual** di bawahnya.

6. Klik kanan pada objek **Dial**, dan pilih **Properties**, kemudian pilih **Scale**, dan ganti nilai **Maximum** dari 10 menjadi 180. Mengapa 180? Karena motor servo hanya dapat berputar antara 0 – 180 derajat.

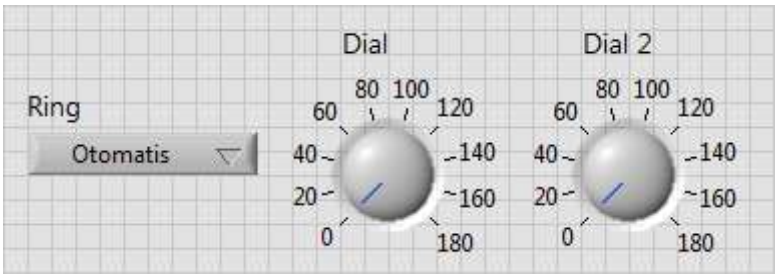
7. Ulangi langkah no. 6 untuk objek **Dial** kedua.



Gambar 11.20 Ketik pada baris di bawah Items: Otomatis dan Manual



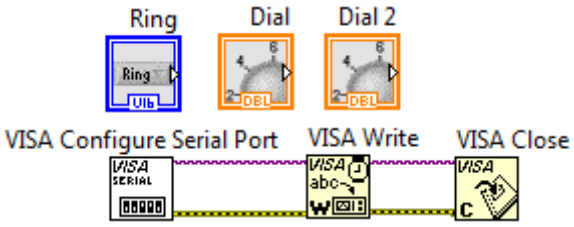
Gambar 11.21 Pada objek Dial, ubah nilai Maximum dari 10 menjadi 180



Gambar 11.22 Penambahan otomatis/manual dan pengubahan skala

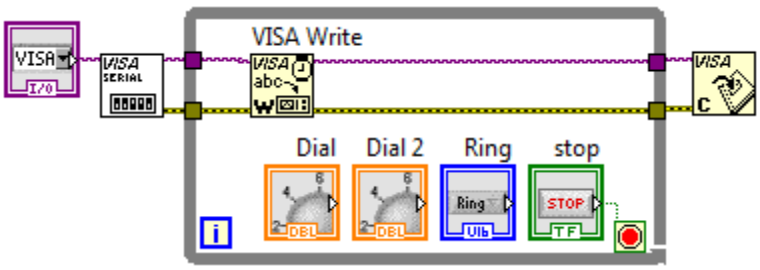
8. Hilangkan label pada ketiga objek di atas dengan meng-klik kanan, pilih **Visible Items**, kemudian hilangkan tanda centang pada **Label**.

9. Buka jendela Block Diagram, dan tambahkan 3 buah icon fungsi serial, yaitu **VISA Configure Serial Port**, **VISA Write** dan **VISA Close**. Kemudian hubungkan ketiganya seperti Gambar 11.23 berikut.



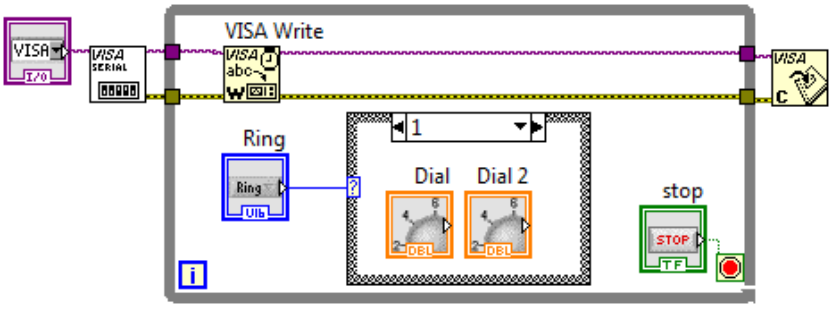
Gambar 11.23 Tambahkan 3 buah icon fungsi Serial

10. **Create Control** pada kaki input ***Visa resource name*** di fungsi **VISA Configure Serial Port**. Kemudian tambahkan **While Loop** pada **VISA Write**, serta masukkan icon **Ring**, **Dial** dan **Dial2** ke dalamnya.



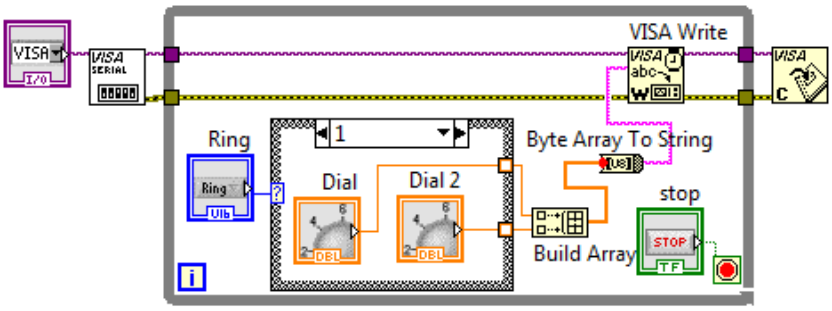
Gambar 11.24 VISA Write, Dial, Dial2 dan Ring di dalam While Loop

11. Tambahkan **Case Structure**, masukkan **Dial** dan **Dial2** ke dalamnya, dan hubungkan **Ring** dengan terminal **Selector Case Structure**.



Gambar 11.25 Tambahkan Case Structure dan masukkan Dial dan Dial2 ke dalamnya, dan Ring sebagai input terminal selectornya

12. Tambahkan fungsi **Build Array** dan **Byte Array to String** untuk membuat tipe data Numerik dari objek **Dial** dan **Dial2** berubah menjadi String ASCII yang kemudian dikirimkan oleh **VISA Write**.



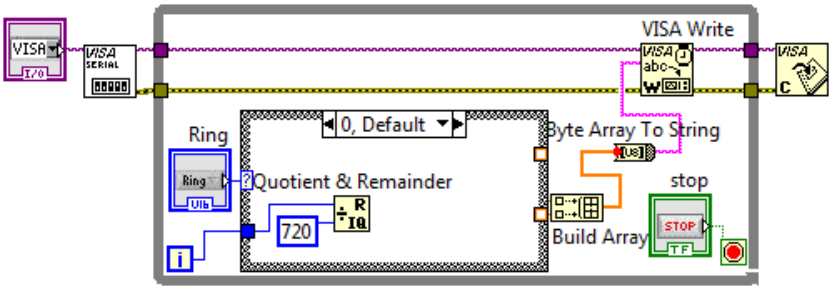
Gambar 11.26 Fungsi Build Array dan Byte Array to String untuk mengubah data Numeric Dial dan Dial2 menjadi ASCII String

13. Klik pada label di bagian atas **Case Structure** untuk membuka blok Case **0** atau blok Case untuk pilihan menu Ring **Otomatis**. Diinginkan pada pilihan **Otomatis** ini, terjadi 4 buah proses berikut ini:

    - Proses 1: motor mini servo1 berputar dari 0 ke 180 derajat.
    - Proses 2: motor mini servo2 berputar dari 0 ke 180 derajat.
    - Proses 3: motor mini servo2 berputar dari 180 derajat ke 0.
    - Proses 4: motor mini servo1 berputar dari 180 derajat ke 0.

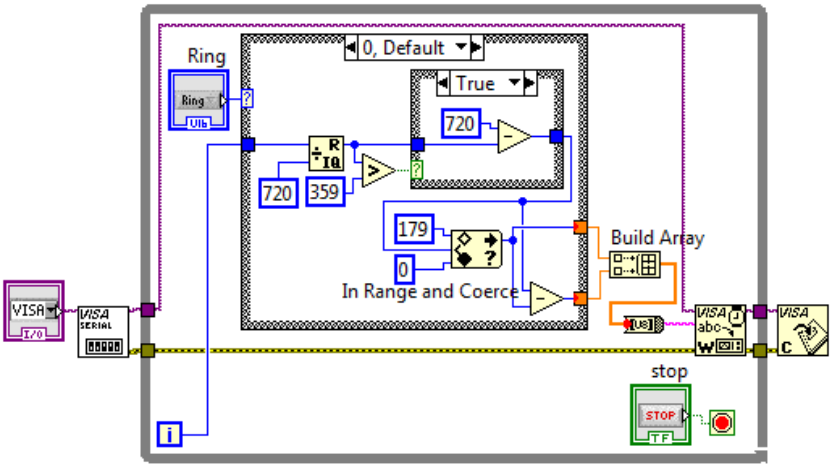
    Keempat proses tersebut seharusnya bekerja secara berurutan dan berulang kali, yang nantinya akan digunakan untuk memutar sensor Ultrasonic sebesar 360 derajat bolak-balik.

    Untuk itu tambahkan fungsi **Quotient & Remainder**, dengan kaki input **x** diberikan nilai terminal **Count** (**i**) **While Loop**, dan kaki input y diberikan nilai perulangan siklus, yaitu sebesar 720. Mengapa 720? Karena 360 derajat bolak-balik itu sama dengan 360 x 2 = 720.



Gambar 11.27 Menambahkan fungsi Quotient & Remainder untuk membuat siklus setiap kelipatan 720 pada nilai terminal Count (i)

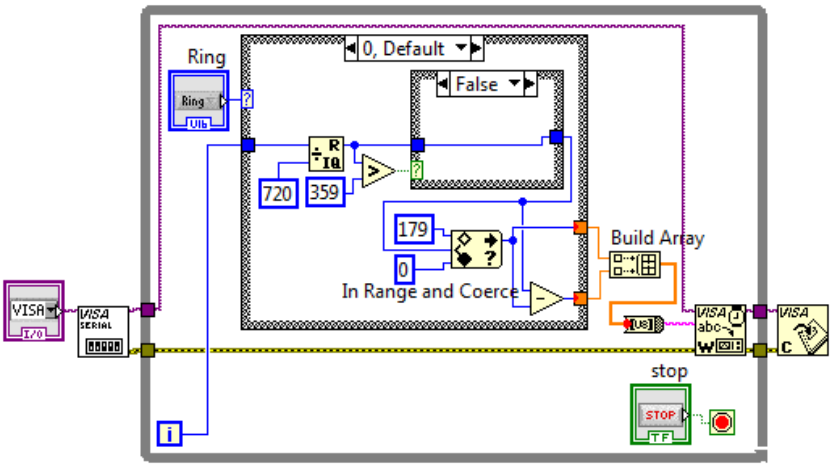
14. Setiap 360 derajat, sensor Ultrasonik harus diputar berlawanan arah. Untuk itu tambahkan fungsi yang membandingkan apakah inputnya lebih besar dari 359. Jika ya, maka motor harus diputar berlawanan arah, jika tidak, maka biarkan arah putaran motor tetap.



Gambar 11.28 Jika nilai lebih besar dari 359, maka putar berlawanan arah (=720-input), jika sebaliknya, maka putar searah (=input).

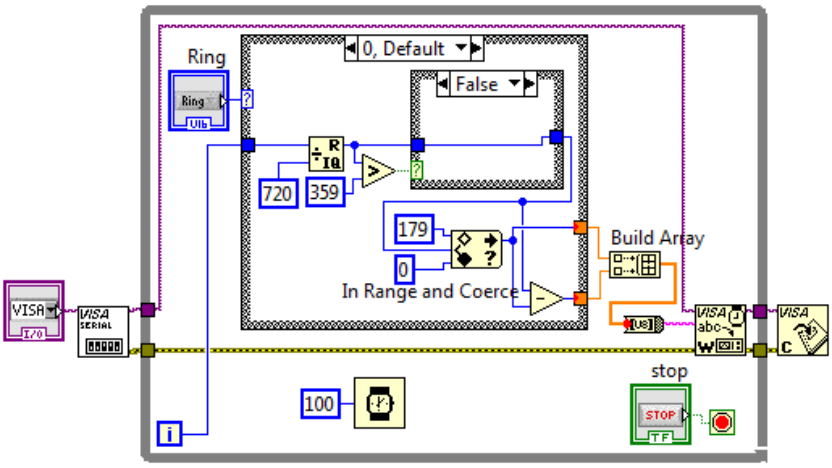
15. Karena motor mini servo hanya bisa berputar 180 derajat, maka lakukan pembagian sedemikian rupa sehingga untuk nilai 0 – 179 diberikan ke motor mini servo1, sedangkan untuk nilai 180 – 359 diberikan ke motor mini servo2, yang tentunya nilai tersebut diubah

dulu menjadi 0 – 179. Gunakan fungsi **In Range and Coerce** untuk mengekang nilai output tidak kurang dan tidak lebih dari 0 – 179.



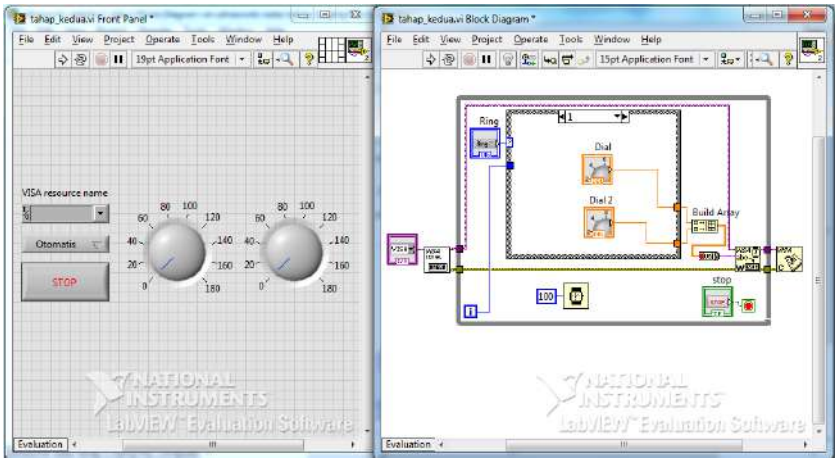
Gambar 11.29 Tambahkan fungsi In Range & Coerce untuk mengekang nilai output tidak kurang dan tidak lebih dari 0-179

16. Agar komunikasi berjalan lancar, tambahkan **Wait(ms)** = 100 ms.



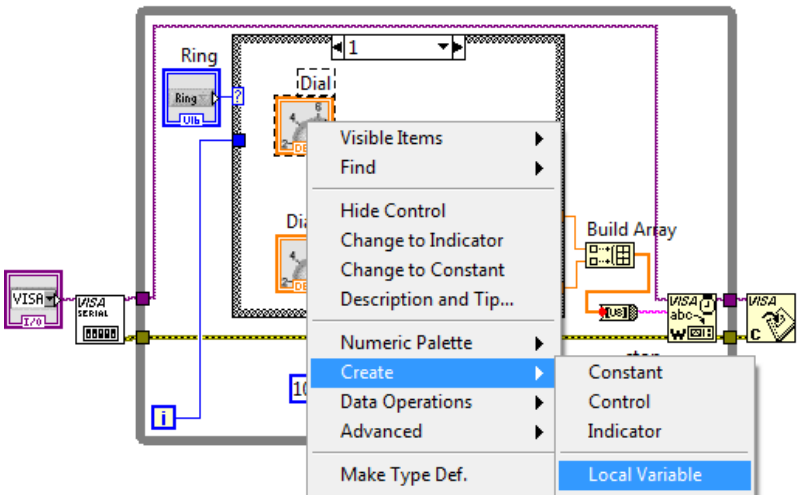
Gambar 11.30 Menambahkan Wait(ms) = 100 ms

17. Sampai di sini, program bekerja sebagai berikut: ketika tombol Run ditekan, dan pilihan ke mode **Otomatis**, maka motor mini servo1 akan berputar dulu sampai maksimum, diikuti motor mini servo2. Kemudian setelah mencapai maksimum, motor mini servo2 akan berputar berlawanan arah hingga maksimum, kemudian diikuti motor mini servo1, dan proses kembali berulang. Kemudian ketika pilihan diubah ke mode **Manual**, maka kedua motor mini servo dapat diatur secara manual menggunakan objek **Dial**.



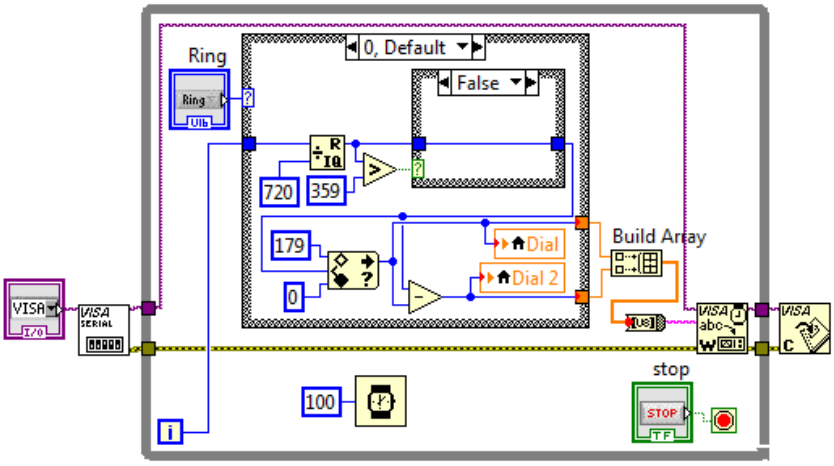
Gambar 11.31 Program pengendalian 2 motor mini servo

18. Selanjutnya, diinginkan agar ketika pilihan ke mode **Otomatis**, kedua objek **Dial** tidak hanya diam, tetapi bergerak juga secara otomatis mengikuti posisi baling-baling masing-masing motor mini servo. Dalam kondisi ini, objek **Dial** tidak berlaku sebagai **Control**, tetapi sebagai **Indicator**. Untuk bisa membuat objek **Dial** menjadi **Control** saat mode **Manual**, dan menjadi **Indicator** saat mode **Otomatis**, maka perlu menambahkan **Local Variable** dari **Dial**, yang dimunculkan dengan cara meng-klik kanan **Dial**, dan kemudian memilih **Create Local Variable**.



Gambar 11.32 Membuat Local Variable untuk objek Dial

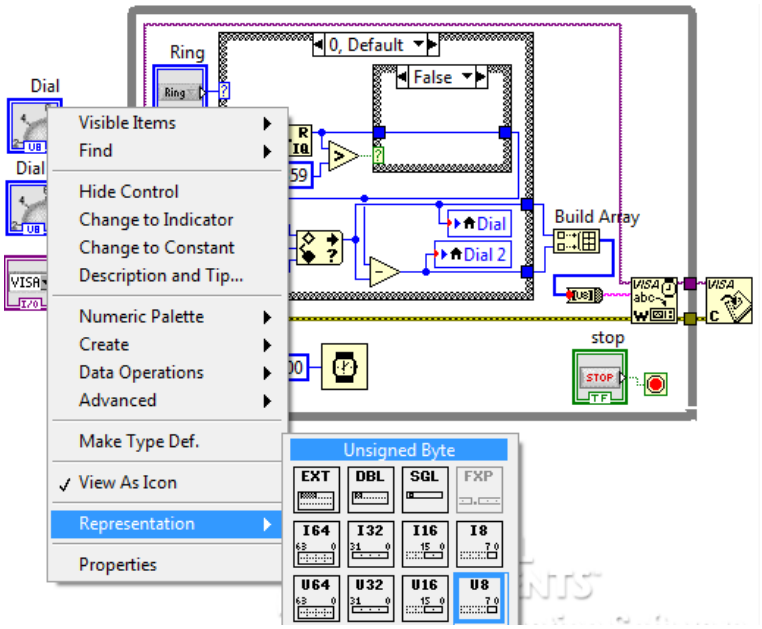
19. Tempatkan **Local Variable** untuk kedua **Dial** tersebut pada mode **Otomatis** atau **blok Case 0**, dan hubungkan masing-masing **Local Variable** tersebut dengan nilai yang akan dikirimkan ke masing-masing motor mini servo.



Gambar 11.33 Hubungkan kedua Local Variable Indicator Dial dengan kedua nilai yang akan dikirimkan ke masing-masing motor mini servo

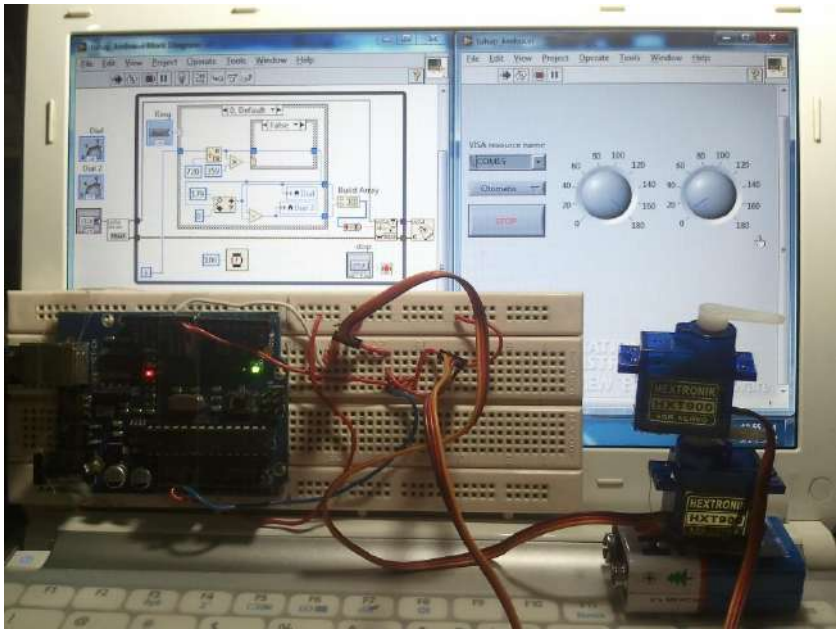
Catatan: apabila diinginkan, **Local Variable** dapat diubah dari tipe **Indicator** menjadi **Control**, atau sebaliknya, dengan cara meng-klik kanan **Local Variable** dan memilih **Change To Read** atau **Change To Write** untuk membuatnya menjadi **Control** atau **Indicator**.

20. Terakhir, karena data Numerik yang dikirimkan ke Arduino adalah data Byte (**U8**), dan bukan data angka pecahan (**DBL**), maka ubah tipe data **Dial** dengan cara meng-klik kanan **Dial**, dan pilih **Representation**, dan kemudian pilih **U8** (***Unsigned Byte***).



Gambar 11.34 Mengubah tipe data kedua Dial dari DBL menjadi U8

21. Jalankan program dengan menekan tombol Run. Perhatikan bahwa ketika mode **Otomatis** dipilih, maka secara otomatis kedua **Dial** akan mengikuti posisi baling-baling motor mini servonya. Sedangkan ketika mode **Manual** dipilih, posisi baling-baling kedua motor mini servo akan bergerak mengikuti posisi **Dial** yang diputar.



Gambar 11.35 Foto pengendalian 2 motor mini servo

11.6 Tahap Ketiga

Berikut langkah-langkah pembuatan Tahap Ketiga:

1. Program Arduino pada Tahap Ketiga ini merupakan penggabungan dari program Tahap Pertama dan Kedua, yang akan membuat sensor Ultrasonik membaca jarak sementara diputar oleh kedua motor mini servo. Ketik program pada Gambar 11.36, dan **Upload** ke Arduino.

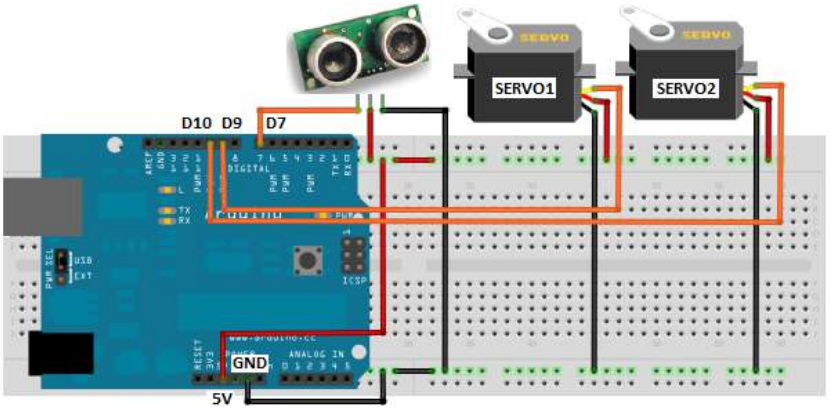
2. Susun sensor Ultrasonik dan 2 buah motor mini servo seperti pada Gambar 11.1. Kemudian buat rangkaian pada Breadboard seperti Gambar 11.37, yang menghubungkan kaki D7 Arduino dengan sensor Ultrasonik, kaki D9 dan D10 Arduino dengan kabel oranye pada kedua motor mini servo. Kemudian hubungkan semua kabel merah ke kaki 5V Arduino, dan kabel hitam ke kaki GND Arduino.

tahap_kedua §

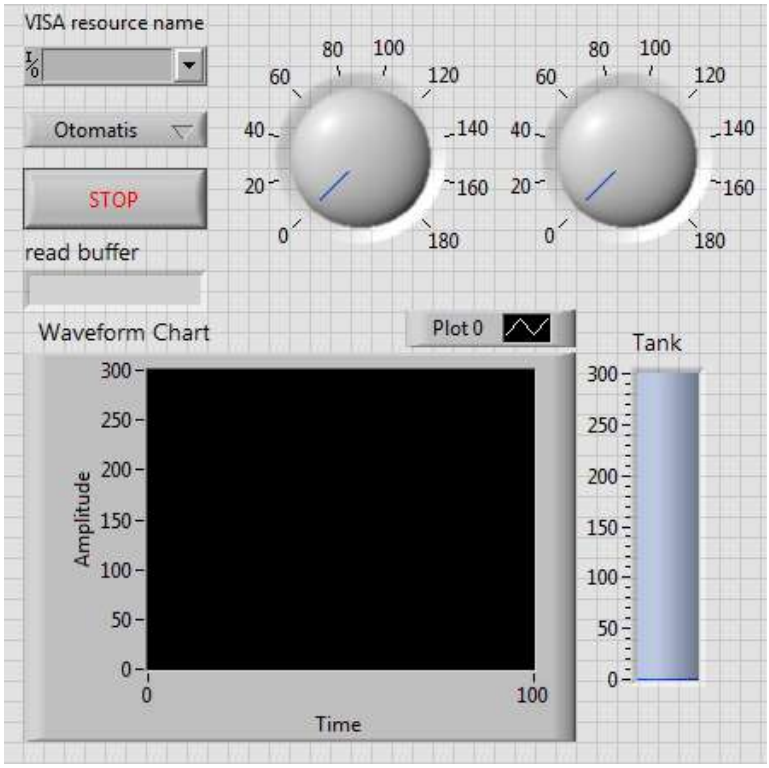
```
#include <Servo.h>
Servo myservo1;
Servo myservo2;
byte data1;
byte data2;
void setup()
{ myservo1.attach(9);
  myservo2.attach(10);
  Serial.begin(9600);}
void loop()
{ if(Serial.available())
  {data1=Serial.read();
   data2=Serial.read();}
  myservo1.write(data1);
  myservo2.write(data2);
  pinMode(7, OUTPUT);
  digitalWrite(7, LOW);
  delayMicroseconds(2);
  digitalWrite(7, HIGH);
  delayMicroseconds(5);
  digitalWrite(7, LOW);
  pinMode(7, INPUT);
  long duration = pulseIn(7, HIGH);
  long cm = duration/58;
  Serial.write(highByte(cm));
  Serial.write(lowByte(cm));
  delay(100);}
```

Gambar 11.36 Program Arduino pada Tahap Ketiga

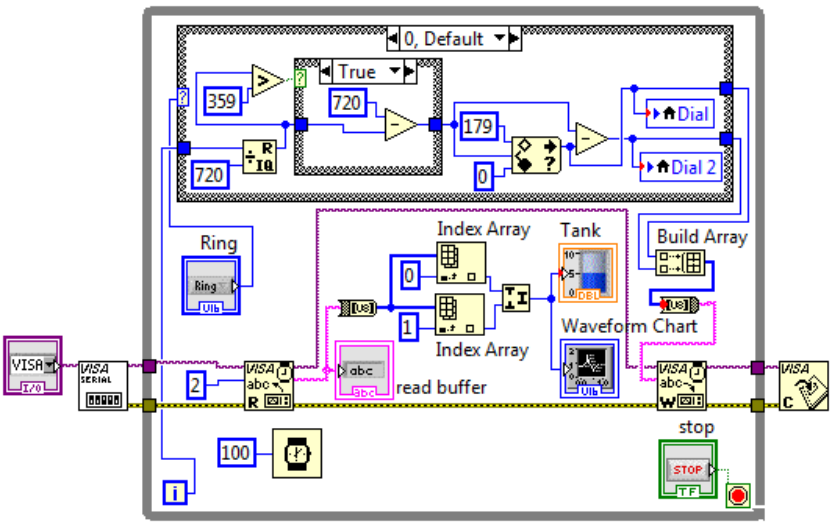
3. Buka LabVIEW, dan gabungkan program LabVIEW Tahap Pertama dengan Tahap Kedua, seperti ditunjukkan pada Gambar 11.38.



Gambar 11.37 Rangkaian pengawatan Arduino dengan sensor Ultrasonik dan 2 buah motor mini servo



Gambar 11.38 Gabungan program Tahap Pertama dengan Tahap Kedua

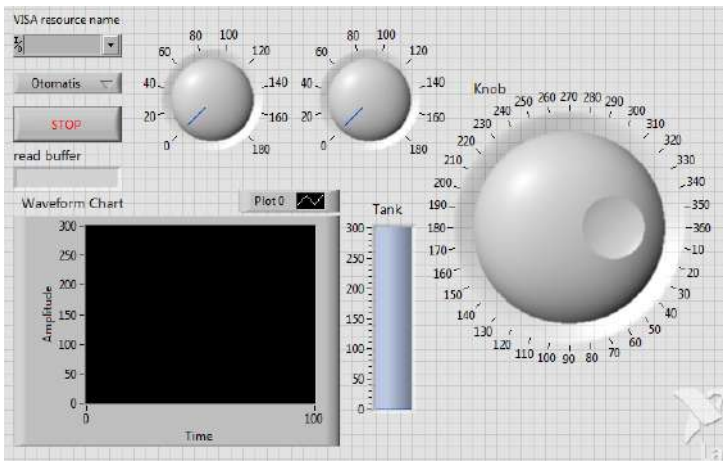


Gambar 11.39 Gabungan program Tahap Pertama dengan Tahap Kedua

4. Sekalipun 2 buah objek **Dial** sebenarnya sudah bisa memberikan informasi mengenai posisi sensor Ultrasonik, namun agar lebih jelas, maka tambahkan sebuah objek **Knob** dari kategori **Num Ctrls** di palet Controls. Buat jangkauan Maksimumnya dari 10 menjadi 360, dengan cara meng-klik angka 10, dan ketik angka 360.

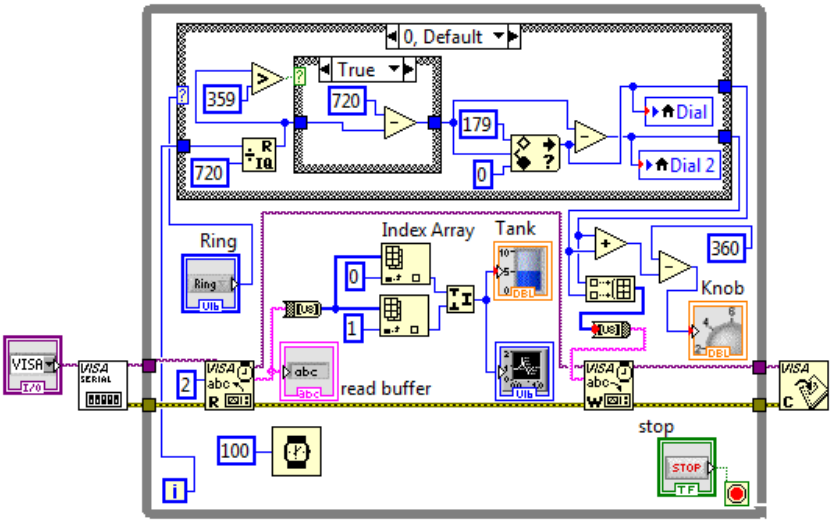
5. Perbesar ukuran **Knob**. Kemudian tempatkan pointer mouse pada strip nilai skala sehingga icon pointer berubah menjadi tanda panah setengah lingkaran. Dengan pointer seperti itu, tarik strip tersebut sehingga angka 360 bertumpukan dengan angka 0. Kemudian geser strip nilai tersebut dengan memutar pointer sehingga nilai 360 berada di samping kanan.

6. Pada jendela Block Diagram, tempatkan icon **Knob** ini di dalam **While Loop**. Karena fungsinya sebagai penampil posisi sensor Ultrasonik, maka ubah tipe objek **Knob** ini dari Control menjadi Indicator, dengan cara meng-klik kanan icon tersebut, dan memilih **Change to Indicator**.



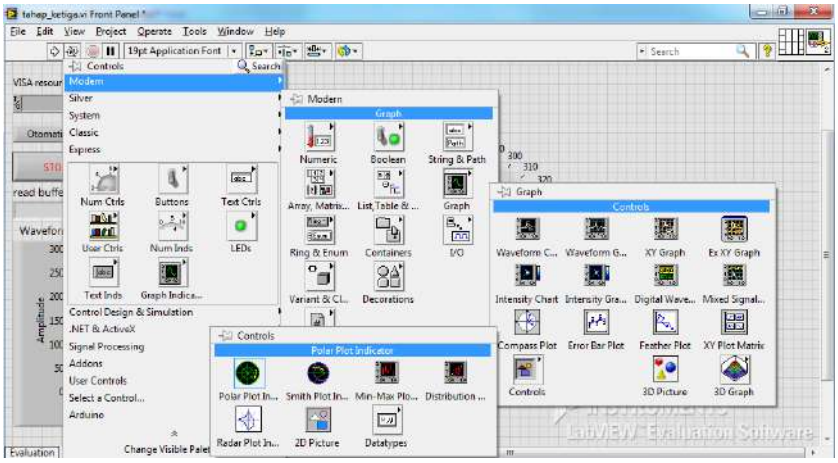
Gambar 11.40 Menambahkan Knob untuk menampilkan posisi sensor ultrasonik, dengan nilai 0-360, yang dimulai dari samping kanan

7. Hubungkan input **Knob** ini dengan penjumlahan data kedua motor mini servo. Berhubung putaran knob berlawanan arah dengan putaran motor mini servo, maka kurangkan nilainya pada angka 360.



Gambar 11.41 Input Knob diperoleh dari jumlah data kedua motor servo, yang kemudian dikurangkan pada angka 360

8. Kemudian, untuk memetakan jarak objek di sekitar sensor Ultrasonik dalam bentuk grafik, maka ambil objek **Polar Plot Indicator** dari kategori **Modern**, **Graph**, **Controls** di palet Controls.



Gambar 11.42 Ambil Polar Plot Indicator dari kategori Graph, Controls

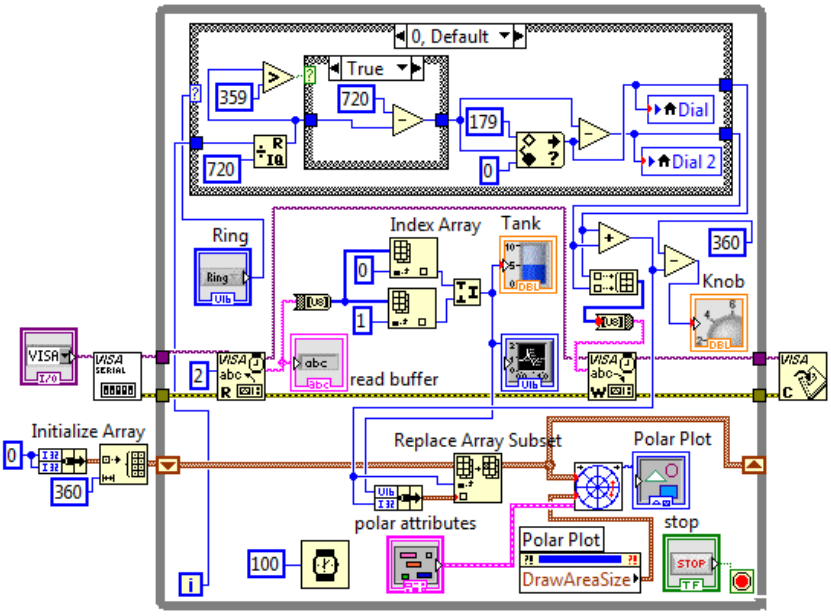
9. **Polar Plot Indicator** ini sebenarnya hampir mirip dengan **XY Graph**, yaitu memerlukan input berupa Array dari Cluster 2 data. Apabila di **XY Graph**, 2 data tersebut adalah nilai X dan Y (disebut koordinat Cartesian), maka di **Polar Plot**, 2 data tersebut adalah nilai jarak dan sudut (disebut koordinat Polar).

10. Karena memerlukan data berupa **Array Cluster**, maka tambahkan fungsi **Initialize Array** untuk menciptakan data **Array Cluster** dengan data (0,0) sebanyak 360 buah, sesuai dengan jumlah data sudut.

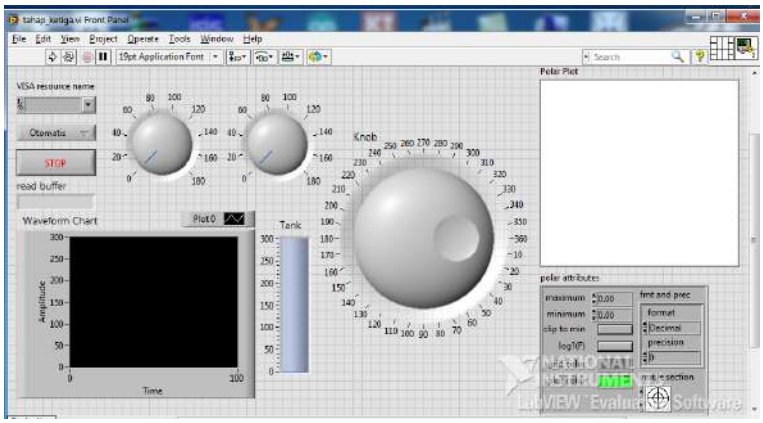
11. Kemudian tambahkan **Shift Register** untuk menyimpan dan mengumpulkan data jarak dan sudut, di mana nilai awal dari **Shift Register** ini diperoleh dari fungsi **Initialize Array** di atas.

12. Berhubung data harus selalu di-update ketika membaca jarak pada posisi yang sama, maka tambahkan fungsi **Replace Array Subset**.

13. Terakhir, agar tampilan pada **Polar Plot Indicator** dapat diatur, maka klik kanan pada kaki input ***polar attributes*** dan pilih **Create Control**.

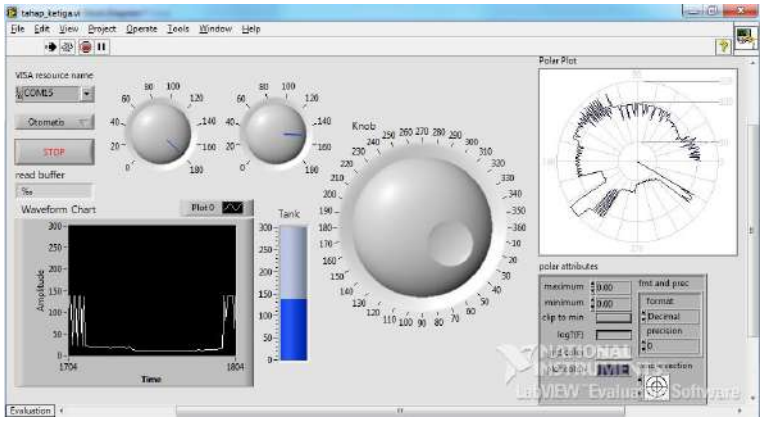


Gambar 11.43 Menambahkan fungsi Initialize Array, Shift Register, Replace Array Subset dan polar attributes untuk Polar Plot Indicator

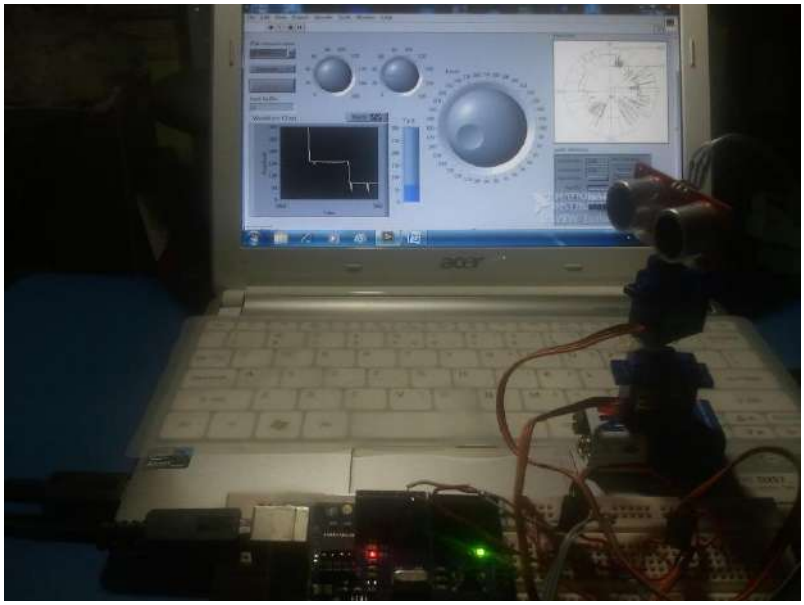


Gambar 11.44 Tampilan Front Panel pada Tahap Ketiga

14. Jalankan program dengan menekan tombol **Run**. Ganti warna **grid color** dan **plot color** pada kotak **polar attributes** untuk mendapatkan gambar pemetaan yang lebih jelas.



Gambar 11.44 Tampilan Front Panel saat program dijalankan

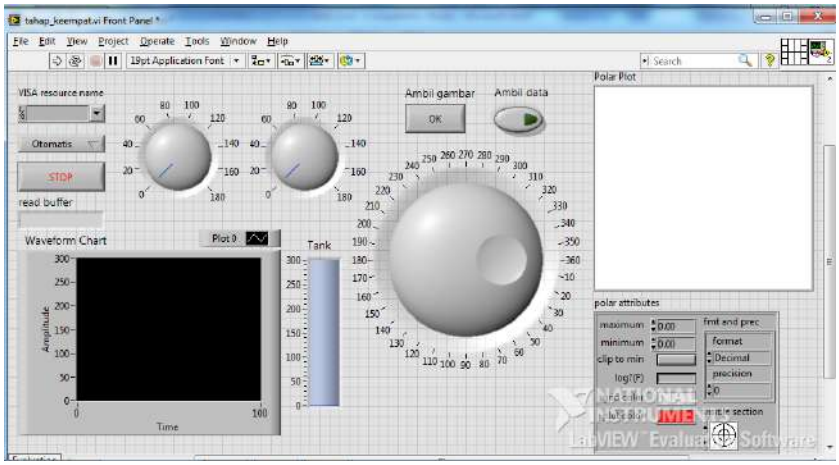


Gambar 11.45 Foto hasil program Tahap Ketiga

11.7 Tahap Keempat

Berikut langkah-langkah pembuatan Tahap Keempat:

1. Buka jendela Front Panel, dan tempatkan objek **Push Button** dan **OK Button** yang diambil dari kategori **Buttons & Switches** di palet Controls. Ganti nama label **Push Button** dengan **Ambil Data**, dan **OK Button** dengan **Ambil Gambar**.

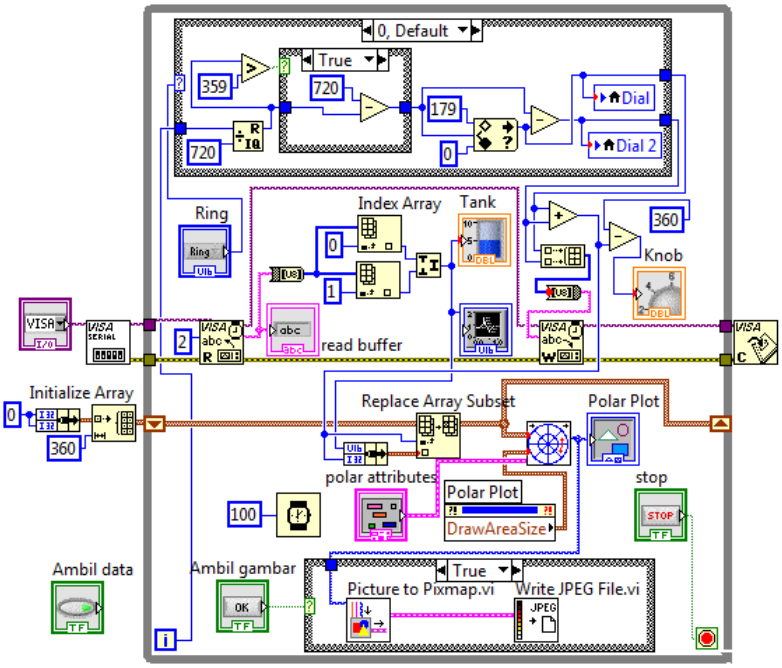


Gambar 11.46 Menempatkan objek OK Button (Ambil Gambar) dan Push Button (Ambil Data) pada Front Panel

2. Tombol **Ambil Gambar** digunakan untuk mengambil gambar grafik **Polar Plot Indicator** dan menyimpannya dalam sebuah file. Untuk itu buka Block Diagram, tambahkan **Case Structure** dengan terminal **Selector**-nya terhubung ke icon tombol **Ambil Gambar**. Kemudian di dalam blok **Case True**, tambahkan fungsi **Picture to Pixmap** dan fungsi **Write JPEG file**. Untuk blok **Case False**, biarkan kosong.

3. Fungsi **Picture to Pixmap** di sini digunakan untuk mengubah sebuah gambar menjadi data yang bisa disimpan dalam sebuah file. Fungsi **Write JPEG File** digunakan untuk mem-format file menjadi file

gambar JPEG. Kaki *path* fungsi **Write JPEG File** ini seharusnya diisi dengan nama file dan lokasi/direktorinya. Namun apabila tidak diisi, maka program akan menampilkan kotak dialog untuk meminta pengguna memasukkan nama dan lokasi file JPEG tersebut.



Gambar 11.47 Menambahkan fungsi-fungsi untuk tombol Ambil Gambar

4. Berbeda dengan tombol **Ambil Gambar** yang bisa sesaat saja menyimpan file gambar, tombol **Ambil Data** membutuhkan beberapa waktu untuk bisa menyimpan data, sehingga jenis tombol yang digunakan adalah tombol yang bisa mengunci, yaitu **Push Button**. Dengan **Push Button** tersebut, pengguna dapat menentukan kapan penyimpanan data akan dimulai dan dihentikan.

5. Apabila pada Sub Bab 8.3.3 menggunakan fungsi **Write to Spreadsheet file** yang ditempatkan di luar **While Loop**, dan bekerja

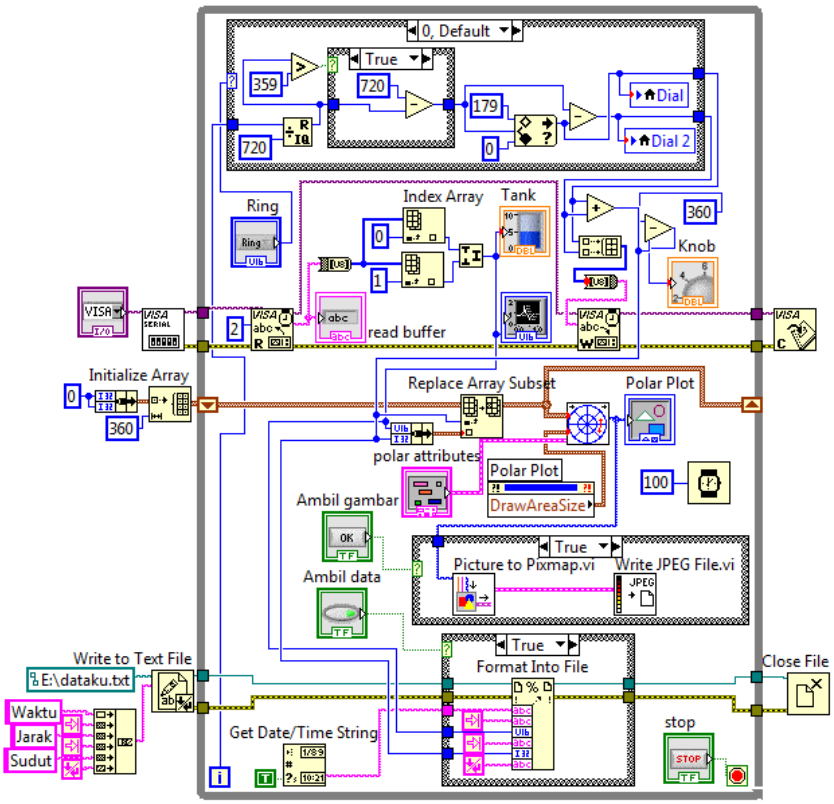
menyimpan data dari saat **While Loop** dimulai hingga dihentikan, maka untuk penyimpanan sewaktu-waktu ini digunakan fungsi yang ditempatkan di dalam **While Loop**, yaitu menggunakan fungsi **Format Into File**, yang sebelumnya diawali dengan fungsi **Write to Text File** dan diakhiri dengan fungsi **Close File**.

6. Push Button kemudian dihubungkan dengan terminal **Selector Case Structure** untuk mengaktifkan atau mematikan fungsi **Format Into File**, yang ditempatkan di blok **Case True**. Untuk blok **Case False**, teruskan saja kedua garis data dari terminal kiri ke terminal kanan.

7. Data yang akan disimpan adalah data **waktu**, **jarak** dan **sudut**. Untuk membuat judul pada isi file, maka tambahkan fungsi **Concatenate String** yang akan menggabungkan ketiga teks tersebut, dengan penambahan **Tab Constant** di tengah-tengahnya, dan diakhiri dengan **End of Line** pada kaki ***Text*** fungsi **Write To Text File**.

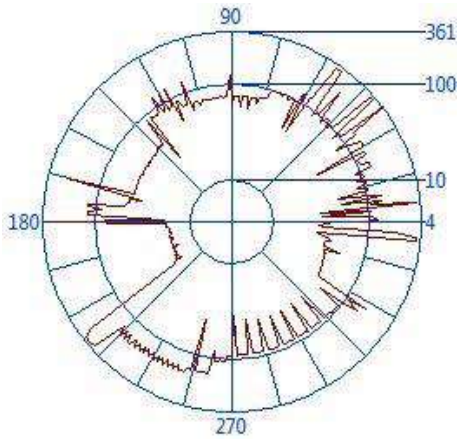
8. Klik kanan kaki ***file*** fungsi **Write To Text File**, dan pilih **Create Constant**. Kemudian isikan dengan nama file dan lokasinya yang akan digunakan untuk menyimpan data nantinya. Apabila kaki ***file*** ini tidak diisi, maka sebuah kotak dialog akan muncul pada saat program dijalankan, yang meminta pengguna untuk memasukkan nama file dan lokasi file tersebut. Untuk menghindari munculnya kotak dialog tersebut, maka pada kaki ***file*** tersebut, dalam contoh di sini diisi dengan **E:\dataku.txt**.

9. Keistimewaan fungsi **Format Into File** di sini adalah menerima tipe data apapun untuk kemudian dijadikan file. Tambahkan fungsi **Get Date/Time String** untuk memberikan nilai waktu (beri **True** pada kaki ***want seconds***), diikuti dengan **Tab Constant**, kemudian data jarak objek, **Tab Constant** lagi, kemudian data sudut sensor, dan diakhiri dengan fungsi **End Of Line** (atau Enter).

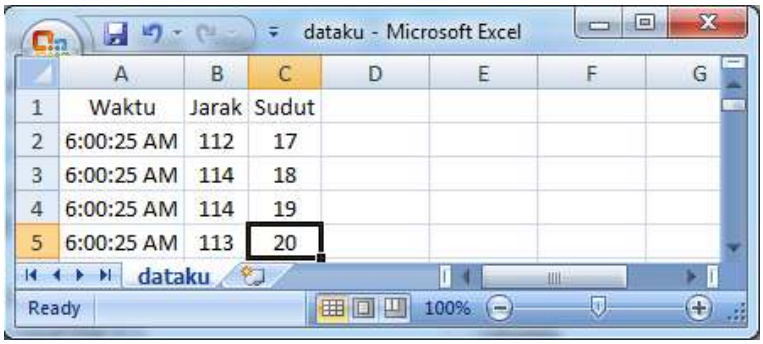


Gambar 11.48 Menambahkan fungsi-fungsi untuk tombol Ambil Data

10. Jalankan program dengan menekan tombol **Run**, kemudian tekan tombol **Ambil Data** hingga menyala. Tunggu hingga **Polar Plot Indicator** menampilkan gambar lingkaran penuh, kemudian tekan tombol **Ambil Gambar**. Maka akan muncul kotak dialog yang menanyakan nama file untuk menyimpan gambar dan lokasinya. Isi kotak dialog tersebut, maka sebuah file data dalam bentuk gambar JPG diciptakan. Berikutnya hentikan pengambilan data dengan menekan kembali tombol **Ambil Data** hingga padam. Maka sebuah file data (**dataku.txt**) diciptakan. Gambar 11.49 dan Gambar 11.50 memperlihatkan isi file gambar dan file teks tersebut.



Gambar 11.49 File hasil Ambil Gambar

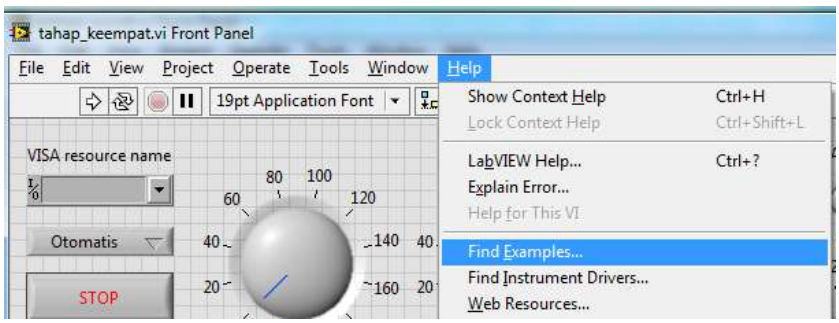


Gambar 11.50 File dataku.txt dibuka dengan Excel

11.8 Tahap Kelima

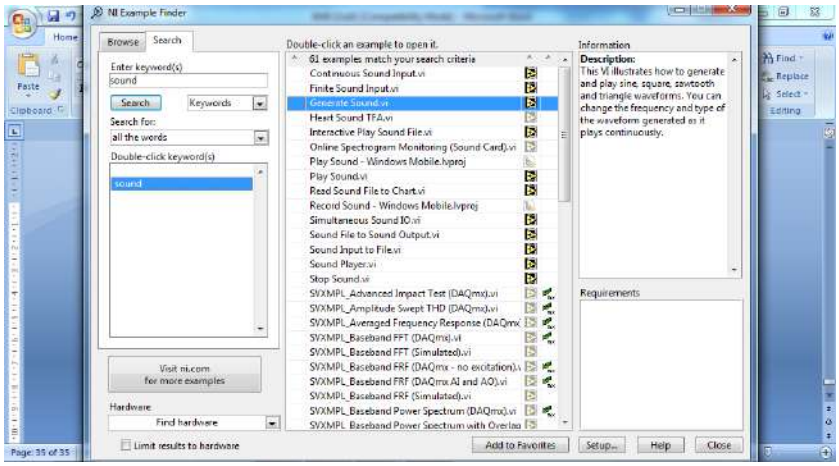
Berikut langkah-langkah pembuatan Tahap Kelima:

1. Salah satu sumber ide untuk pembuatan program dalam LabVIEW adalah contoh-contoh program yang telah disediakan oleh LabVIEW. Untuk mengetahui bagaimana membangkitkan suara, maka buka contoh program yang berkaitan dengan suara (sound), dengan membuka menu **Help**, dan kemudian pilih **Find Examples**.



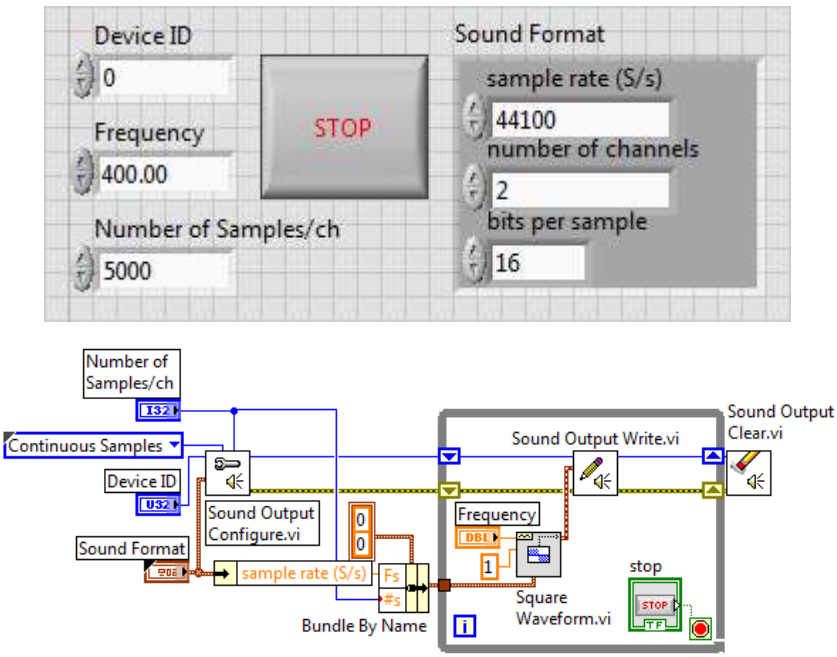
Gambar 11.51 Membuka menu Help dan memilih Find Examples

2. Dari jendela **Find Examples** yang muncul, ketik kata **Sound** pada kolom di bawah **Enter keywords** dan kemudian tekan tombol **Search**, dan kemudian klik 2 kali pada **keyword** yang muncul. Dari daftar contoh program yang muncul, pilih **Generate Sound.vi**.



Gambar 11.52 Memilih Generate Sound.vi pada daftar contoh program

3. Sederhanakan program **Generate Sound.vi** dengan menghilangkan fungsi **Sound Output Set Volume.vi**, fungsi **Waveform Type** dan **Case Structure**, sisakan hanya fungsi **Square Waveform**, dan hilangkan grafik **Generate Sound**, sehingga menjadi seperti berikut:

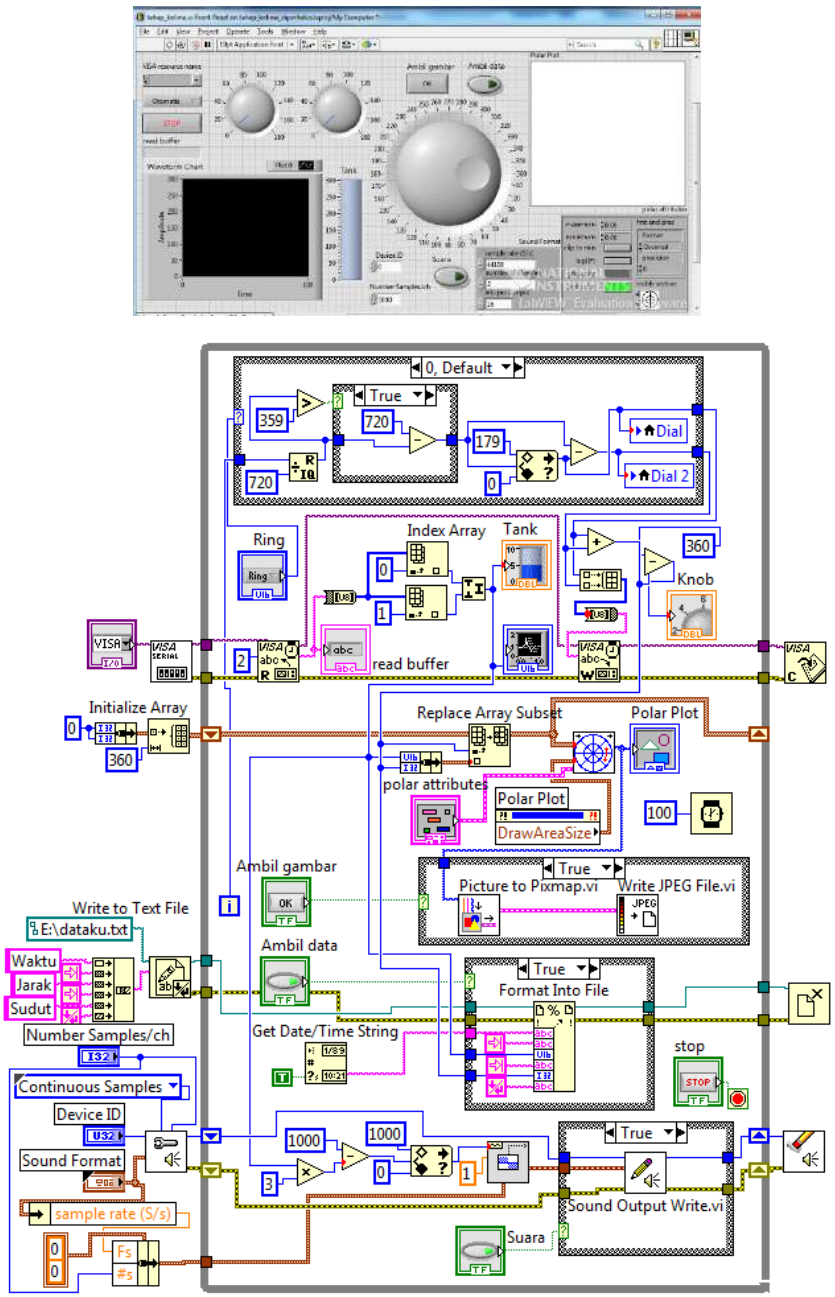


Gambar 11.53 Menyederhanakan program Generate Sound.vi

4. Jalankan program dengan menekan tombol **Run**, dan naik-turunkan besarnya frekuensi dari 0-1000, maka seharusnya terdengar suara dengan tinggi nada yang turun naik mengikuti besarnya frekuensi.

5. Buka kembali program Ultrasonik Radar, dan tambahkan program pembangkit suara seperti pada Gambar 11.53 di atas. Kemudian agar suara semakin tinggi frekuensinya ketika jarak objek semakin dekat, maka tambahkan perhitungan **1000–3 x jarak objek** pada nilai input frekuensi. Kemudian agar nilai frekuensi berada antara nilai 0 – 1000, maka tambahkan fungsi **In Range & Coerce**.

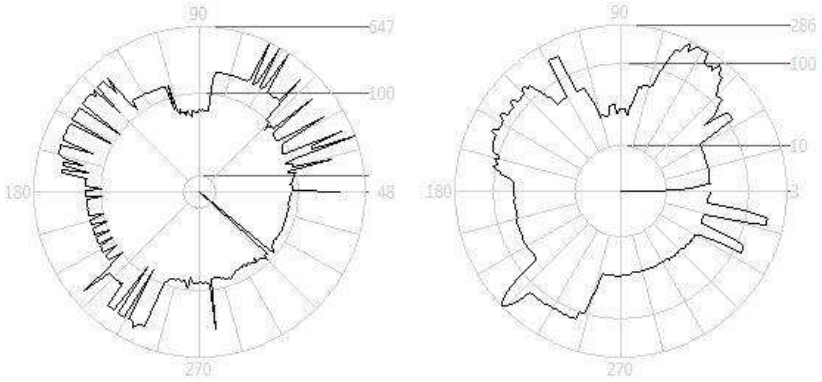
6. Tambahkan sebuah **Push Button** yang akan menghidupkan atau mematikan suara. Hubungkan **Push Button** ini dengan sebuah **Case Structure** yang dipasang pada fungsi **Sound Output Write**.



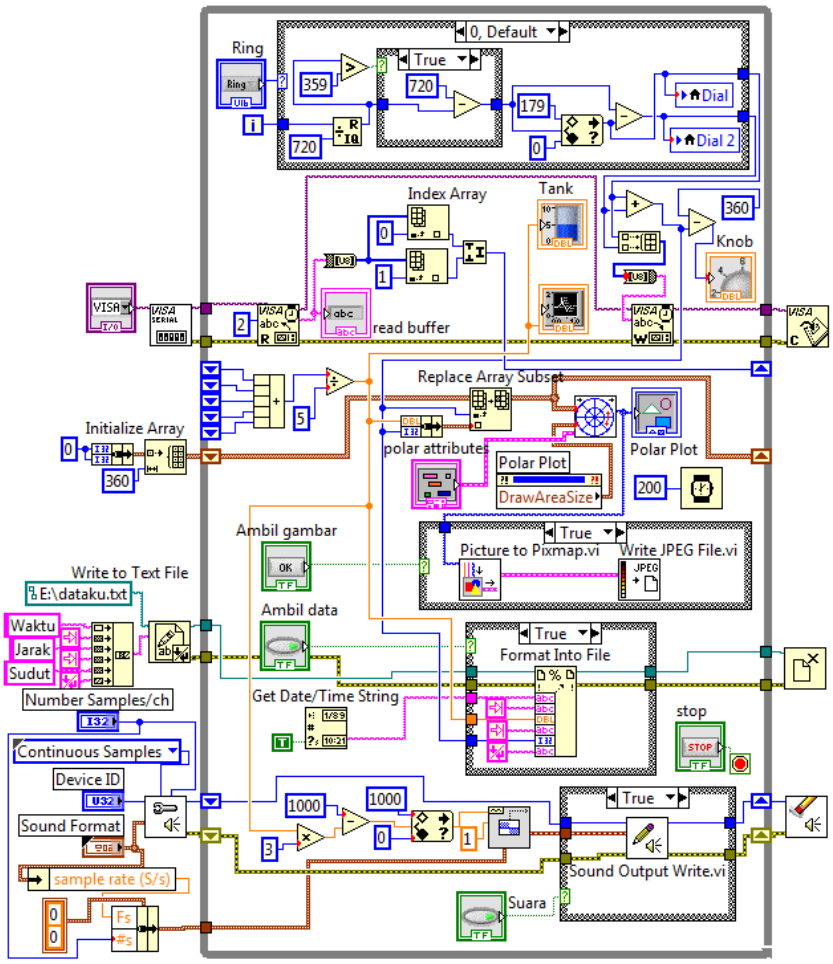
Gambar 11.54 Front Panel dan Block Diagram Tahap Kelima

7. Jalankan program dengan menekan tombol **Run**. Perhatikan bahwa program mengalami komunikasi yang buruk (hilang data) akibat fungsi pembangkitan suara ternyata menghabiskan waktu untuk komunikasi. Untuk mengatasi hal ini, maka perbesar tunda waktu dari 100 ms menjadi 200 ms, baik di sisi Arduino, maupun di sisi LabVIEW. Jadi untuk program Arduino, gunakan program yang sama dengan program di Tahap Ketiga, hanya pada baris terakhir, ganti **delay(100);** menjadi **delay(200);**. Begitu pula di LabVIEW, ganti nilai input fungsi **Wait(ms)** dari yang semula **100** menjadi **200**.

8. Terakhir, akibat adanya getaran pada sensor Ultrasonik karena digerakkan oleh motor servo, maka hasil pembacaan jarak menjadi tidak mulus, atau beriak. Untuk mengatasi hal ini, yaitu supaya hasil pembacaan jarak lebih halus grafiknya, maka gunakan rata-rata dari beberapa data hasil pembacaan. Sebagai contoh di sini, dilakukan rata-rata dari 5 data pembacaan. Tambahkan fungsi **Shift Register** untuk menyimpan data pembacaan sebelumnya. Tarik ke bawah terminal kiri **Shift Register** hingga ada 5 data tersedia. Kemudian jumlahkan dan bagi dengan angka 5. Maka grafik hasil pembacaan jarak menjadi lebih halus seperti terlihat pada gambar berikut:



Gambar 11.55 Kiri sebelum dirata-rata, kanan setelah dirata-rata



Gambar 11.56 Front Panel dan Block Diagram Tahap Kelima dengan data yang diperhalus menggunakan rata-rata dari 5 data pembacaan

Pembuatan Ultrasonik Radar telah selesai sampai di sini. Bab berikutnya, akan menyajikan pembuatan Infrared Radar yang hampir mirip dengan Ultrasonik Radar, hanya bedanya komunikasi yang digunakan adalah komunikasi serial dengan Firmata. Apabila pembaca merasa kesulitan membaca program karena gambarnya tidak terlihat jelas, maka pembaca dapat membuka programnya di CD pendukung buku ini.

BAB 12
INFRARED RADAR

12.1 Apa itu Infrared Radar

Alat Infrared Radar ini sebenarnya sama fungsinya dengan Ultrasonik Radar, yaitu merupakan alat pengukur jarak yang dapat memetakan jarak objek-objek di sekitarnya, menggunakan sensor Jarak Infrared.

Hanya saja sensor Jarak Infared ini tidak sejauh sensor Ultrasonik jangkauan pendeteksiannya. Jangkauan maksimum sensor Ultrasonik bisa mencapai 3 m lebih, sedangkan sensor Jarak Infrared maksimum hanya sampai 80 cm.

Namun kelebihannya, pembacaan nilai jarak dengan sensor Infrared ini dapat dilakukan hanya dengan membaca sinyal analog yang dikeluarkan oleh output sensor. Bandingkan dengan sensor Ultrasonik yang harus menambahkan instruksi pengukuran lebar pulsa (**pulsein**) pada kode program Arduinonya. Akibatnya, untuk sensor Ultrasonik, komunikasi serial Firmata tidak dapat digunakan, karena belum tersedia fungsi pengukuran lebar pulsa pada **Firmata**. Sedangkan untuk sensor Infrared, komunikasi serial **Firmata** dapat digunakan, karena sudah tersedia fungsi untuk pembacaan sinyal analog.

Dengan pembuatan Infrared Radar ini, diharapkan pembaca dapat lebih memahami pemakaian komunikasi serial **Firmata**, dan kemudian dapat membandingkan hasilnya dengan komunikasi serial biasa di Bab 11.

12.2 Spesifikasi Alat

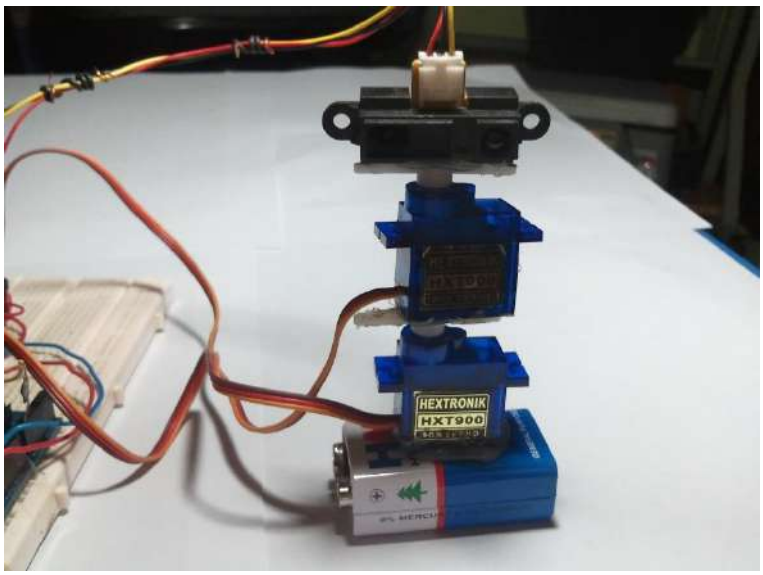
Sama seperti spesifikasi pada Ultrasonik Radar, maka Infrared Radar ini juga memiliki spesifikasi atau kemampuan sebagai berikut:

5. Mampu mendeteksi jarak semua objek yang berada dalam radius daerah jangkauannya.
6. Mampu memetakan hasil pendeteksian tersebut dalam bentuk grafik koordinat polar.
7. Mampu menyimpan hasil pemetaan tersebut dalam bentuk file.
8. Mampu memberikan indikator suara untuk jarak objek tertentu.

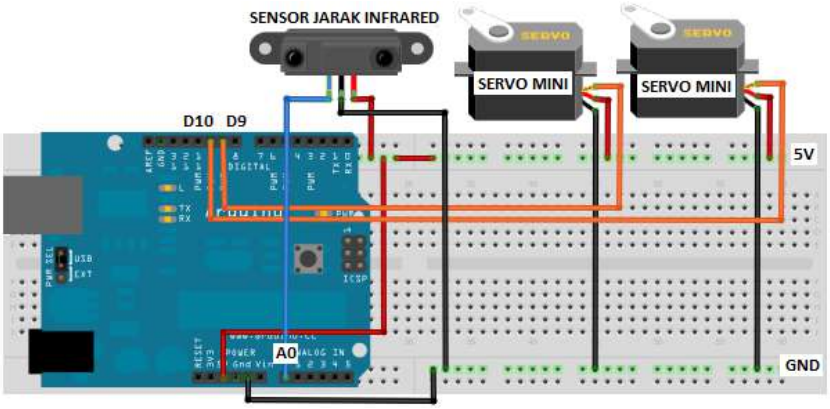
12.3 Cara Pembuatan

Untuk bisa menghasilkan kemampuan sesuai spesifikasi di atas, dibutuhkan 3 jenis komponon, yaitu Arduino (1 buah), motor mini servo (2 buah) dan sensor Jarak Infrared (1 buah). Sensor Jarak Infrared ini tersedia dengan beberapa tipe, di antaranya: GP2D120, GP2D12, GP2Y0A21, dll. Informasi toko yang menyediakan sensor Jarak Infrared ini dapat dilihat pada Lampiran.

Pasang sensor Jarak Infrared di atas baling-baling motor servo, seperti ditunjukkan pada Gambar 12.1. Gambar pengawatan rangkaian Infrared Radar ini dapat dilihat pada Gambar 12.2, di mana kabel kuning sensor Jarak Infrared dihubungkan dengan kaki analog A0 Arduino. Kabel merah dan kabel hitam dihubungkan dengan kaki 5V dan GND Arduino.



Gambar 12.1 Pemasangan sensor Jarak Infrared



Gambar 12.2 Rangkaian Infrared Radar

Tahapan pembuatan sama seperti pada Ultrasonik Radar, yaitu:

6. Tahap pertama: melakukan pembacaan jarak objek menggunakan sensor Jarak Infrared yang hasilnya ditampilkan dalam bentuk grafik di software LabVIEW.

7. Tahap kedua: mengendalikan kedua motor mini servo secara otomatis (berputar bolak-balik) dan secara manual dengan objek Dial di software LabVIEW.
8. Tahap ketiga: melakukan pembacaan jarak objek di semua sudut putar sensor Jarak Infrared dan memetakan hasilnya pada grafik dengan koordinat polar di software LabVIEW.
9. Tahap keempat: menambahkan tombol untuk menyimpan hasil pembacaan jarak yang dilengkapi dengan data waktu.
10. Tahap kelima: membangkitkan suara yang besarnya frekuensi suara tersebut ditentukan oleh jarak objek terhadap sensor, semakin dekat semakin tinggi frekuensinya.

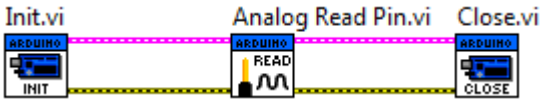
Berikut satu-persatu uraian dari tahapan pembuatan tersebut.

12.4 Tahap Pertama

Berikut langkah-langkah pembuatan Tahap Pertama:

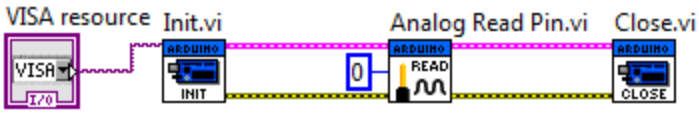
41. Hubungkan board Arduino dengan port USB komputer.
42. Buka software Arduino, dan **Upload** kode Firmata atau **LVIFA_Base** ke Arduino, yang dapat diambil di direktori:

    C:\Program Files\National Instruments\LabVIEW2011\vi.lib\ LabVIEW Interface for Arduino\Firmware\LVIFA_Base
43. Sekali kode Firmata tersebut tertanam di Arduino, maka untuk seterusnya pembaca tidak lagi perlu memprogram Arduino. Cukup memprogram dari LabVIEW saja. Untuk itu buka LabVIEW.
44. Di jendela Block Diagram LabVIEW, tambahkan fungsi Toolkit Arduino, yaitu **LIFA Init**, **LIFA Analog Read Pin** dan **LIFA Close**, dan hubungkan ketiganya seperti berikut:



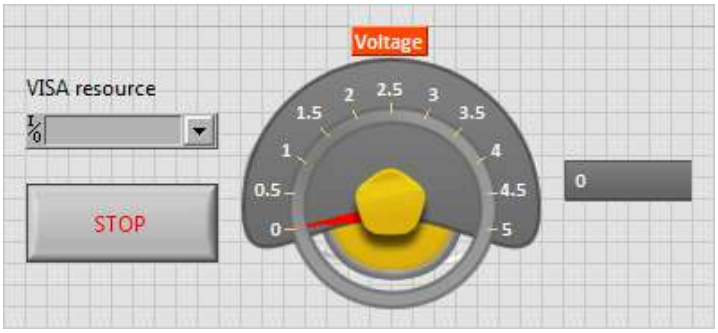
Gambar 12.3 Fungsi-fungsi Toolkit Arduino untuk membaca nilai analog, LIFA Init, LIFA Analog Read Pin dan LIFA Close

45. **Create Control** pada kaki input ***VISA resource*** **LIFA Init**, dan **Create Constant** pada kaki ***Analog Input Pin***. Biarkan nilainya 0, karena sensor Jarak Infrared dipasang pada kaki A0 Arduino. Untuk kaki ***Board Type*** **LIFA Init**, apabila pembaca menggunakan tipe **Uno**, atau **Duemilanove**, maka biarkan kaki tersebut tidak terhubung.

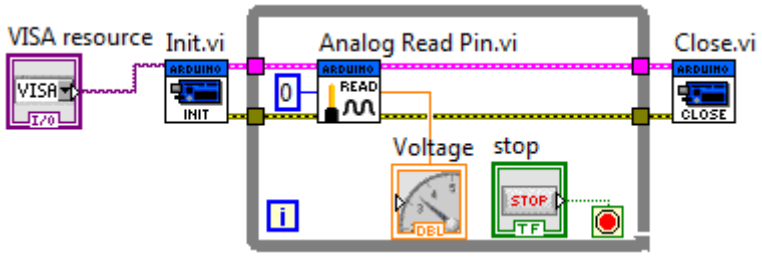


Gambar 12.4 Create Control pada kaki input LIFA Init dan Create Constant pada kaki input LIFA Analog Read Pin

46. **Create Indicator** pada kaki output **LIFA Analog Read Pin**, maka akan muncul icon voltage di Block Diagram dan tampilan **voltage** di Front Panel. Tambahkan **While Loop** pada fungsi **LIFA Analog Read Pin** ini, beserta input outputnya, maka selesailah program untuk membaca sinyal analog di kaki A0 Arduino dan menampilkannya di LabVIEW.



Gambar 12.5 Tampilan Voltage pada Front Panel



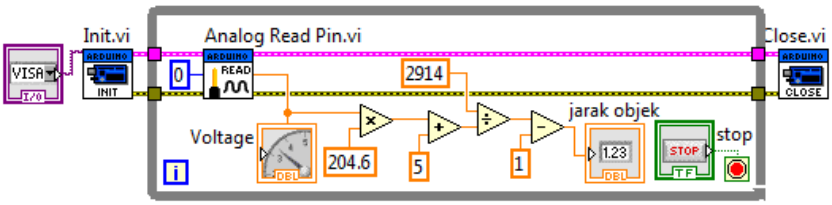
Gambar 12.6 Menambahkan Indicator pada kaki output LIFA Analog Read Pin dan struktur While Loop pada fungsi tersebut

47. Jalankan program dengan menekan tombol **Run**. Tunggu beberapa saat. Perhatikan bahwa nilai output pada tampilan Voltage berbanding terbalik dengan jarak objeknya. Untuk bisa menghasilkan jarak objek, maka perlu penambahan rumus berikut:

Jarak objek (dalam cm) = (2914 / ((voltage x 204,6) + 5)) - 1

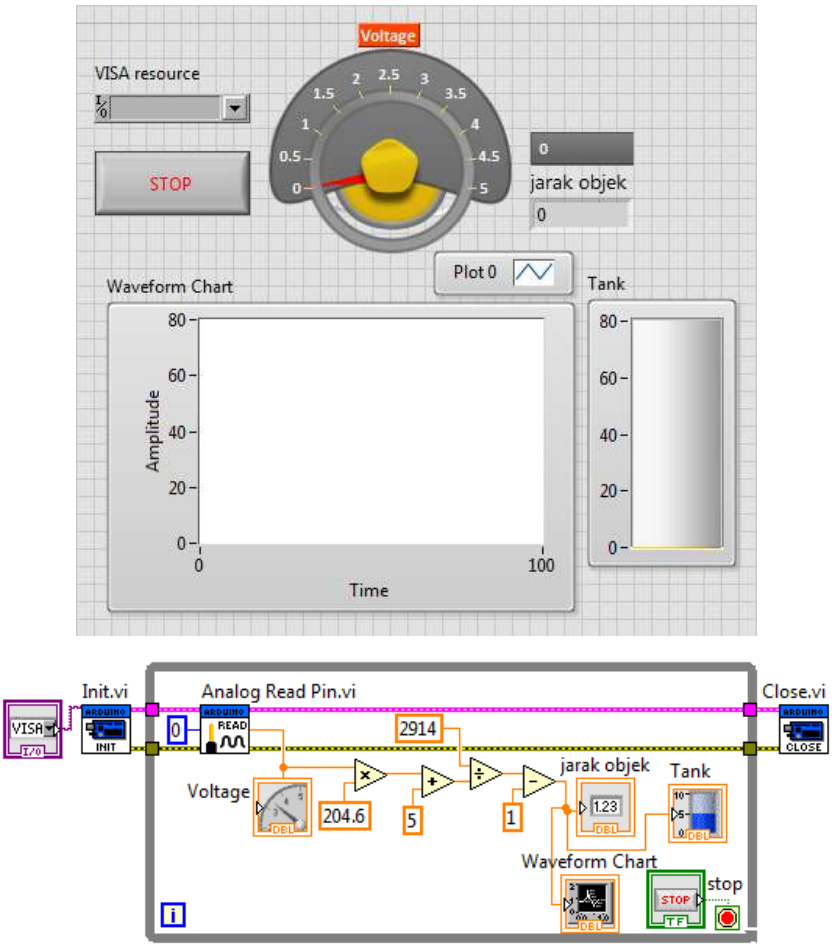
Keterangan: rumus di atas diperoleh dari artikel di internet dengan judul **Linearizing Sharp Ranger Data**, dimana sebenarnya dibuat khusus untuk sensor GP2D120. Namun demikian, apabila ketelitian pengukuran tidak terlalu penting, maka rumus di atas dapat juga digunakan untuk tipe sensor Jarak Infrared yang lain.

48. Buat rumus di atas dengan input dari icon voltage. Kemudian pada output fungsi Subtract (-), klik kanan dan pilih **Create Indicator** untuk menampilkan hasil jarak objek di jendela Front Panel.



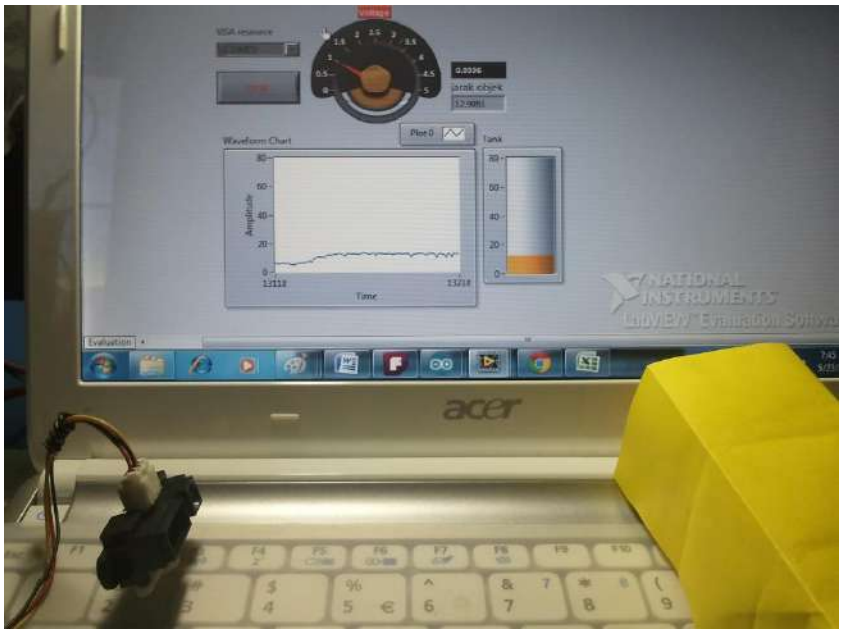
Gambar 12.7 Menambahkan persamaan untuk mengubah nilai tegangan analog sensor menjadi data jarak objek dalam cm

49. Agar menarik, maka tambahkan di jendela Front Panel, objek **Waveform Chart** dan **Tank** dari **Numeric** dan **Graph** dari kategori **Silver** di palet Controls. Buat nilai maksimum keduanya sebesar **80**, dengan **Autoscale Waveform Chart** untuk sumbu **Y** tidak diaktifkan atau tidak dicentang. Hubungkan input kedua icon objek tersebut dengan output dari fungsi Subtract (-) di jendela Block Diagram.



Gambar 12.8 Menambahkan objek Waveform Chart dan Tank dari kategori Silver, dengan nilai Maksimum 80

50. Jalankan program dengan menekan tombol **Run**. Tunggu beberapa saat. Kemudian letakkan sebuah benda di depan sensor Jarak Infrared, maju mundurkan benda tersebut dan amati nilai jaraknya yang ditampilkan dalam bentuk angka, grafik dan isi tangki.



Gambar 12.9 Foto hasil program pembacaan sensor Jarak Infrared

Catatan: Apabila program di LabVIEW sudah benar, tidak ada pesan error, dan sudah sesuai dengan yang ada di buku ini, namun hardware tidak bekerja seperti yang dituliskan di buku ini, padahal Arduino sudah terhubung dengan komputer dan rangkaian sudah benar, maka 2 hal yang mungkin bisa menjadi solusi adalah:

1. **Upload** sekali lagi kode Firmata **LVIFA_Base** ke Arduino.
2. Tambahkan Adaptor, ada kemungkinan tegangan dari port USB tidak cukup kuat menyediakan arus untuk 2 buah motor servo.

12.5 Tahap Kedua

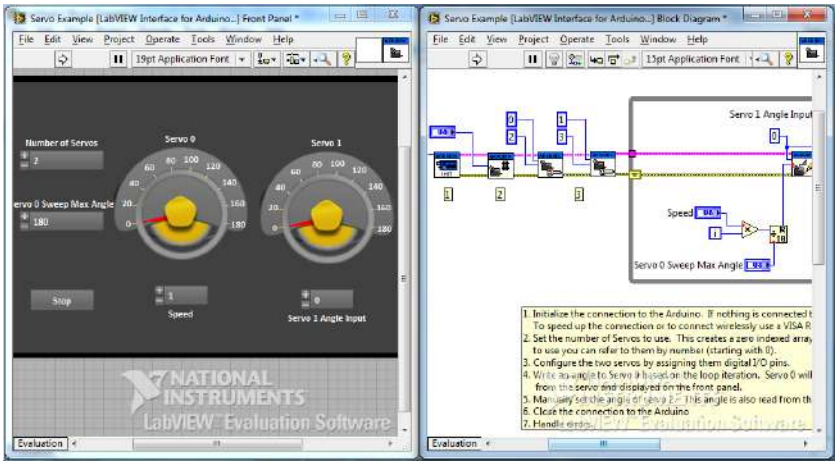
Berikut langkah-langkah pembuatan Tahap Kedua:

22. Karena kode Firmata sudah ditanam di Arduino, maka untuk seterusnya, pemrograman hanya akan dilakukan di LabVIEW.

23. Untuk mengetahui program pengendalian pada 2 buah motor servo dengan fungsi Toolkit Arduino di LabVIEW, pembaca dapat melihat dari contoh program yang disediakan, yaitu di kategori **Examples**.

24. Buka palet Functions di Block Diagram, pilih **Arduino**, pilih **Examples**, ambil icon **Servo Example** dan tempatkan di Block Diagram.

25. Klik 2 kali pada icon **Servo Examples** di Block Diagram, maka akan akan muncul Front Panel dan Block Diagram **Servo Examples**.



Gambar 12.10 Front Panel dan Block Diagram Servo Example

26. Dari jendela Block Diagram **Servo Example** tersebut tampak ada 6 jenis fungsi Toolkit Arduino, namun hanya 5 jenis yang benar-benar diperlukan untuk pengendalian servo pada Tahap Kedua ini, yaitu seperti berikut :

a. **LIFA Init** untuk mengatur seting komunikasi dan membukanya.

b. **LIFA Set Number of Servos** untuk menentukan berapa banyak servo yang akan dikendalikan.

c. **LIFA Configure Servo** untuk penugasan kaki digital mana yang akan digunakan untuk mengendalikan servo.

d. **LIFA Servo Write Angle** untuk menentukan berapa besar sudut putar servo. Nilai sudut ini dari 0 – 180 derajat.

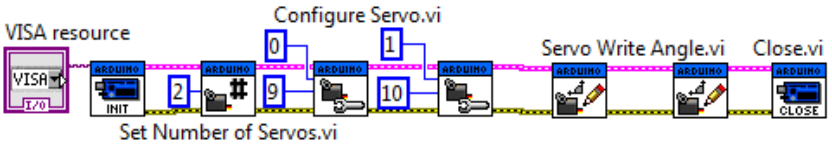
e. **LIFA Close** untuk menutup komunikasi.

27. Buka jendela Block Diagram yang baru, dan tempatkan kelima jenis fungsi Toolkit Arduino di atas pada Block Diagram tersebut. Karena ada 2 motor servo yang akan dikendalikan, maka fungsi **LIFA Configure Servo** dan **LIFA Servo Write Angle** juga harus 2. Kemudian hubungkan semua fungsi tersebut seperti berikut:

Gambar 12.11 Fungsi-fungsi toolkit Arduino untuk pengendalian 2 servo

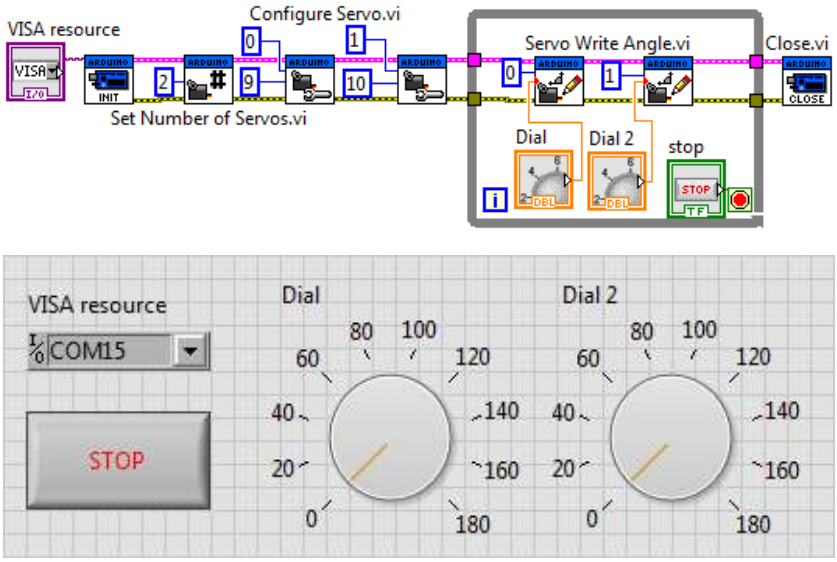
28. **Create Control** pada kaki input ***VISA resource*** **LIFA Init**. Karena ada 2 servo yang harus dikendalikan, **Create Constant** pada kaki input **LIFA Set Number of Servos** dan isi dengan angka 2.

29. **Create Constant** juga pada kaki-kaki input **LIFA Configure Servo** pertama, beri angka 0 untuk kaki ***Servo Number***, dan angka 9 untuk kaki ***DIO Pin***, (karena menggunakan kaki digital D9). Berikutnya untuk **LIFA Configure Servo** kedua, beri angka 1 untuk kaki ***Servo Number***, dan angka 10 untuk kaki ***DIO Pin*** (karena menggunakan kaki digital D10).



Gambar 12.12 Create Control pada kaki input fungsi LIFA Init, dan Create Constant untuk kaki input fungsi-fungsi yang lain

30. Berikutnya untuk **LIFA Servo Write Angle** pertama dan kedua, berturut-turut beri angka 0 dan 1 untuk kaki ***Servo Number***, sedangkan untuk kaki ***Angle (Degrees)*** masing-masing, tambahkan objek **Dial** yang diambil dari kategori **Numeric**, **Silver** di palet Controls di Front Panel. Ubah nilai maksimum **Dial** dari 10 menjadi 180. Agar fungsi ini bekerja terus-menerus, tambahkan **While Loop**.



Gambar 12.13 Menambahkan objek Dial dengan nilai 0-180 dan While Loop pada kedua fungsi LIFA Servo Write Angle

31. Jalankan program dengan menekan tombol Run. Tunggu beberapa saat untuk Arduino bekerja. Kemudian putar objek Dial, baik Dial pertama maupun kedua, seharusnya motor mini servo pertama dan

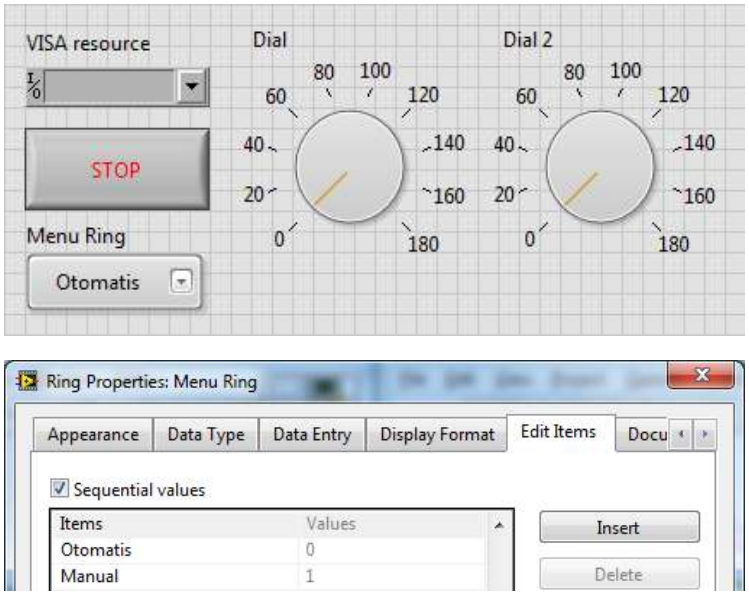
kedua (yang terhubung ke D9 dan D10 Arduino) berputar dengan sudut yang sama seperti objek Dialnya (hanya berlawanan arah).

Catatan: Perhatikan hal yang menarik di sini, yaitu tipe data **Dial** secara default adalah **DBL**, atau bilangan riil, yaitu bilangan dengan angka-angka di belakang koma atau angka-angka pecahan. Dengan komunikasi Firmata, tipe data **DBL** pada objek Dial tersebut dapat dikirimkan dengan baik. Sebaliknya apabila menggunakan komunikasi serial VISA, pengiriman data **DBL** ini membutuhkan penambahan kode (yang tidak mudah) untuk menerjemahkan tipe data **DBL** tersebut. Untuk menghindarinya, maka sebaiknya untuk komunikasi serial VISA, semua data menggunakan tipe data **U8** (***Unsigned Byte***), agar program lebih sederhana dan memastikan komunikasi dapat berjalan dengan baik.

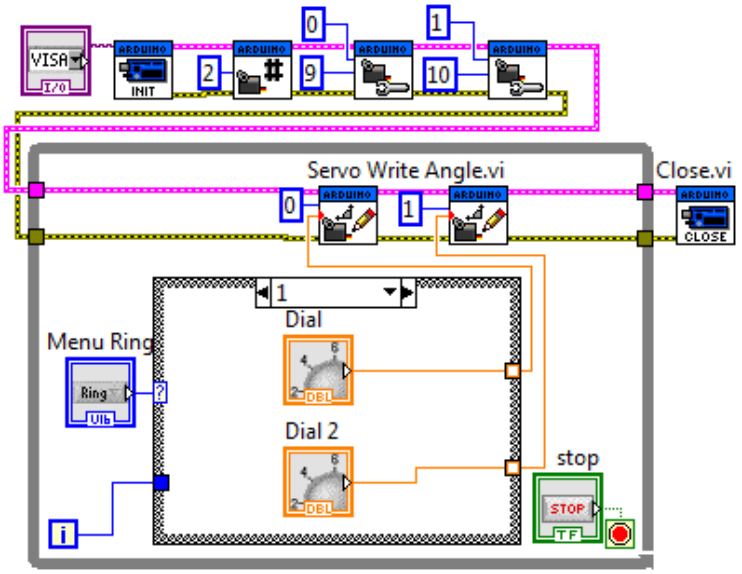
32. Pengendalian servo di atas adalah pengendalian servo secara manual. Untuk memberikan pilihan pengendalian servo secara otomatis, maka tambahkan objek **Menu Ring** dari kategori **Silver** di Front Panel. Klik kanan objek **Menu Ring** tersebut, pilih **Properties,** dan isi pada daftar **Items** di **Edit Items**: Otomatis dan Manual seperti ditunjukkan pada Gambar 12.14.

33. Berikutnya, pada Block Diagram, tambahkan **Case Structure** pada kedua icon objek **Dial**, dan hubungkan terminal **selector Case Structure** tersebut dengan icon **Menu Ring**, seperti ditunjukkan pada Gambar 12.15.

34. Buka blok **Case 0, Default**, dan tambahkan fungsi-fungsi untuk pengendalian otomatis seperti terdapat juga pada Tahap Kedua pembuatan Ultrasonik Radar di Bab 11, yaitu tambahkan fungsi **Quotient & Remainder** untuk membuat siklus setiap 720 kali perulangan. Kemudian tambahkan fungsi **Greater** 359 untuk memisahkan arah putaran (berbalik setiap 360 derajat).

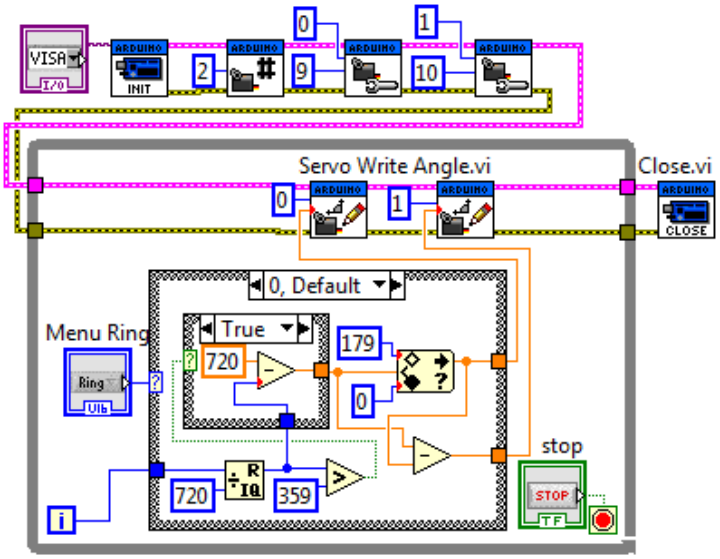


Gambar 12.14 Menambahkan Menu Ring dan mengisi pada daftar Itemsnya, pilihan Otomatis (=0) dan Manual (=1)



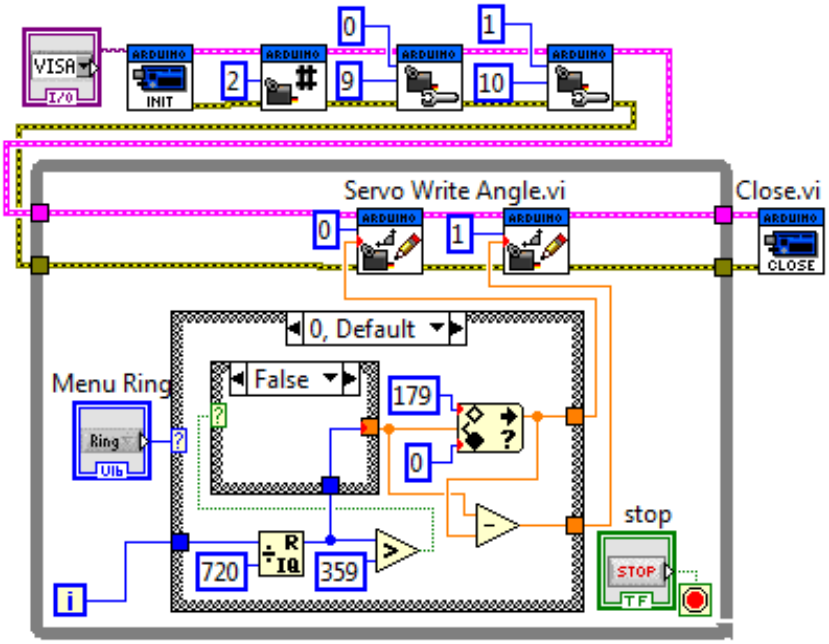
Gambar 12.15 Bila blok Case 1, dilakukan fungsi pengendalian Manual

35. Kemudian, karena motor servo hanya bisa berputar 180 derajat, maka lakukan pembagian sedemikian rupa sehingga untuk nilai 0 – 179 diberikan ke motor mini servo1, sedangkan untuk nilai 180 – 359 diberikan ke motor mini servo2, yang tentunya diubah dulu nilainya menjadi 0-179 dengan mengurangkan pada angka 360. Gunakan fungsi **In Range and Coerce** untuk mengekang nilai output tidak kurang dan tidak lebih dari 0 – 179.



Gambar 12.16 Bila blok Case 0, dilakukan fungsi pengendalian Otomatis

36. Sampai di sini, program bekerja sebagai berikut: ketika tombol Run ditekan, dan pilihan ke mode **Otomatis**, maka motor mini servo1 akan berputar dulu sampai maksimum (180 derajat), diikuti motor mini servo2. Kemudian setelah mencapai maksimum, motor mini servo2 akan berputar berlawanan arah hingga maksimum, kemudian diikuti motor mini servo1, dan proses kembali berulang. Kemudian ketika pilihan diubah ke mode **Manual**, maka kedua motor mini servo dapat diatur secara manual dengan objek **Dial**.

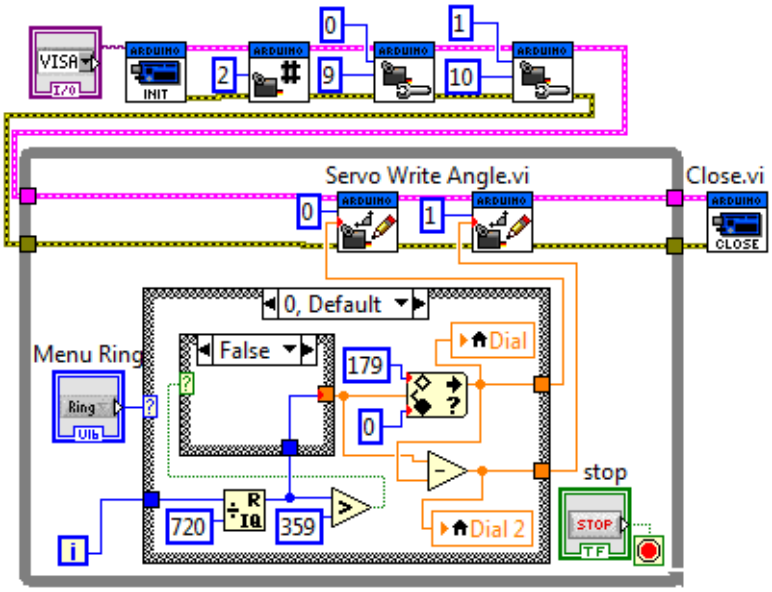


Gambar 12.17 Bila kurang dari 359 (False), maka nilai akan bertambah 1, bila lebih dari 359 (True), maka nilai akan berkurang 1 hingga 0

37. Selanjutnya, diinginkan agar ketika pilihan ke mode **Otomatis**, kedua objek **Dial** tidak hanya diam, tetapi bergerak juga secara otomatis mengikuti posisi baling-baling masing-masing motor mini servo. Dalam kondisi ini, objek **Dial** tidak berlaku sebagai **Control**, tetapi sebagai **Indicator** (menerima sinyal).

38. Untuk bisa membuat objek **Dial** menjadi **Indicator** saat mode **Otomatis**, maka perlu menambahkan **Local Variable** dari **Dial**, yang dimunculkan dengan cara meng-klik kanan **Dial**, dan kemudian memilih **Create Local Variable**. Kemudian tempatkan **Local Variable** untuk kedua **Dial** tersebut pada mode **Otomatis** atau **blok Case 0**, dan hubungkan masing-masing **Local Variable** tersebut dengan nilai yang akan dikirimkan ke masing-masing motor mini servo. Lebih jelasnya lihat Gambar 12.18.

39. Jalankan program dengan menekan tombol **Run**. Tunggu beberapa saat hingga Arduino bekerja. Kemudian perhatikan bahwa ketika mode **Otomatis** dipilih, maka secara otomatis kedua **Dial** akan mengikuti posisi baling-baling motor mini servonya. Sedangkan ketika mode **Manual** dipilih, posisi baling-baling kedua motor mini servo akan bergerak mengikuti posisi objek **Dial** yang diputar.



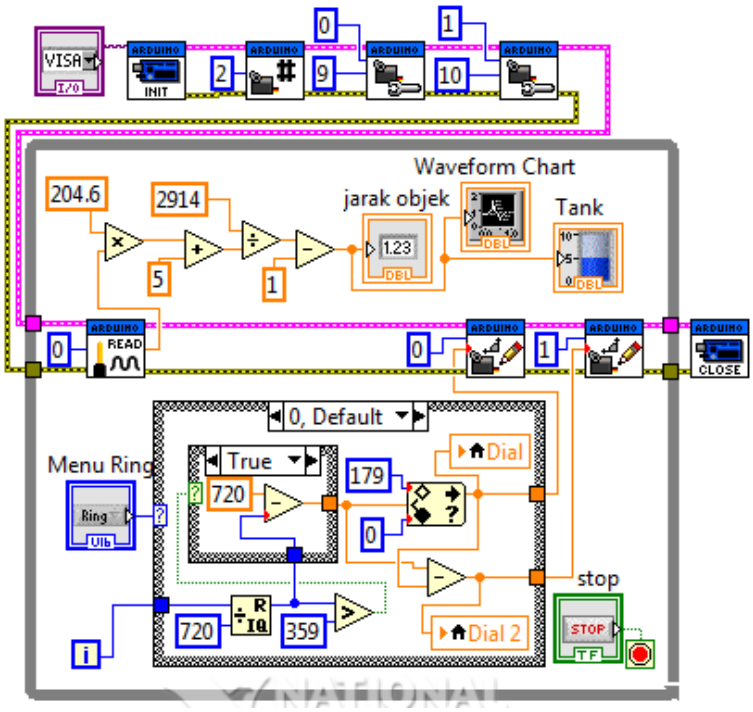
Gambar 12.18 Menambahkan Local Variable kedua Dial untuk membuat kedua Dial menjadi Indicator saat mode Otomatis (=0)

12.6 Tahap Ketiga

Berikut langkah-langkah pembuatan Tahap Ketiga:

15. Program LabVIEW pada Tahap Ketiga ini merupakan penggabungan dari program Tahap Pertama dan Kedua, yang akan membuat sensor Jarak Infrared membaca jarak objek, sementara sensor tersebut diputar oleh kedua motor mini servo.

16. Buka LabVIEW, dan gabungkan program LabVIEW Tahap Pertama dengan Tahap Kedua, dan hilangkan tampilan Voltage.

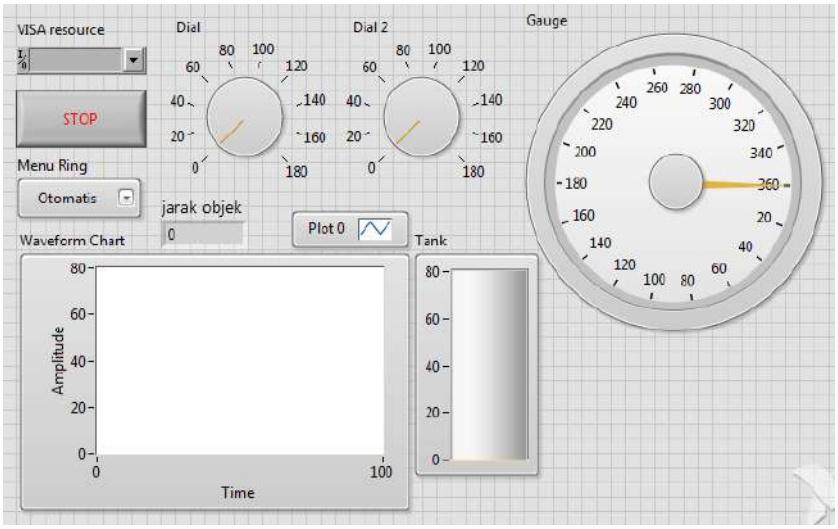


Gambar 12.19 Gabungan program LabVIEW Tahap Pertama dan Kedua, dengan menghilangkan tampilan Voltage

17. Tambahkan objek **Gauge** dari kategori **Silver** untuk memberikan informasi posisi sensor Jarak Infrared dengan lebih jelas. Perbesar ukuran **Gauge**. Ubah nilai maksimumnya dari 10 menjadi 360.

18. Kemudian tempatkan pointer mouse pada strip nilai skala **Gauge** sehingga icon pointer berubah menjadi tanda panah setengah lingkaran. Dengan pointer seperti itu, tarik strip tersebut sehingga angka 360 bertumpukan dengan angka 0.

19. Kemudian geser strip nilai tersebut dengan memutar pointer sehingga nilai 360 berada di samping kanan, seperti berikut:

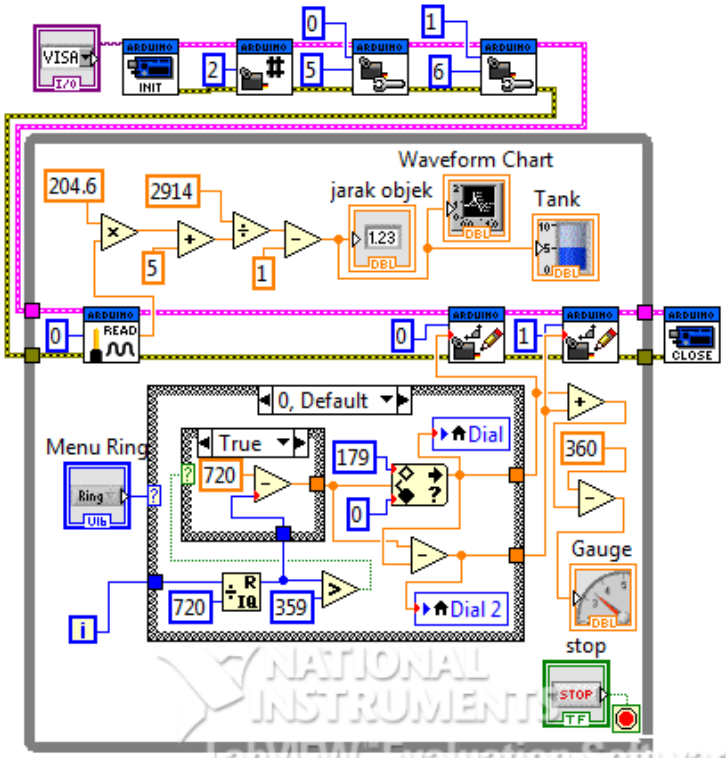


Gambar 12.20 Menambahkan Gauge untuk menampilkan posisi sensor Jarak Infrared, dengan nilai 0-360, yang dimulai dari samping kanan

20. Hubungkan input **Gauge** ini dengan penjumlahan data kedua motor mini servo. Berhubung putaran **Gauge** berlawanan arah dengan putaran motor mini servo, maka kurangkan nilainya pada angka 360, seperti ditunjukkan pada Gambar 12.21.

21. Berikutnya, untuk memetakan jarak objek di sekitar sensor Jarak Infrared dalam bentuk grafik, maka ambil objek **Polar Plot Indicator** dari kategori **Modern**, **Graph**, **Controls** di palet Controls.

22. Karena **Polar Plot Indicator** ini memerlukan data berupa **Array Cluster** dari 2 data, di mana data pertama adalah data jarak dan data kedua adalah data sudut, maka tambahkan fungsi **Initialize Array** untuk menciptakan data **Array Cluster** dengan data awal (0,0) sebanyak 360 buah, sesuai dengan jumlah data sudut (0-360).

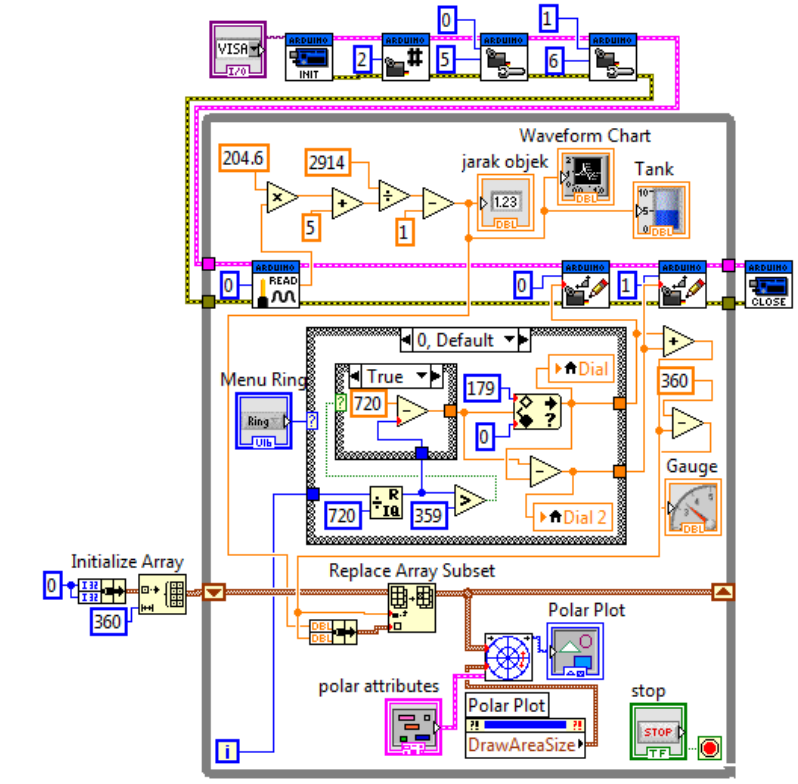


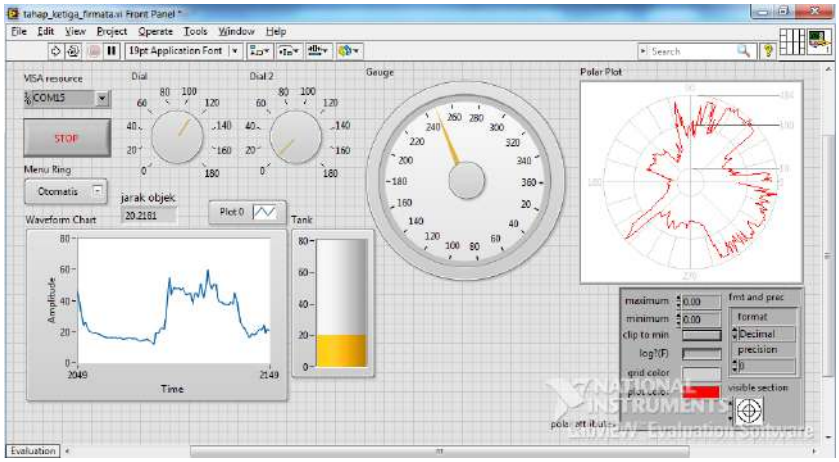
Gambar 12.21 Menghubungkan input icon Gauge dengan penjumlahan data dari kedua motor, dan hasilnya dikurangkan pada angka 360

23. Kemudian tambahkan **Shift Register** untuk menyimpan dan mengumpulkan data jarak dan sudut, di mana nilai awal dari **Shift Register** ini diperoleh dari fungsi **Initialize Array**.

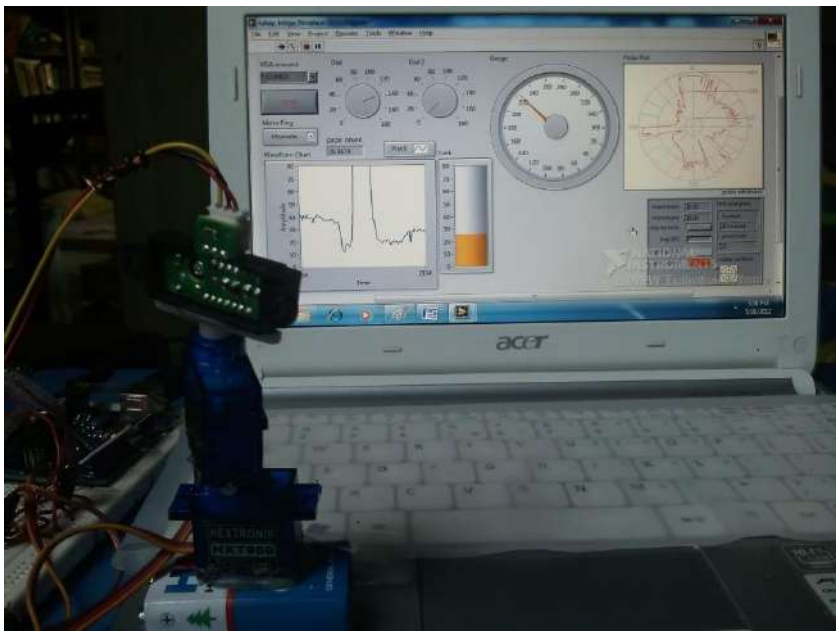
24. Berhubung data harus selalu di-update ketika membaca jarak pada posisi yang sama, maka tambahkan fungsi **Replace Array Subset**.

25. Terakhir, agar tampilan pada **Polar Plot Indicator** dapat diatur, maka klik kanan pada kaki input ***polar attributes*** dan pilih **Create Control**. Kemudian jalankan program dengan menekan tombol **Run**. Ganti warna **grid color** dan **plot color** pada **polar attributes** untuk mendapatkan gambar yang lebih jelas.





Gambar 12.22 Program lengkap Tahap Ketiga



Gambar 12.23 Foto hasil program Tahap Ketiga

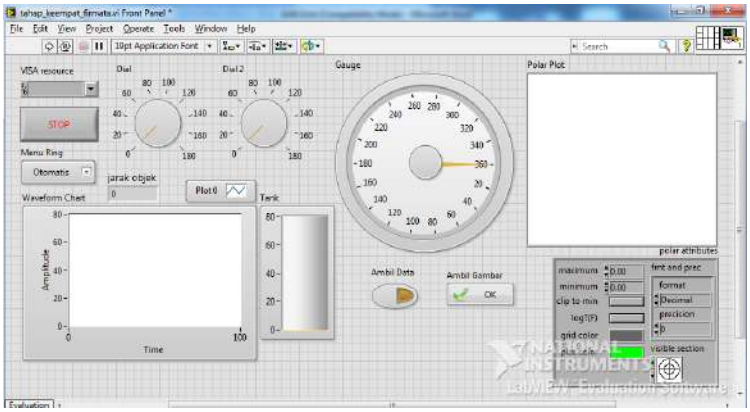
12.7 Tahap Keempat

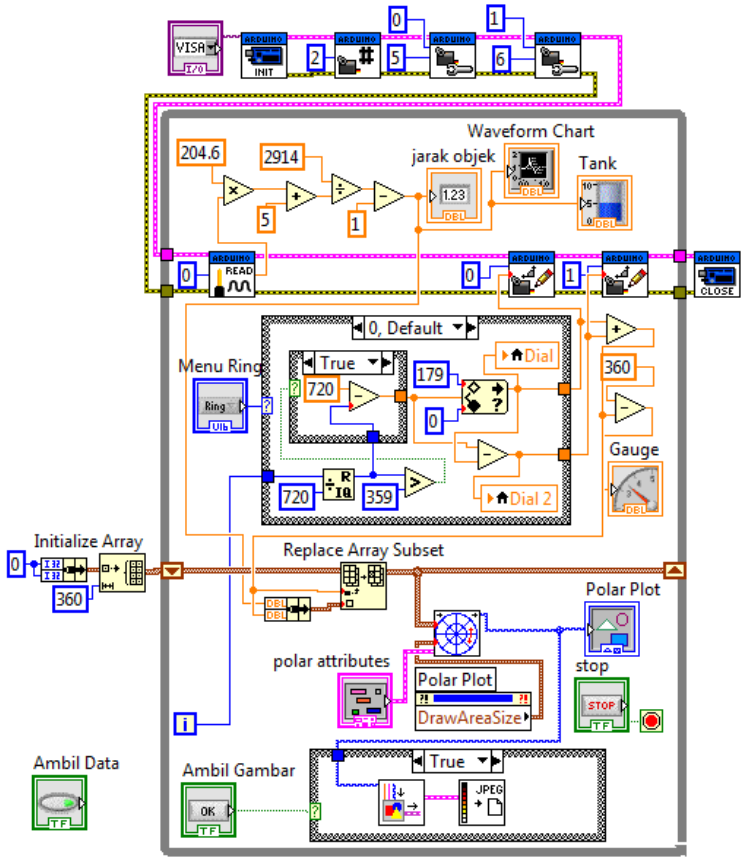
Berikut langkah-langkah pembuatan Tahap Keempat:

11. Ambil objek **Push Button** dan **OK Button** dari kategori **Silver** di palet Controls dan tempatkan pada Front Panel. Ganti nama label **Push Button** dengan **Ambil Data**, dan **OK Button** dengan **Ambil Gambar**.

12. Tombol **Ambil Gambar** digunakan untuk mengambil gambar grafik pada **Polar Plot Indicator** dan menyimpannya dalam sebuah file. Untuk menghasilkan fungsi tersebut, buka Block Diagram, tambahkan **Case Structure** di dalam program dengan terminal **Selector-**nya terhubung ke icon tombol **Ambil Gambar**.

13. Kemudian di dalam blok **Case True**, tambahkan fungsi **Picture to Pixmap** dan **Write JPEG file**. Untuk blok **Case False**, biarkan kosong.





Gambar 12.24 Menambahkan fungsi untuk tombol Ambil Gambar

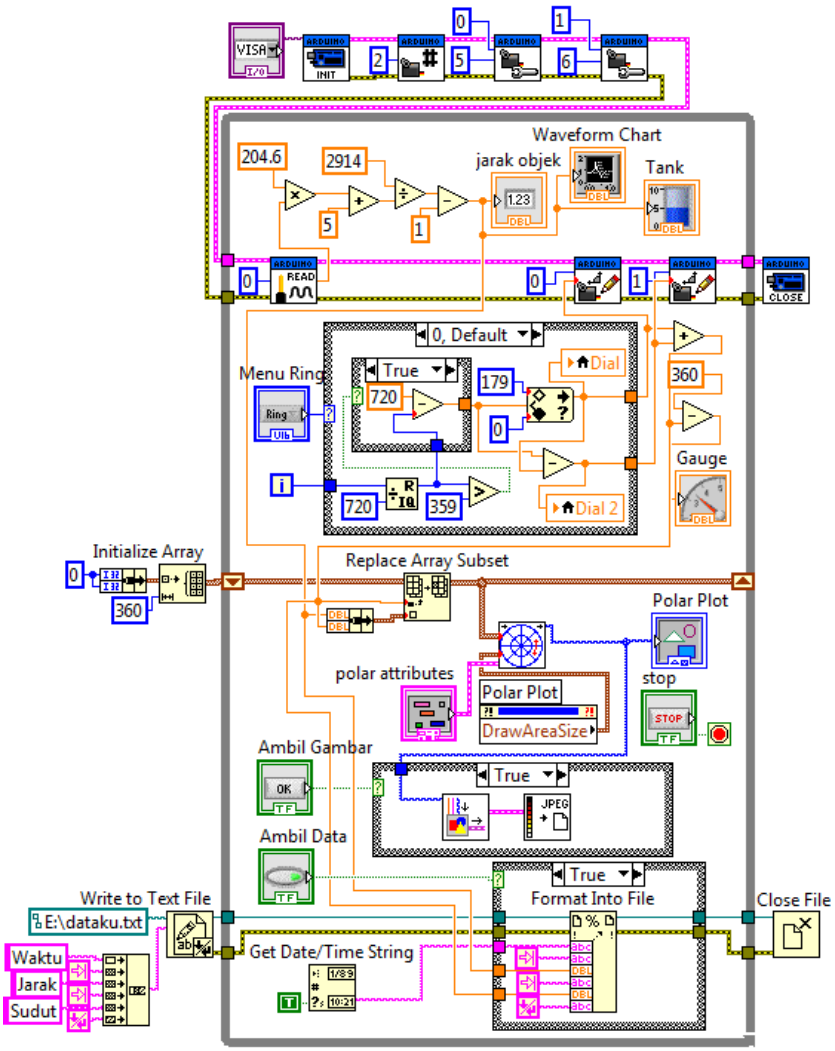
14. Sama seperti tombol **Ambil Gambar**, tombol **Ambil Data** juga dihubungkan dengan terminal **selector Case Structure**, yang mana akan mengaktifkan atau menghilangkan fungsi penyimpanan data. Hanya bedanya, apabila tombol **Ambil Gambar** menggunakan tombol yang sesaat (*momentary*), tombol **Ambil Data** menggunakan jenis tombol yang mengunci (*detent*), agar pengguna dapat menentukan kapan pengambilan data akan dimulai dan dihentikan.

15. Ada 3 fungsi untuk pengambilan data, yaitu fungsi **Write To Text File**, fungsi **Format Into File** dan fungsi **Close File**. Kecuali fungsi **Format Into File**, fungsi yang lain ditempatkan di luar **While Loop**. **Format Into File** inilah yang ditempatkan di blok **Case True**, sedangkan di blok **Case False**, data diteruskan dari kiri ke kanan. Dengan **Format Into File**, selain bisa menyimpan data, keistimewaan yang lain adalah fungsi ini menerima berbagai tipe data.

16. Mula-mula sebuah file akan diciptakan dengan fungsi **Write To Text File**, untuk itu perlu ditambahkan sebuah nama dan lokasi file pada kaki ***file*** (klik kanan kaki ***file*** tersebut dan pilih **Create Constant**). Sebagai contoh di sini, kaki ***file*** tersebut diisi dengan **E:\dataku.txt** (apabila kaki ***file*** ini tidak diisi, maka nantinya akan muncul kotak dialog yang menanyakan nama file dan lokasi penyimpanannya).

17. Kemudian tambahkan 3 buah **String Constant** bertuliskan “waktu”, “jarak” dan “sudut” dengan diselingi **Tab Constant** dan ditutup dengan **End Of Line** dan gabungkan semuanya dengan **Concatenate String** untuk menjadi input kaki ***text*** fungsi **Write To Text File**. Dengan cara ini akan dihasilkan judul kolom data.

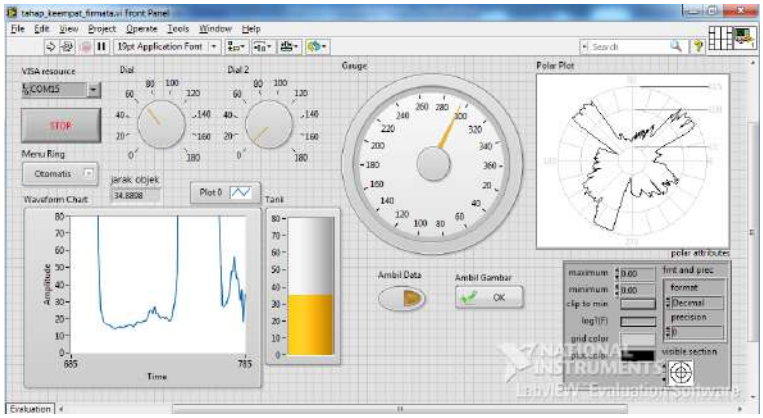
18. Untuk menyimpan isi data sebenarnya, hubungkan data waktu, data jarak dan data sudut ke kaki input fungsi **Format Into File**. Terakhir tutup file dengan fungsi **Close File**.

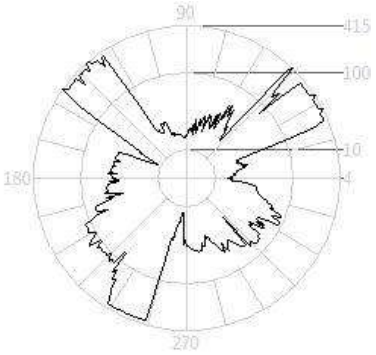


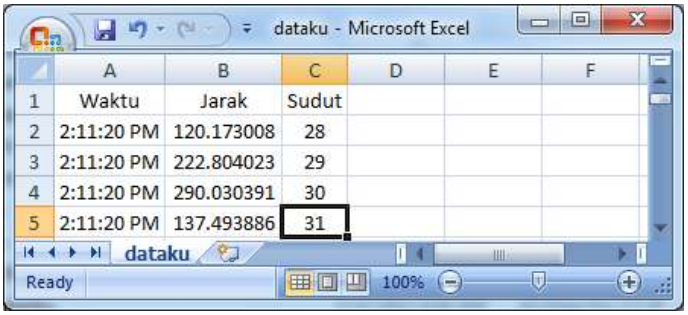
Gambar 12.25 Fungsi penyimpanan data untuk tombol Ambil Data

19. Jalankan program dengan menekan tombol **Run**, kemudian tekan tombol **Ambil Data** hingga menyala. Tunggu hingga **Polar Plot Indicator** menampilkan gambar lingkaran penuh, kemudian tekan tombol **Ambil Gambar**. Isi nama file dan pilih lokasi file pada kotak dialog yang muncul. maka sebuah file gambar JPG diciptakan.

20. Berikutnya hentikan pengambilan data dengan menekan kembali tombol **Ambil Data** hingga padam, maka sebuah file data (**dataku.txt**) di lokasi drive **E** diciptakan.





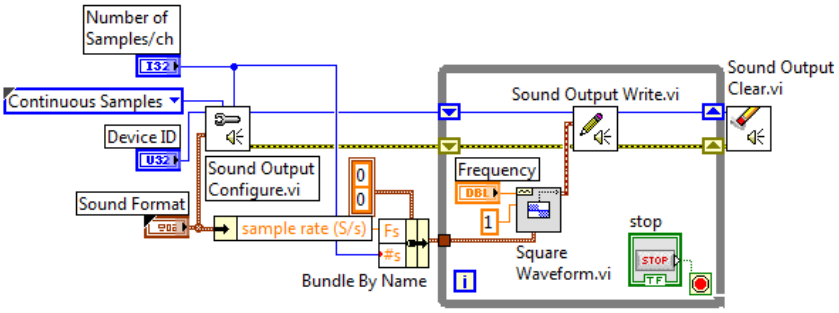


Gambar 12.26 File gambar dan file data dihasilkan pada Tahap Keempat

12.8 Tahap Kelima

Berikut langkah-langkah pembuatan Tahap Kelima:

9. Sama seperti pada Ultrasonik Radar, untuk membangkitkan suara pada Infrared Radar, tambahkan fungsi pembangkitan suara berikut:

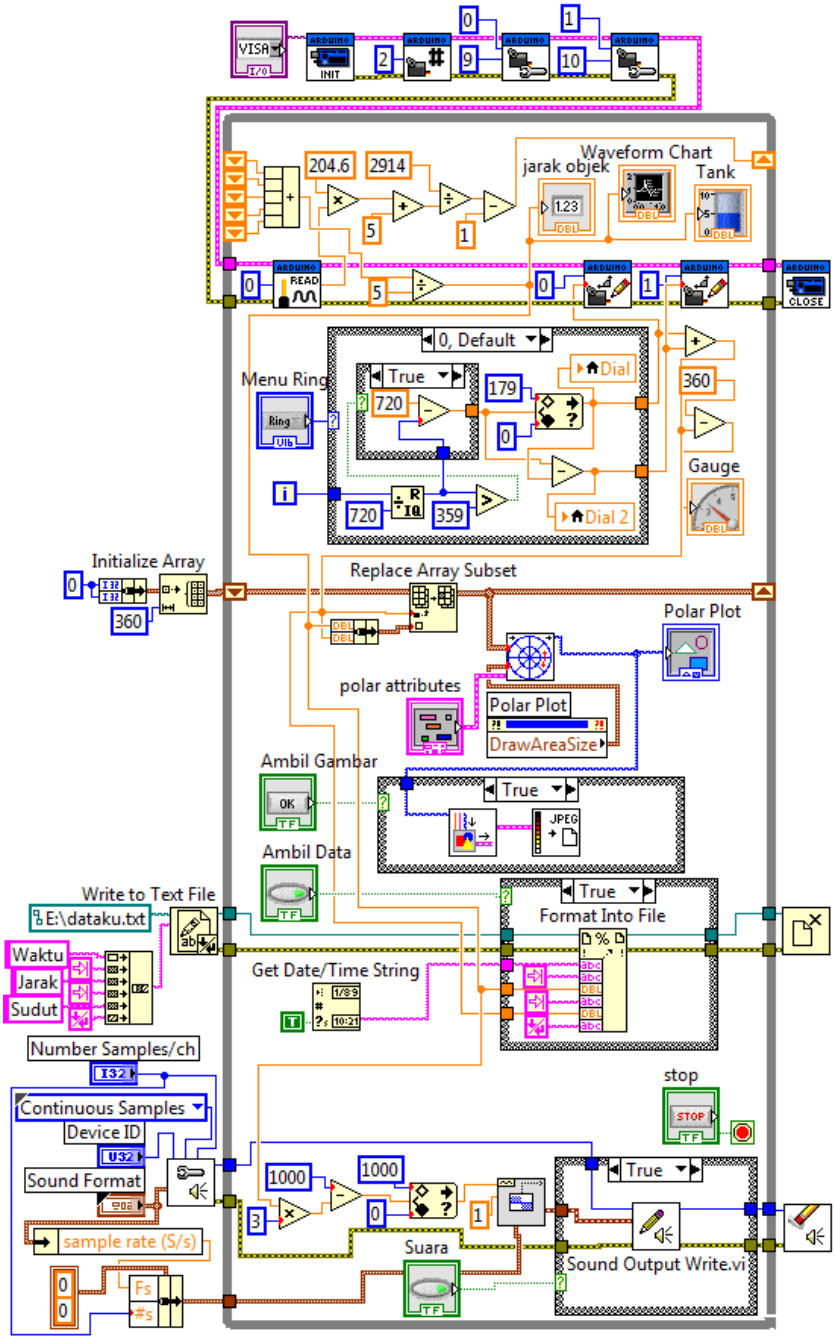


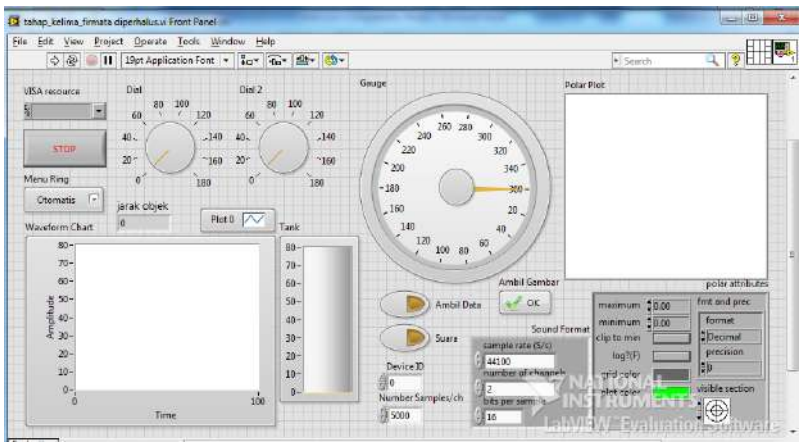
Gambar 12.27 fungsi pembangkitan suara dari Generate Sound.vi

10. Tempatkan fungsi tersebut, dengan masukan nilai frekuensi diambil dari perhitungan jarak objek dikalikan 3, dan hasilnya dikurangkan pada angka 1000. Kemudian agar nilai perhitungan tersebut berada di antara nilai 0 – 1000, maka tambahkan fungsi **In Range & Coerce**.

11. Tambahkan sebuah tombol **Push Button** (**Silver**) yang akan menghidupkan atau mematikan fungsi pembangkitan suara ini. Hubungkan **Push Button** ini dengan terminal **selector** sebuah **Case Structure** yang dipasang pada fungsi **Sound Output Write**.

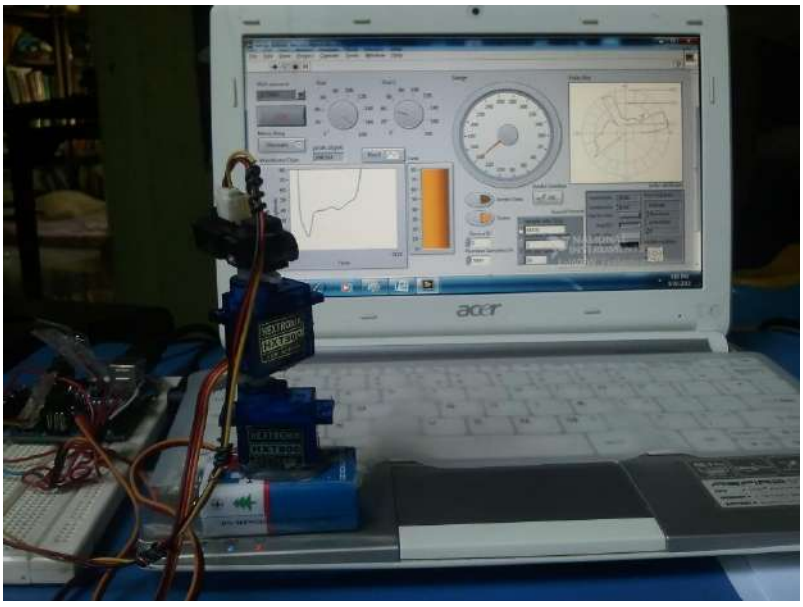
12. Akibat getaran motor, grafik pembacaan jarak menjadi beriak tajam. Untuk membuat agar hasil pembacaan jarak lebih halus grafiknya, maka gunakan rata-rata dari beberapa data hasil pembacaan. Sebagai contoh di sini, dilakukan rata-rata dari 5 data pembacaan, yang dilakukan dengan bantuan **Shift Register** untuk mengumpulkan kelima nilai, **Compound Arithmetic** untuk menjumlahkan semuanya dan fungsi **Divide** untuk membaginya dengan 5.


VISA
INIT
204.6
2914
5
1
jarak objek
Waveform Chart
Tank
READ
CLOSE
0, Default
Menu Ring
True
720
179
Dial
360
Gauge
359
Dial 2
Initialize Array
Replace Array Subset
Polar Plot
polar attributes
DrawAreaSize
Ambil Gambar
OK
JPEG
Ambil Data
Write to Text File
E:\dataku.txt
Waktu
Jarak
Sudut
Format Into File
Get Date/Time String
Number Samples/ch
Continuous Samples
Device ID
Sound Format
sample rate (S/s)
Fs
stop
STOP
1000
Suara
Sound Output Write.vi



Gambar 12.28 Block Diagram dan Front Panel program Tahap Kelima

13. Jalankan program dengan menekan tombol **Run**. Tunggu beberapa saat hingga bekerja. Perhatikan bahwa sekalipun tidak diberi tunda waktu, komunikasi serial dengan Firmata ini berjalan dengan baik.



Gambar 12.29 Foto hasil program Tahap Kelima

Sampai di sini contoh penggunaan komunikasi serial dengan Firmata untuk membuat Infrared Radar telah selesai. Beberapa keuntungan dan kerugian menggunakan Firmata, dibandingkan dengan menggunakan komunikasi serial biasa, telah dituliskan pada SubBab 10.1, dengan beberapa tambahan yang diperoleh selama pembuatan sebagai berikut:

Keuntungan menggunakan **Firmata**:

4. Tidak diperlukan tunda waktu dalam proses pembacaan maupun pengiriman data, sehingga mempersingkat pengolahan data, di samping kecepatan komunikasi yang lebih tinggi (115200 bps), sehingga komunikasi dengan **Firmata** ini lebih efektif untuk komunikasi data dengan jumlah variabel yang lebih banyak.
5. Seperti telah disebutkan, modifikasi program menjadi lebih mudah, karena hanya dilakukan di sisi LabVIEW saja, tanpa perlu memprogram lagi Arduino. Sebagai contoh, misalnya untuk mengendalikan motor servo, diinginkan kaki D5 dan D6 sebagai ganti D9 dan D10. Maka pembaca hanya tinggal mengubah angka 9 dan 10 menjadi 5 dan 6 pada fungsi **Configure Servo** (tentu saja jalur pengawatan rangkaian juga harus diubah mengikuti program).

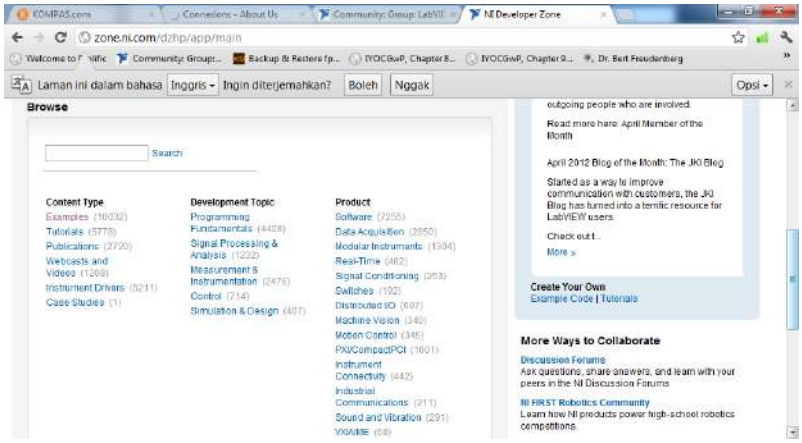
Kerugian menggunakan **Firmata**:

1. Karena pengolahan data yang lebih cepat, membuat konsumsi daya juga menjadi lebih besar. Dari pengalaman penulis, untuk komunikasi dengan Firmata ini, motor servo tidak mau berputar apabila tidak diberi tambahan adaptor yang diberikan ke Arduino. Sedangkan bila tanpa Firmata, hanya dengan suplai arus dari USB komputer, sudah cukup untuk menggerakkan kedua motor servo.

Berhubung keterbatasan halaman buku, maka keuntungan dan kerugian lainnya diserahkan kepada pembaca, agar bisa mengalaminya sendiri.

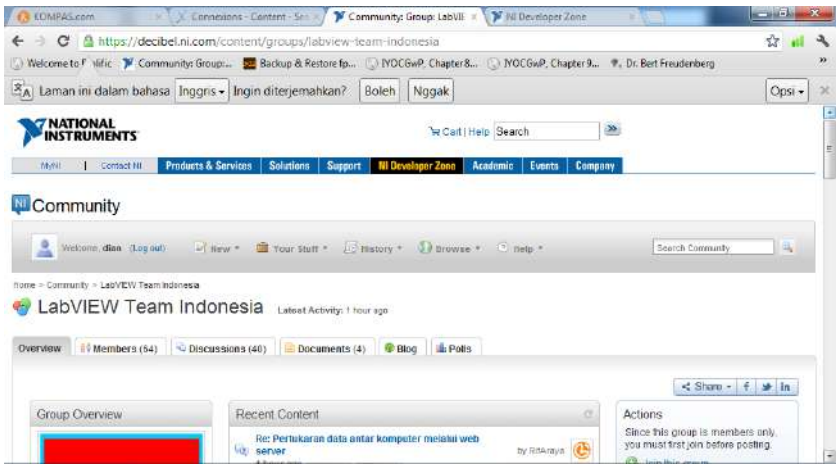
LAMPIRAN
KE MANA SETELAH INI?

Apabila pembaca menginginkan aplikasi yang lebih banyak lagi, pembaca dapat membuka alamat berikut ini: **zone.ni.com**. Tersedia ribuan contoh aplikasi dalam berbagai bidang seperti terlihat dari gambar berikut ini.



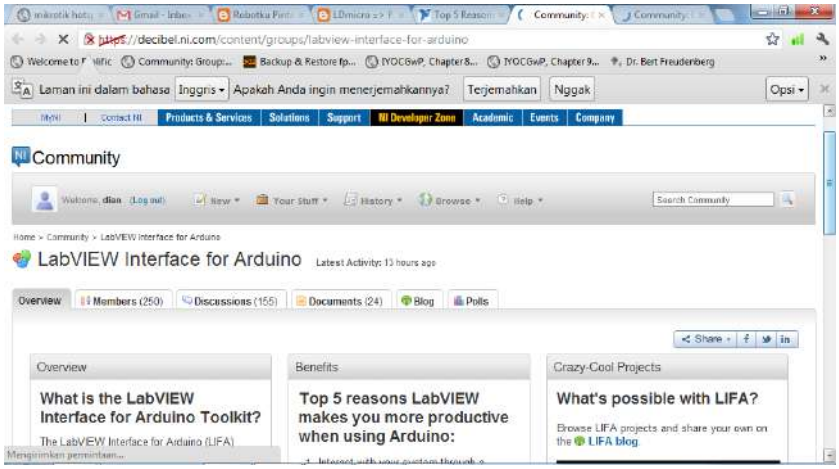
Gambar 13.1 Ribuan contoh aplikasi tersedia di zone.ni.com

Tersedia juga komunitas LabVIEW tim Indonesia di alamat **https://decibel.ni.com/content/groups/labview-team-indonesia/**.
Pembaca dapat bergabung dan menanyakan lebih banyak hal tentang LabVIEW di forum ini.



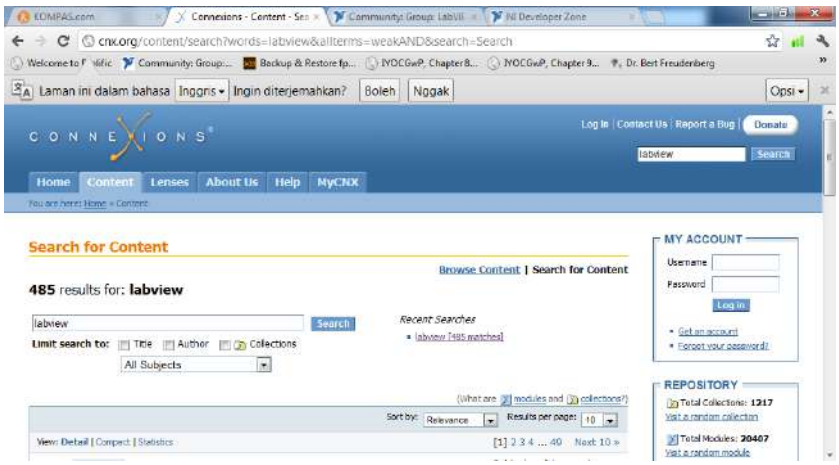
Gambar 13.2 Forum komunitas LabVIEW Team Indonesia

Pembaca juga dapat bergabung di komunitas LabVIEW Arduino, di **https://decibel.ni.com/content/groups/labview-interface-for-arduino/**



Gambar 13.3 Forum komunitas LabVIEW Arduino

Untuk sumber bahan atau buku panduan, selain dari **zone.ni.com**, pembaca juga dapat mencarinya di **CONNEXIONS** di **cnx.org**, dan ketikkan kata kunci LabVIEW pada kolom **Search**.



Gambar 13.4 Tersedia bahan-bahan LabVIEW di Connexions

Untuk mendapatkan komponen-komponen yang digunakan di dalam buku ini, penulis merekomendasikan 2 toko on-line berikut ini, yaitu toko famosa studio dan gerai cerdas.

Gambar 13.5 www.famosastudio.com

Gambar 13.6 www.geraicerdas.com

DAFTAR PUSTAKA

Sumber acuan untuk buku ini adalah sebagai berikut:

1. Brian W. Evans (2008), ***Arduino Programming Notebook***, di bawah Creative Commons Attribution-Share Alike 2.5 License.
2. Joshua Noble (2012), ***Programming Interactivity***, O'Reilly.
3. Massimo Banzi (2011), ***Getting Started with Arduino***, O'Reilly.
4. National Instruments, Malan Shiralkar dkk. (2007), ***LabVIEW Graphical Programming Course***, Connexions, dapat didownload di http://cnx.org/content/col10241/1.4/
5. NI-Tutorial 12879 (2011), ***Top 5 Reasons LabVIEW Makes You More Productive When Using Arduino***, www.ni.com.
6. NI-Tutorial 8534 (2012), ***Advantages of Using LabVIEW in Academic Research***, www.ni.com.
7. Rick Bitter dkk. (2001), ***LabVIEW Advanced Programming Technicques***, CRC Press LLC.

Informasi lebih lanjut mengenai buku ini dapat dilihat di alamat blog berikut ini: **interaksiarduinodanlabview.blogspot.com**